Data MATTERS: Customizing Economic Indices to Measure State Competitiveness

Bogatov, Dmytro dbogatov@wpi.edu

Hennessy, Jillian jrhennessy@wpi.edu

supervised by

Professor Elke Rudensteiner

Abstract

The goal of this project was to expand the functionality of the Massachusetts Technology, Talent, and Economic Reporting System (MATTERS) for the Massachusetts High Technology Council (MHTC), a protechnology advocacy and lobbyist organization, through the addition of two new features an Application Program Interface (API) and the Metric Builder. This API defines a communication protocol between MATTERS and other computational-based systems. We also wrote extensive API documentation. The Metric Builder is a tool that lets users create their own metrics with their own rules out of existing MATTERS metrics. Users are now able to define their own indexes and track individual states' performance.

Executive Summary

Data is a large part of every industry in the business world and big data can help create value in these industries by providing various analytics and performance measures that were once not readily available. Gathering and analyzing this data can drive growth and lead to better decision making across industries. The Massachusetts High Technology Council (MHTC) actively works to maintain Massachusetts' reputation as a competitive business environment in technology based sectors. With this in mind, MHTC worked to create MATTERS, the Massachusetts Technology, Talent, and Economic Reporting System, which is a web dashboard designed that allows users to access and visualize data collected from a large number of other sites together. MATTERS allows users to compare economic, educational, demographic and other aspects across states.

The goal of our project was to introduce two new features to MATTERS that would allow for users to interact with the collected data in more ways than previously available. The ability to display and analyze data in new ways would help to continue MHTC's goal of showing Massachusetts as a competitive state for businesses that are high tech. To achieve this goal we completed the following:

- Create the Metric Builder to allow MHTC along with other registered users to create

their own metric formula using the preexisting metrics stored in MATTERS

- Create a public facing API and corresponding documentation for the site to allow developers a system-friendly way to access the data that MATTERS stores

- Test the features to ensure that they are easy to use for both technical and non-technical users

Before the creation of the Metric Builder page, users could compare states and years by looking at the preexisting metrics found in the MATTERS database. This limited the way users could innteract with the data. MHTC wanted a way to store their own custom metrics which led to the creation of the Metric Builder page. Using the Metric Builder, users now have the ability to choose any number of metrics and assign different weights to indicate how important each metric was compared to the others selected. The Metric Builder feature would build the custom metric formula for the users and normalize the data values, allowing users to visualize their custom metrics just as they would any metric in the site already.As a result of the creation of the Metric Builder, MHTC used our metric formula tool to create four custom metrics that will be displayed on the home page of the MATTERS site.

Meanwhile, the creation of the API helped to provide developers with a means of extracting desired data values in a system friendly way. Previous to implementing a public facing API, the only way to access the raw data would be through parsing the HTML. Having an API for the site, makes the data more widely available to the public. During this implementation two things needed to be considered; a way to have users access all of the data securely and how developers would know how to send proper API requests. To make sure the data was accessed securely, a common practice which we adopted was the creation of an

API key. Users will receive a specific API key identifier and this will give them access to the data while also allowing us to keep track of who is accessing the data values at any time. To make the API easy to use, API documentation was created following standard practices of API documentation for sites.

With these new features now in place, users will be able to interact with the available data in even more ways than previously available from both the user side and the system side. This will continue to make MATTERS a usefu tool for both MHTC and other users, and continue to aid MHTC's goal to make Massachusetts a competitive environment for high-tech business.

Upon completion of this project we developed a set of recommendations for those who will continue to work on MATTERS in the future. The recommendations are as follows:

- MATTERS system

  1. Let users define more complex equations for their user metrics. User should not be limited by simple weighted average formula.

  2. Let users suggest their own data or data sources. Right now MATTERS' administrators and data management team are working on adding data. By involving users in this process MATTERS may have more complete, accurate and up-to-date data.

  3. Role management. Right now the system supports regular users and API users. Each user metric must have one and only one author. It might be a good idea to merge these two user entities and introduce shared user metrics that are not binded to specific users.

- Software development

  1. Refactor the client side and server side code. Since the system has been developed by a number of teams with different design patterns and approaches, there is a significant technical debt. It might be worth spending some time rewriting some parts of the system according to the latest coding and technical standards.

  2. Documentation. Right now there is a steep learning curve for the new developers who start working on the project. Good documentation may dramatically decrease the time it takes for developers to start working on the project.

Acknowledgements

We would like to thank and acknowledge the following for all their help, support, and contributions to this project:

- Massachusetts High Technology Council for making this project possible and working with Worcester Polytechnic Institute to create and grow MATTERS

- Professor Elke Rundensteiner from WPI for her assistance and guidance throughout the entire project

- Caitlin Kuhlman from WPI for her help throughout the project and providing guidance and feedback on all aspects of the project

- Worcester Polytechnic Institute for allowing us the opportunity to work on this MQP project

Contents

List of Figures

List of Tables

Listings

Introduction

With the total amount of data in the world growing steadily at a fast rate, the need to analyze large amounts of data in datasets, also called big data, is a key driver for productivity growth and innovation [10]. Examples of data sets found throughout society include social media accounts, census data, medical records, and ticket sales [1]. With this increase in information often freely available across the web, the Internet will continue to provide a basis for the continued growth of data in the future. Data is a part of every industry and business in today's world from retail to government. Big data can help to create value in these industries by making information more usable, showing more accurate performance measures, and providing detailed analytics on this data that leads to better decision making.

In this data-driven time, it is important to be able to make decisions quickly using the data that is available. Big data visualization is an effective way to present the important information amongst the large amount of data used and helps to drive complex analysis [2]. Big data analysis, it can make otherwise too large amounts of data meaningful to those who looking to interpret significance from it.

The Massachusetts High Technology Council, or MHTC [3], is a group of technological, professional and higher education executives across the state of Massachusetts. They are

comprised of higher education CEOs, senior executives from Massachusetts and more. For over thirty-eight years, MHTC has advocated for policies and programs to create and keep both a healthy and competitive business climate in Massachusetts. Their goal is to establish and keep Massachusetts as a competitive place for successful business and talent building, including in particular with a specific focus on the technology sectors. In part because of the MHTC and its members, who are regarded as the region's most venerable technology association, Massachusetts is regarded as one of the top competitive areas for businesses that are high tech [3].

The Massachusetts Technology, Talent, and Economic Reporting System, also known as MATTERS, was developed by MHTC along with Worcester Polytechnic Institute (WPI) and other institutions as a collaborative effort designed to aid users in understanding a variety of public data in an effort to help make Massachusetts the top location for high technology businesses. MATTERS uses this data collected to both measure and evaluate the current state of Massachusetts compared to other states in the country. It provides policy makers and advocates with easily accessible and search-able dynamic data to assist in their efforts regarding decisions to help retain and grow business in the state. MATTERS integrates a wide range of talent, cost, and economic metrics and state rankings from federal and state government sources, non-profit organizations and media outlets into one location via a data warehouse for its users [4].

