A Data-Agnostic Dashboard Visualization Framework With Customizable Layout For e-Commerce Data Analytics

Arun Sai K
Department of Computer Science and Engineering,
National Institute of Technology
Tiruchirappalli, India
+91-9731014249
arunsaikrish@gmail.com

Sriram Venkatapathy
Teritree Technologies Pvt Ltd
Bangalore
India
+91-9945512893
sriram.v@teritree.com

ABSTRACT
Different e-commerce vendors have different data-visualization requirements for their analytics dashboards. For instance, the format of data-input, the chart to be rendered and the layout of the dashboard vary widely and depend on the vendor's requirements. This framework was developed to automate the process of visualizing data on a dashboard with a customizable layout. The framework is independent of the type of input data and it renders charts based on the configuration set by the user. It also provides the flexibility to render charts using multiple charting libraries and third party APIs. Another key feature in this framework is the customizable layout of the charts rendered on the dashboard.

In this framework, data-agnostic visualization in customizable layouts is achieved by the use of an XML file, which stores all the configuration settings of the dashboard.

Data to be visualized is fed into the framework in one of the supported data-input formats. This input also contains the type of chart to be rendered and other meta-data. Charts and graphs can be rendered from multiple sources, which include charting libraries and third party APIs like Google Charts. The meta-data stored in the XML file contains information regarding the charts to be rendered and their sources.

The layout of the dashboard is specified using various parameters such as position and size of the charts to be rendered and customized using appropriate tags and values in the XML file. This framework was developed and deployed at Teritree Technologies Pvt Ltd and has received a very good customer feedback.

Categories and Subject Descriptors
D.2.2 [**Software Engineering**]: Design Tools and Techniques – *modules and interfaces.*

D.2.m [**Software Engineering**]: Miscellaneous – *reusable software.*

General Terms
Design

Keywords
Data-Visualization, Dashboard Framework, E-Commerce, Data Analytics, Customizable Layout

1. INTRODUCTION
E-commerce has grown substantially in the last few years with an increasing number of people preferring to shop online. This is a multi-billion dollar industry with many prominent players competing to gain a significant market-share. There are a variety of factors such as price points, delivery times and after sales service, which affect the performance of an e-commerce vendor. An e-commerce vendor needs to constantly review and analyze various data points such as user-behavior statistics and product-related statistics to make good strategic decisions and stay competitive in the market.

One interesting user-behavior is adding a product to the shopping cart but not completing the transaction. This behavior gives a clear indication of the user liking the product, and a follow-up from the vendor to understand and address the customer's concerns may result in a sale. For instance, let us assume that a user had added a product to the shopping cart, but did not go through with the transaction due to high pricing. If this key piece of information is captured and analyzed, the vendor can consider reducing the price of the product or get back to this customer at a later point in time, when the product's price has dropped, to make a sale. Another example of this would be when the number of consumers viewing a certain product is very low. If this piece of valuable information is captured, the vendor can try to increase sales of the product by giving it more advertisements or offering good discounts or offers on the same.

Once the required data has been collected, visualizing it effectively is the next key step. Different e-commerce vendors may have different visualization requirements to analyze their data effectively. The data may be served in different formats such as XML (Extensible Markup Language) and JSON (JavaScript Object Notation) and may have to be visualized using different types of charts such as pie charts, line charts, column charts etc, on a dashboard portal. Different users might prefer different layouts for their charts on the dashboard portal. The fact that the data may be stored in different database platforms such as MySQL, MongoDB, SQLite etc, adds to the complexity of this problem.

This paper proposes a solution for the above-mentioned problem through the development of a data-agnostic dashboard visualization framework, which enables an e-commerce vendor to customize the layout, format of data-input and the type of charts used in the dashboard. This proposed solution was implemented successfully and the framework developed is currently being used by Teritree Technologies Pvt Ltd.

Related work to this problem is presented in the next section. Details of our approach and implementation are discussed in Section 3. The results are discussed in Section 4, followed by the conclusion in Section 5.

2. RELATED WORK
In recent years, there has been a lot of research on data mining and data analytics with respect to user-behavior patterns on the internet. Several techniques to cluster large amounts of customer information into segments have been developed using the K-means algorithm [3]. Other similar techniques, which use the web logs of e-commerce sites, have also been reported [16]. There has also been some research in applying web-mining techniques for discovering meaningful usage patterns from web-usage data [8]. A few dashboard framework solutions are currently available in the market. The Splunk Enterprise Framework [14] provides an