Previously to our project, WPI IQP, MQP and graduate student teams worked to develop the MATTERS dashboard as it was when we began working. These previous projects are discussed in the *Background and Related Work Section.* The goal of our Major Qualifying Project (MQP) is to provide additional features to the MATTERS website that will give our

users advanced features to interact with the data and metrics found within the site. These metrics are measures of quantitative assessments which allow people to track performance and make comparisons on the specific data values [5]. One key feature for our MQP team is to develop the Metric Builder as a means for the MHTC to develop and then work with their own *MATTERS* indicators, which are statistics used in order to measure the current conditions of different sectors as well as examine trends and predict future trends of the data, that they have been in the process of creating [6]. In particular, MATTERS users will be able to create an account on the MATTERS site, and be able to build and display the results of their own unique metric formulas for ranking states. Users will also be able to retrieve any of the data points found throughout the site for their own personal use via a standard API, which is "a set of requirements that govern how one application can talk to another" [7]. The API is an important tool to add convenience for developers who wish to access the data and can save users a lot of time [7]. In total, these new features will provide authorized users an opportunity to use and share the data in even more innovative ways, aiding in the MHTC's goal toward making Massachusetts a leading state for high technology business. These two features provide further access to our site's data and additional ways to manipulate and visualize the data to interpret it in new ways. These features also create an expanded user base for MATTERS. Different privileges are offered to different types of users, such as the already existing administrative users, as well as general users who also would like to create their own indicators.

Background and Related Work

The creation of a Metric Builder page and an API for the MATTERS site must leverage as well as seamlessly integrate with the overall MATTERS site, which has been created by several prior project groups at WPI. These students worked to establish the initial features and the design framework for MATTERS to exist. The initial development of MATTERS in Spring of 2014 started with a team of eleven students composed of one undergraduate team completing their Interactive Qualifying Project (IQP), and 7 WPI graduate students. This team of IQP students researched and came to the first core decisions regarding the front-end visualization tools for the MATTERS dashboard [8]. This includes the inclusion of decisions regarding colors, and visualization types such as charts and tables, that would be used to the represent the data within the system in a manner that was both intuitive for the users and a quick way to show analytics results. More importantly, the IQP team conducted a survey of the MHTC board members to determine their familiarity and preferences concerning visual interaction components of dashboards. In parallel, the graduate student teams developed the first back-end version for MATTERS. They developed components to extract the desired data from diverse websites, parse and clean this retrieved data, before then uploading it into a data warehouse. This data would then be called upon to be shown in the various visual

methods for the users [9].

Following this initial front-end and back-end design and implementation of the MAT-TERS dashboard, a team of students working on their Major Qualifying Project (MQP) at WPI improved the administration center and the data integration for the dashboard. This involved the data integration Pipeline Manager, which allows for data to be downloaded from sources and uploaded to the database in order to populate the MATTERS Dashboard with this data. The pipelines require no coding experience to use and made it easy for MHTC members to be able to add new data sources quickly [8]. In addition, the Administration Center's development to allow for MHTC and other administrators who had access to be able to easily upload their desired data and metrics, as well as an intuitive way to view this data on the administrative end. This MQP team was able to integrate 20 new data sources for the MATTERS dashboard, make the Administration Center usable for non-technical users, and improve upon the look-and-feel of the MATTERS dashboard as a whole [9].

2.1   System Architecture

The MATTERS Website consists of two subsystems - the data acquisition and management *admin panel* and the visual-interaction *dashboard*. While the *admin panel* is used by admin-istrators to work with data (basic create, read, update, and delete, or CRUD, operations), in this project we focused on the *dashboard* subsystem. This subsystem follows MVC (Model-View-Controller) architecture to clearly separate data, business and presenting logic. You can see its visual structure in Figure 2.1. Communication between *View* and *Controller* oc-curs through HTTP requests in JSON format. *Controller* talks to *Model* through the stored
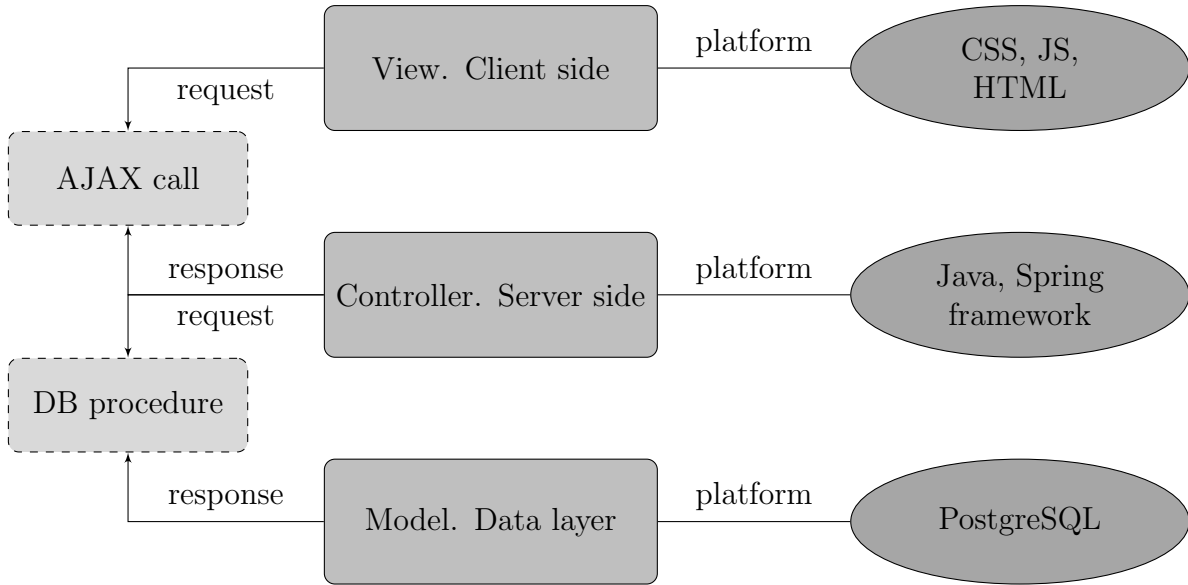
Figure 2.1: MATTERS System architecture

procedures. *View* and *Model* never communicate with each other directly.

2.1.1   Model

The MATTERS' data layer is built with the PostgreSQL database. The main tables are:

Statistics holds metrics' data in the form of a year, state, metric and value tuple (see Table C.1 in Appendix D)

States holds states' metadata like name, abbreviation and if it is a peer state (see Table C.3 in Appendix D)

Metrics holds metrics' metadata like description, trend and category (see Table C.2 in Appendix D)

The access to these tables is provided via stored procedures as a data access layer.

2.1.2   Controller

The business logic of the system resides on a Java-based server. Spring framework is used as a server architecture solution. A front end request that comes to the system goes through controllers which, in turn, invoke services. Services query the data layer and return data stored in models. These data models get pushed back to the front end.

2.1.3   View

The presentation layer is represented by the HTML based website. The client side generates queries to the server via JavaScript AJAX calls, then use the returned data to generate the content. The website has a significant amount of logic on the client side as it needs to build tables, plots and charts. The two main JavaScript libraries used on the client side are jQuery and D3. jQuery makes it easy to manipulate DOM while D3 is used for generating vector based plots and charts.

Visual representation and displays are based on the responsive Twitter Bootstrap 3 framework, which is a collection of CSS classes and small JavaScript snippets.

2.2   Requirements

Before we could embark on creating an API for MATTERS as well as designing the Metric Builder page for the site, we had to conduct research. This included to look into other similar features of different sites and make decisions regarding implementation, design and security features for these aspects of the MATTERS site.

2.2.1   API and Documentation

An API is an "Application Programming Interface". APIs are tools that allow other programs to interact with the software program that the API belongs to without having to give someone complete access to your code base. In a sense they expose some of the internal workings of software to the public but in a way that is limited by what the API chooses to provide. APIs aid in providing a way to share a potentially large amount of data in an efficient way. According to the chief data officer of Philadelphia, Mark Headd, APIs "allow a specific audience to use data more quickly, easily, and efficiently when they are looking to do something specific with the information." APIs are beneficial for not only keeping certain aspects of code private, but also for saving users time. Even in cases of open source programs where all of the code for a program is visible, the code base can be so large that it is inefficient to search through the whole code base for specific data [10].

APIs are beneficial from both a business perspective and for programs that to aim share information and research with many people. An advantage of an API being an automated tool is that it can process a large number of requests without any added cost and work from the developers of the initial site. APIs also present the data requested in a manner that is useful and easily manipulated for the users' purposes, especially if the users could be programmatic access. For APIs whose goals are to share certain data across systems there is a great benefit efficiency wise, as many manual procedures are cut out by the APIs automatic generation. Developers can create code to send requests to a system's API and receive the desired data. This system to system interaction can for example replace the need to manually parse the HTML code for the information they are looking for. By interacting

from one computer to the other essentially, there is room for humans to be working on other tasks [11]. When these tasks are done manually, they waste time and are both laborious and repetitive, as well as being more prone to error; all of these things being costly on the developer end. While it is more time consuming and costly to develop an API initially, the overall benefit once it is developed is apparent and can be reaped long-term [10].

Additionally, APIs not only encourage innovation and the ability to manipulate and extrapolate data by external collaborators, they provide a means to do so that is more secure for the group creating the API. They reduce the risk of how and what data is obtains and encourage good practice for how to properly manage the data available [12].

One important step to gaining the full benefit from an API is making sure that the API is secure. This can be achieved through requiring authentication and authorization of users before they gain access to the API's features. It is good practice to ensure that the users are who they say they are and that they are given permission before using data or parts of a developer's program. This also allows for a record of who is accessing the data, in the event that there is an issue so that a specific user can be tracked down [13].

In addition to creating the API for a site, it is necessary to create good documentation to accompany the API, to make the API as user friendly and simplistic as possible. API Documentation thus must be provided to the developers that details everything that they need to know in a concise manner in order to use the APIs features with other applications. The API Documentation needs to contain the classes, functions and other important aspects of the API [14]. The documentation guide allows developers to easily interact with a site's API and their own code, as many developers style their code in their own unique way.

In order to get the data that is stored in MATTERS, the teams of developers labor to get

this data from over thirty different sites across the web. Some of these sites do not currently have an API, which makes the process of getting the desired data much more difficult. For these sites it takes extra effort just to get the data into the MATTERS site. As a result of this issue seen when trying to interact with other data driven sites, we do not want our site to give developers the same issues. An API will give other developers an easier means of accessing the data stored by MATTERS rather than only seeing the data from the front end of the site.

2.2.2   Metric Builder

The main service that is to be developed by our MQP team is the Metric number of the metrics already available in the MATTERS database and combine their data using a custom formula to Builder page. This service will be available for the registered users on the MATTERS website only. A user will be able to select any create their own personal economic "indicator". The user can then use their created formula to display the states' rankings based on this indicator. This will allow users to see and evaluate states based on what they think is most important as a whole or a combination of various factors, and interpret the data in their own way in a simple manner. By enabling users to create their own data set from all the metrics available and then provide their own weights to each of these data values, we are allowing users to see the competitive advantage that each state has based on how important these metrics are for the user's purpose.

It is important to ensure that different data types can be combined to create one final index value. It is also important to consider the effect that certain data types will have on the final value computed from the user's indicator formula. For example, the MATTERS

database contains rankings, percentages, nominal values, and other data types. Certain data types could end up dominating in a user-created indicator when combined with significantly smaller value data types. There are also complexities involved with certain data types that are represented in the reverse of the majority of the data types. For example, both rankings and some other data types use numerical values in which a lower value indicates a better performance by the state, versus for the majority of other metrics the larger the value is the better a state is performing in that area. This corresponds to the data type incompatibility problem, requiring us to learn about data normalization and transformation.

Finally, there are many metrics that do not contain a value for every single year that other metrics hold a value, the missing value problem. Situations where one metric being used in the user's Metric Builder equation does not contain values for years that another metric being used in the same equation need to be considered. There needs to be a uniform solution for what to do in these instances in a manner that does not deceive the users of the indicator, as this would affect the calculations and subsequent visualizations across years where these issues exist.

These factors need to be taken into consideration to provide accurate and understandable visualizations, when users create their own indicator in the Metric Builder. It is for this reason good to consider special handling or removal of certain data types, as well as potential ways to normalize the final numerical values received from the formulas the users have created in the Metric Builder.

Lastly, we researched various economic indicators in the literature and on other public sites to come up with an idea for how to allow our users to combine sets of data to use for comparisons. One common approach in economic indicators is to use weighted averages

[15]. For example, one of the most popular economic indicators, CPI, or the Consumer Price Index, uses weighted averages of data from hundreds of different consumer goods and services to measure inflation and deflation. Each of the goods considered by the CPI is weighted based on its importance [16].

Weighted averages work by taking each number in a given data set, and multiplying the value by its given weight. This will give you a new value reflecting the product of the data and its weight. Lastly, we then add all of these products together from the data set to get the total value. Then, we add up the total of all the weights. Finally divide the total value by the total weight in order to receive the weighted average of your data set. The weights are used to show the importance that each specific item in a data set has in order to calculate an overall value of all the data together [17].

Methodology of Added Features

3.1 Metric Builder

3.1.1 Back End

For the Metric Builder the first question we have to address is how to model, store and compute a user metric. We distinguish between the following two options:

1. Precompute and then store the user metric value for each metric in the database

   - Pros

     – Fast to retrieve; no computation needed no computation needed

   - Cons

     – Waste of space in the database

     – User metric data remains static; update in the underlying data does not cause an update in the user metric

2. Store a user metric as a metadata only, in the form of a derivation formula, and compute values on-the-fly (on-demand)

- Pros

  - Efficient use of the database space

  - User metric data is always in sync with underlying data

- Cons

  - Might be time-consuming to compute

Looking at the cons of each option, we attempted to approximate how bad they are. Having n users each creating m user metrics each of which has data for k years we have $m \cdot n \cdot k \cdot 50$ entries in the database. This data would be a snapshot. If any data point in the metrics changes, user metrics remain out of sync.

For the second option, it was difficult to approximate time it takes to compute a user metric as it depends on the server load and the number of metrics involved in the user metric. We implemented the second option and benchmarked it with 3 users trying to compute largest possible user metric (all metrics with all states selected). It took around 1.5 seconds to load all 3 user metrics. This drove our design decision towards the second option - computing metrics on-the-fly.

3.1.1.1 Note on normalization and missing values

The basic formula for the user metric is

$$V_y = \frac{\sum_{i \in I} v_{iy} \cdot c_i}{100}$$

where:

- V_y is the value of the user metric for the year y

- I is a set of metrics in the user metric

- v_{iy} is a value of i_{th} metric for the year y

- c_i is a user defined coefficient of i_{th} metric ($c \in [0, 100]$)

However, this does not account for the different magnitudes of different metrics. For example, one metric could be a population and be measured in hundreds of millions, the other one could be taxes, and be measured in percents. If we simply take the average of values in these two metrics, it would not make sense as the value of the higher scale metric (population) will take over the value of the lower scale metric (taxes).

This problem is solved with *normalization* - adjusting values measured on different scales to a notionally common scale.

We used *z-transform* method to normalize data in the user metric. For each year and metric, normalized piece has the following form.

$$\text{value}_{\text{normalized}} = \frac{\text{value}_{\text{denormalized}} - \text{mean}}{\text{standard deviation}}$$

This way we can put together metrics of different scales.

The other decision we had to make was about missing values. Different metrics may have values only for certain years. We decided to take the closest older value for the missing year. If there are no values prior to the missing year, we use the oldest value.

3.1.2 Front End

The main goals for the front end of the Metric Builder feature was to design a webpage that was easy to use, easily accessible, intuitive, and consistent with the design of other pages on
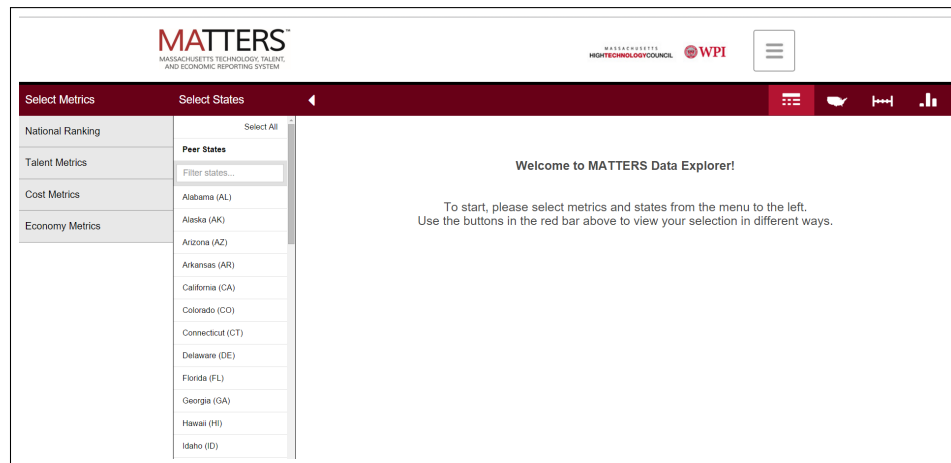
Figure 3.1: Data Explorer Page

the MATTERS site. The Metric Builder feature needed to be a tool that allowed users with any level of technological background to easily build their metric formula from preexisting metrics, assign weights to their chosen metrics, and be able to make changes to their metric formula at any time.

The front end for the Metric Builder page was created through the use of HTML, CSS and JavaScript. To make the page consistent with the rest of the site, code was reused from the Data Explorer page. The Data Explorer page, as seen in Figure 3.1, within MATTERS allows users to select preexisting metrics (from the left dropdown menus) to then visualize this on the site in multiple different visual display types, including a bar chart, line graph, heat map, or table, by selecting the display type on the right side of the black bar via an icon. The framework for the Data Explorer was the page most similar to how we wanted the Metric Builder to look like. Thus we opted to reuse and modify code from that page to create our new feature.

The Data Explorer page allows users to select metrics and states that they would like

Figure 3.2: Metric Builder in use

to view through the use of the left sidebar (See Figure 3.1). Because the intended users for the Metric Builder feature are meant to be authorized users with experience using the MATTERS Data Explorer, we used the sidebar from the Data Explorer as the basis for how the users would select their metrics on the Metric Builder page as well. Since the Metric Builder does not need a user to select states, that part of the sidebar was removed. We also removed rankings from the metric selection column of the sidebar since rankings are manipulations of various other metrics already found in the database. The Metric Builder can be seen in Figure 3.2

Once a user selects the metrics they wish to use in order to create their own metric formula, they have the ability to assign each of them different weights to decide how much
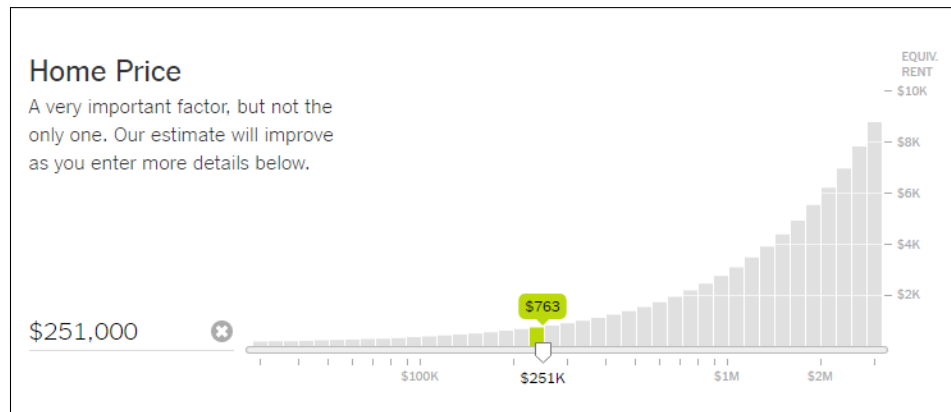
Figure 3.3: Example of New York Times' slider showing Home Price

of a factor each of the metric's data will be in the formula. To adjust the weight of a metric, users can use the sliders that appear under each metric that they have selected. The sliders work horizontally from left to right and allow a weight from zero (which would be the same as not including the metric) to one hundred. Users can click on a section of the slider to move to that weight, click and drag to a specific weight or use the arrow keys to move the slider's value up or down. The idea for the sliders came from a New York Times site which allows users to interactively calculate their rent and other values using visual sliders, as seen in Figure 3.3 [18]. The sliders are visually appealing and simple to use and understand.

The Metric Builder displays options to name the user's custom metric, save their metric and also go back and edit or delete the metric they have created. There is a simple to use button that says "Remove" next to each metric chosen, making it easy for users to change the metrics included in their formula. The input box to name the custom metric and save the changes can be found below all of the chosen metrics and sliders. This was done to help users and encourage them to properly weigh all the selected metrics before leaving saving

their custom metric formula to use, as users typically look at a webpage from top to bottom.

Finally, above all of the metrics selected is a box displaying the basics of their created metric formula. As the users update metric selections and their various weights, the formula will update and show the basics of the math that needs to be done on the back end. This includes showing the users that they are multiplying each data point by the chosen weight before adding each metric's data values together, and that sum is being divided by the total weight as a weighted average approach. This is shown in a simple way to make sure that users can follow along and are aware of exactly what they changing when they adjust their formulas. Once content, each time a user saves and updates their formula they will be prompted and subsequently directed to the Data Explorer page where they can begin using their custom metrics and displaying the data with the various visualization options. The Data Explorer page also has a link back to the Metric Builder in the sidebar so that users can easily go back and forth between visualizing metrics and creating or editing custom metrics at any time. All of the features of the Metric Builder went through testing and the final design can be seen in Section 4.3.

3.2   API

Implementing an API, there are several design decisions to consider, including:

- Do we implement a closed API or open API? If closed, how do we implement authentication.

- Which methods do we support via this API?

- What is the syntax of the query string?

- What is the output format?

3.2.1 Authentication

Although MATTERS system is a public resource, we decided we would like the ability to track users of our API, how and how often. This requirement made us chose a closed API option. Thinking of authentication, we had a few options, including:

- Bind to a particular user

- Simple *API key* (token) authentication

- Complex *handshake* mechanism

- *Handshake* mechanism with random value

Binding to API user t regular user would allow any registered user to access API. Although this kind of authentication is the easiest one to implement we wanted a separate API users.

Simple *API key* authentication is the one when API users are given unique token (long random string) which they need to provide with each request. Based on this token system is able to accept or deny connection and track individual users.

Handshake mechanism with random value involves the following steps

- Client sends in username (not password)

- Server responds by sending back unique random number

- Client encrypts using user's password plus number as key

- Client sends hashed or encrypted value back to server

- Server encrypts using the user's same password plus number as key

- Server compares two hashed or encrypted values, if same then grant access

Provided that the main purpose of the API being closed is tracking an activity for statistical purposes, *API key* authentication was implemented. The user has to apply for this key separately of creating a regular account. The key will uniquely identify the user. If user forgets the key, he or she can recover it through the email.

3.2.2   Methods

Main method implemented in the API is `api/data?`. This method returns all data points for given metrics, states and years.

Since `api/data?` needs specific IDs as arguments, there is a need for supporting methods which show information about metrics and states. So we also implemented `api/metrics?` and `api/states?`.

3.2.3   Query string and output format

The are two options for constructing a query string, namely:

- `api/method?argument1=value1&argument2=value2&...`

- `api/method/value1/value2/...`

To make out API user-friendly, we chose to implement the first option since it clearly shows which value corresponds to which argument. Furthermore, these argument-value pairs may be in any order. Lastly, arguments that have no value can be omitted.

The final query strings are given is Listing 3.1. See Appendix A for an extensive documentation of the API.

Listing 3.1: API methods examples

```
1   /* method */
2   /api/data?metric=&state=&year=&apiKey=
3   /* example */
4   /api/data?metric={16,32}&state=MA&year=*&apiKey=key
5
6   /* method */
7   /api/metrics?apiKey=
8   /* example */
9   /api/metrics?apiKey=key
10
11  /* method */
12  /api/states?apiKey=
13  /* example */
14  /api/states?apiKey=key
```

The output format JSON (JavaScript Object Notation) was chosen as it both comapact and human and computer readable. See Listing D.1 in Appendix ?? for the example of API method output.

3.2.4 Documentation

In order for the developers to know how to make proper API calls, it was necessary to create the API Documentation page for MATTERS. The documentation providers users with full information on the MATTERS API and how to use it, as well as examples of valid API calls and the output that they would return. It is important to ensure that the API Documentation is free of both natural language and code sample errors, as this tool is what developers unfamiliar with using the API rely on to understand how to properly access and extract data [19].

In order to make the documentation easy to follow, the documentation was divided into sections. The main sections of the API Documentation were: an overview, the methods, the parameters, and examples. The layout of the MATTERS API Documentation was based on the API Documentation of Tradestation, which had was very simple and easy to follow [20]. The Tradestation documentation provided a summary, path, parameters, and examples of both an API call and what it would return. An example of their documentation can be found at: http://tradestation.github.io/webapi-docs/en/users/accounts/ [20]. For the MATTERS API, it was necessary to include information on how to retrieve a user's API key. Further we described the proper format to get the data desired for different, years, states, and metrics. The API Documentation sections are explained in the following sections below. The final, complete API Documentation can be found in Appendix 1.

3.2.4.1 Overview of the API

The API Documentation begins with a summary about what the MATTERS API is and what data points the API tool gives users access to. This section also includes an interactive table

of contents which allows users to quickly navigate to a specific section of the documentation to look for information. This allows users to easily find what they are looking for without having to scroll through the entire documentation page. It provides a brief overview of everything that follows.

3.2.4.2 API Key Subsection

Developers without authorization do not have the ability to use the MATTERS API until they have been given an API key. These users are directed to the Contact Us page of the MATTERS site in order to send an email request to become an authorized user with their own key. This section also explains that the API key is used in all API calls in order to validate the user's request.

3.2.4.3 States Subsection

In this section, users are instructed on the proper format needed to retrieve a single state, list of states, or all states. If a user does not know any other information regarding the states in the MATTERS system, they are shown where to find the relevant information needed via an API request.

3.2.4.4 Metrics Subsection

In this section, users are given the proper format in order to request data for a single metric, list of metrics or all metrics in the MATTERS data warehouse. The documentation explains that in order to access data from metrics, the metric parameter values must be the proper metric's ID number. The user is also given the path to all information regarding the metrics,

which will give the user the metric names, ID numbers and details.

3.2.4.5   Years Subsection

Users are given the proper format to request data from specific years. Users may request data for a single year, range of years or all years in the MATTERS data warehouse.

3.2.4.6   Examples of API Calls

Following the parameters and methods of the MATTERS API, the documentation provides the users with two examples of valid API requests and the data each would return. The example requests include the proper format of the parameter values and the various different types of calls the users can make to request specific data. Under the path name showing an API request the option to display the return value of that specific example. Users can view this to see the return data as well as the JSON format it would be given.

3.3   Changes to the database

We added two new tables to the database - **APIUser** and **User Metric**. See Tables 3.1 and 3.2. An example of *Value* in **User Metric** relation:

```
["metricId":"131","coefficient":10,"metricId":"504","coefficient":30]
```

Name	Type	Comment	Reference
Id	*INT*	Unique auto-incremented identifier	none
Name	*VARCHAR*	API user's name	none
ApiKey	*VARCHAR(160)*	Unique random string; used for authentication	none

Table 3.1: **API User** database relation

Name	Type	Comment	Reference
Id	*INT*	Unique auto-incremented identifier	none
Name	*VARCHAR*	User metric's title	none
Value	*TEXT*	JSON formatted list of metric id - coefficient	none
UserID	*INT*	Id of a user - author of the user metric	Id in **User** relation

Table 3.2: **User Metric** database relation

Testing

After the back end code and the initial front end of the Metric Builder, API and API Documentation features were complete, our team decided to receive user feedback to see where we could improve our design. We created a two part user study to first test the usability, intuitiveness, and simplicity of the Metric Builder's user interface and finally to test the clarity and completeness of API Documentation and to make sure there are no errors in the code samples or the natural language text within the documentation.

4.1   User Testing Procedure

All of the users were students from WPI and were asked to complete the user tasks on a laptop provided by our team. Even though the users in our study are not the typical expected users of the MATTERS site, we believe that their input is valuable. If the features of the site are easy to use for users inexperienced with the MATTERS site, then we should expect the features to be easy for the authorized users in the future as they are more experienced with the MATTERS site or with developing APIs. After completing the tasks for each part of the user study, the users filled out a survey asking them to rate the easiness of different aspects of the tasks on a scale of one to five. While the users were completing their tasks, our team

was also there observing their actions in order to get better insight on their clicking behavior and thought process while navigating the site. We also recorded the operating system and browser the users were doing the study on to note any important differences this may bring about.

4.1.1 User Tasks

The user tasks given to the users in the following order and were printed on a sheet of paper so that the users could reference the task details at any time:

User Study Tasks: Part 1, Metric Builder feature

1. *Navigate to the Metric Builder feature*

2. *Please create your own metric formula, which includes* Unemployment Rate, Corporate Income Tax Rate *and* Median Household Income, *with weights* 10, 5, *and* 22 *respectively.*

3. *Name your new metric* "My Metric" *and save it.*

4. *Please find and select your created metric on the Data Explorer page.*

5. *Edit your metric formula so that Unemployment Rate is no longer a part of the metric and Median Household Income has a weight of 15.*

6. *Save your changes to My Metric.*

7. *Please delete My Metric*

8. *You will now receive a short survey about your experience using the Metric Builder.*

User Study Tasks: Part 2, API feature and Documentation

1. *Navigate to the API Documentation page*

2. *Your API Key is* "secret"

3. *Please get the data for the metric* Unemployment Rate *for* Massachusetts *and* Florida *for the year* 2012 *using the API tool and the documentation as a guide*

4. *You will now receive a short survey to complete about your experience using the API and API Documentation.*

The survey that the users completed at the end of each part of the user study can be found in Appendix 2.

4.2   Results

Thirteen users completed the user study and filled out the subsequent surveys. Most users performed the tasks on Google Chrome or Safari and all users performed the tasks on either a Mac Operating System or Windows Operating System. All of the data collected from the user studies can be found in Appendix 2. Table 4.1 shows the average of the users' ratings on a scale of one to five for each question on the survey regarding the Metric Builder feature.

Table 4.2 shows the average of the users' ratings on a scale of one to five for each question on the survey regarding the API Documentation.

Our team considered an average rating of above four to be a good score, meaning that the specific task was easy to perform with our user interface. The only tasks that did not score above our desired rating was for How easy was it to edit 'My Metric' to make the appropriate changes? and How easy was it to find the API Documentation?. User feedback on the open ended survey responses, as well as observations made by our team during the

Survey Question	Average User Rating
How easy was it to find the Metric Builder?	4.385
How easy was it to select the metrics "Unemployment Rate", "Corporate Income Tax Rate", and "Median Household Income" in the Metric Builder?	4.308
How easy was it to weigh the metrics accordingly?	4.615
How easy was it to name and save your metric?	5.000
How easy was it to find My Metric on the Data Explorer page?	4.231
How easy was it to edit My Metric to make the appropriate changes?	3.846
How easy was it to remove My Metric?	5

Table 4.1: Average ratings for the user study on the Metric Builder

Survey Question	Average User Rating
How easy was it to find the API Documentation?	3.250
How useful is the API Documentation?	4.167

Table 4.2: Average ratings for the user study on the API Documentation

user studies were able to confirm that these two tasks were the most difficult or confusing and provided insight as to how to improve it.

For the Metric Builder, users noted that they wanted an option to allow them to edit their metric from the Data Explorer page. Our observations confirmed that while on the Data Explorer page, users were actively searching for an edit button while they were viewing their custom made metric. Many times the users would find the "Create new metric link quickly, which if clicked would bring the user to the Metric Builder page where they could also edit their metric, but were hesitant to click the link because they were not interested in starting over with a new metric.

For the API Documentation, many users commented on their difficulty finding the link
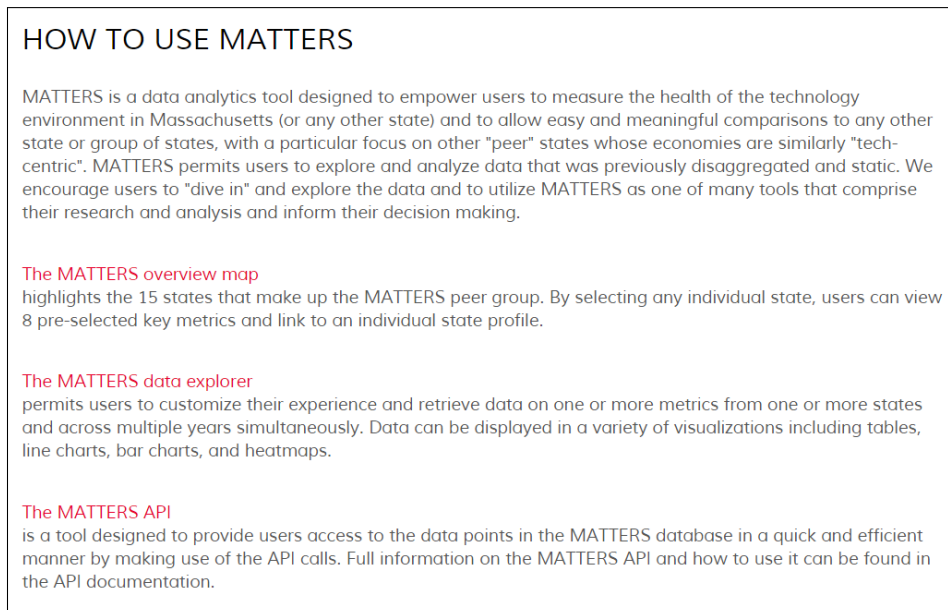
Figure 4.1: Initial location of the API Documentation link

for the API Documentation page. Ultimately, twelve out of the thirteen users did not have much of an issue eventually navigating to the API Documentation link found on the How to use MATTERS page as seen in Figure 4.1; however, it was more time consuming than it should have been and many users clicked on multiple other links before navigating to the correct page. The users who stated that they did not know what an API was mainly looked for the API Documentation link in the main drop-down menu of the site before navigating to the About page, followed by clicking on the link to How to use MATTERS and finally finding the API section which led them to the documentation. Most of the users who were aware of what an API was, ultimately navigated to the API Documentation page the same way; however, many of them scrolled to the bottom of the home page of the site looking for a link similar to the Developers links found at the bottom of many webpages.

Figure 4.2: Footer

4.3   Changes made

As a result of the user ratings, comments and our observations made while completing the user study tasks, we made a few front end changes to make our features easier to use and easier to find. This included adding a link to the API Documentation in the footer of the site in addition to the link on the "How to use MATTERS" page, as seen in Figure 4.2, and as shown in Figure 4.3, changing the Create new metric link on the Data Explorer page to mention that the link would also allow users to edit their metric.

These changes led to the final design of the front end of the Metric Builder and API Documentation which can be seen in the figures below. Figures 4.4 and 4.5 show the Metric Builder page when a user first visits the page and after a user has selected and assigned weights to some of the metrics from the Metric Selection column of the sidebar. Once the user has decided to save their metric formula, if they choose to go back and edit any part of it, the "Save" button will be replaced by an "Update" button and options are added to "Cancel" any changes or remove the custom metric completely. These additional options can all be seen in Figure 4.6. Finally, there were no changes that needed to made to the API Documentation itself as a result of the user testing. Figure 4.7, shows the API Documentation
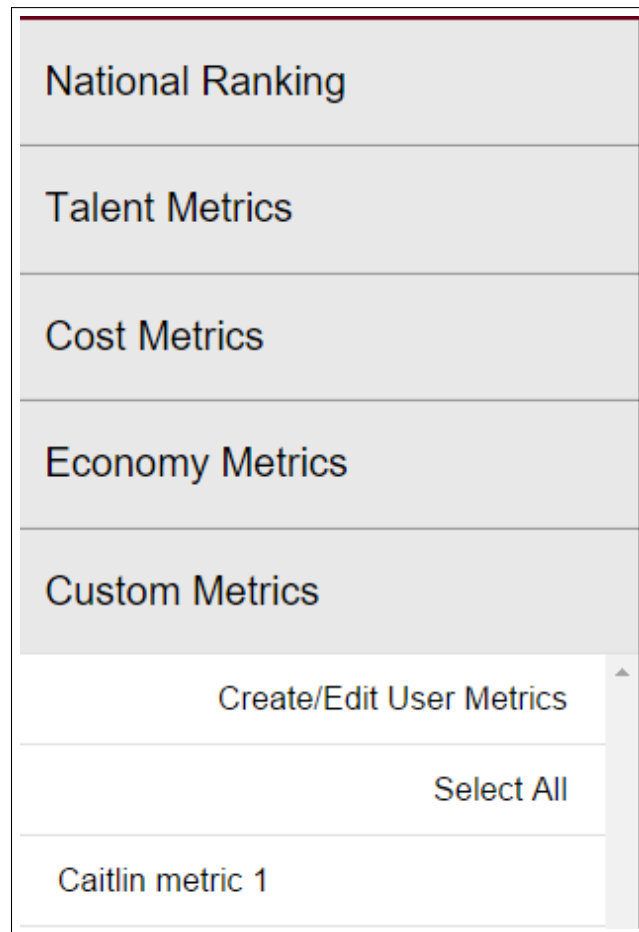
Figure 4.3: Data Explorer sidebar with edit link

page as it appears when a user first navigate to the page from either the footer or the "How To Use MATTERS" page.
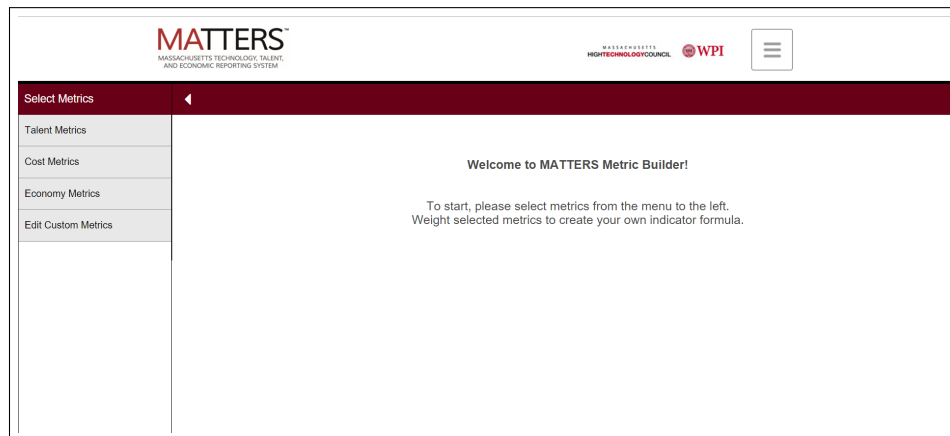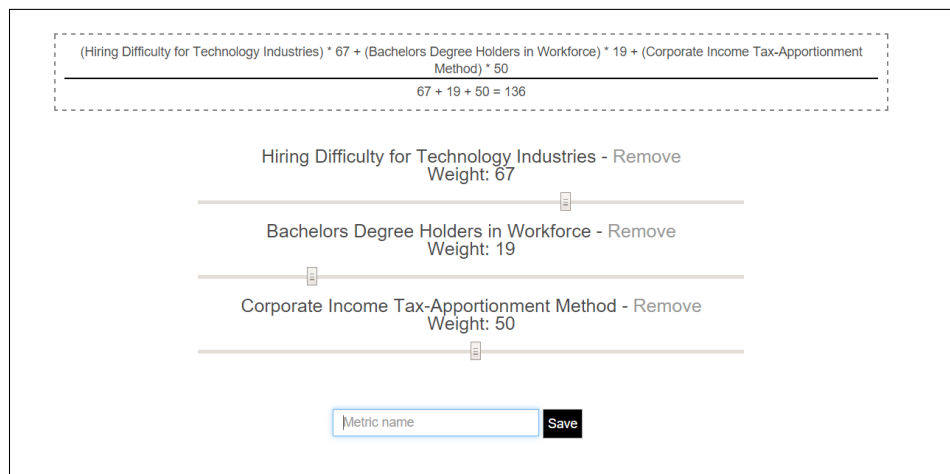
Figure 4.4: Metric Builder

Figure 4.5: Metric Builder's canvas
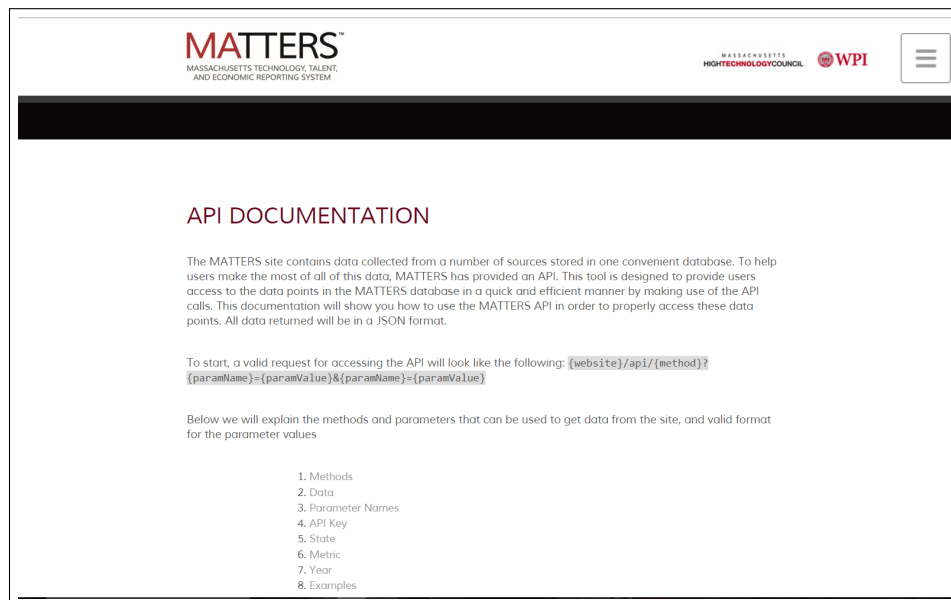
Figure 4.6: "Save" menu

Figure 4.7: API Documentation page

Conclusion

In this project we have built two services for the MATTERS site. The API provides a way for other computer-based systems to work with the MATTERS system. Specifically, it is now possible to query MATTERS data directly, bypassing the presentation layer. The other feature is a Metric Builder. This subsystem allows users to create their own unique metrics based on the existing metrics. Users are able to work with their custom metrics seamlessly in the Data Explorer as if they were just regular ones.

5.1 Recommendations

Recommendations for other teams, either MQP or other teams, working on improving the MATTERS system in the future are as follows:

- MATTERS system

  1. Let users define more complex equations for their user metrics. User should not be limited by simple weighted average formula.

  2. Let users suggest their own data or data sources. Right now MATTERS' administrators and data management team are working on adding data. By involving

48

users in this process MATTERS may have more complete, accurate and up-to-date data.

3. Role management. Right now the system supports regular users and API users. Each user metric must have one and only one author. It might be a good idea to merge these two user entities and introduce shared user metrics that are not binded to specific users.

- Software development

1. Refactor the client side and server side code. Since the system has been developed by a number of teams with different design patterns and approaches, there is a significant technical debt. It might be worth spending some time rewriting some parts of the system according to the latest coding and technical standards.

2. Documentation. Right now there is a steep learning curve for the new developers who start working on the project. Good documentation may dramatically decrease the time it takes for developers to start working on the project.

Appendices

API Documentation

API Documentation can be found in the *Appendices* folder under the name *API Documentation.pdf*.

User Study

User study cconsent form can be found in the *Appendices* folder under the name *User Study Consent Form.pdf.*

Script for investigators can be found in the *Appendices* folder under the name *Script For Investigators.pdf.*

Survey can be found in the *Appendices* folder under the name *Survey.pdf.*

Full user survey results and comments can be found in the *Appendices* folder under the name *User Results.xlsx.*

Database Sample Data

Column Name	Type	Sample Value
StateId	*INT*	3
MetricId	*INT*	14
Year	*INT*	2013
Value	*DOUBLE*	7000
DateAdded	*TIMESTAMP*	2014-04-19 18:28:34.63874

Table C.1: **Statistics** database relation sample data

Column Name	Type	Sample Value
Id	*INT*	12
Name	*VARCHAR*	Capital Gains Tax Rate
Visible	*BOOLEAN*	TRUE
IsCalculated	*BOOLEAN*	FALSE
DatType	*VARCHAR*	percentage
DisplayName	*VARCHAR*	Capital Gains Tax Rate
URL	*VARCHAR*	KPMG/Tax Foundation
Source	*VARCHAR*	KPMG/Tax Foundation
DisplayOrder	*INT*	7
TrendType	*VARCHAR*	reversed

Table C.2: **Metrics** database relation sample data

Column Name	Type	Sample Value
Id	*INT*	21
Abbreviation	*VARCHAR*	MA
Name	*VARCHAR*	Massachusetts
IsPeerState	*BOOLEAN*	TRUE

Table C.3: **States** database relation sample data

API Output Sample

Listing D.1: API Data Method Output Sample

```
1  GET: api/data?state={AL,MA,NH}&metric={18}&year=2012&apiKey=secret
2
3  [
4    {
5      "state" : "Alabama",
6      "metrics" :
7        [
8          {
9            "MetricName" : "Total Tech Employment",
10           "Data" :
11             [
12               {
13                 "Year" : 2012,
14                 "Value" : 209044.0
15               }
16             ]
17         }
18       ]
19   },
20   {
21     "state" : "Massachusetts",
22     "metrics" :
23       [
24         {
```

```
25          "MetricName" : "Total Tech Employment",
26          "Data" :
27            [
28                {
29                  "Year" : 2012,
30                  "Value" : 508666.0
31                }
32            ]
33        }
34      ]
35  } ,
36  {
37    "state" : "New Hampshire",
38    "metrics" :
39      [
40        {
41          "MetricName" : "Total Tech Employment",
42          "Data" :
43            [
44                {
45                  "Year" : 2012,
46                  "Value" : 74200.0
47                }
48            ]
49        }
50      ]
51  }
52 ]
```

References

[1] Examples of big data projects.

[2] Big data: The next frontier for innovation, competition, and productivity.

[3] About mhtc.

[4] About matters.

[5] Metrics.

[6] Indicator.

[7] Brian Proffitt. What apis are and why they're important.

[8] Alex Fortier, Long Hoang Nguyen Duc, Kevin Mee, and Westley Russell. Advancing massachusetts technology, talent and economy reporting system (matters). `http://www.wpi.edu/Pubs/E-project/Available/E-project-042915-094249/unrestricted/Advancing_MATTERS.pdf`.

[9] Nilesh Patel, Stefan Gvozdenovic, and Ted Meyer. State cost competitiveness dashboard. `http://www.wpi.edu/Pubs/E-project/Available/`

```
E-project-050814-222635/unrestricted/Final_report_5.6.2014_
FINAL-revisions.pdf.
```

[10] Greg Sleter. What's an api and why do you need one?

[11] Erick Schonfeld. Amazon takes the humans out of fulfillment with new api.

[12] Brian Proffitt. What apis are and why they're important.

[13] Andy McGregor, David Flanders, and Malcolm Ramsey. The advantage of apis.

[14] Gregory Koberger. The importance of good api documentation: Creating meaningful partnerships for businesses.

[15] Weighted average.

[16] Consumer price index - cpi.

[17] The top 10 economic indicators: What to watch and why.

[18] Mike Bostock, Shan Carter, and Archie Tse. Is it better to rent or buy.

[19] Hao Zhong and Zhendong Su. Detecting api documentation errors.

[20] Get user accounts.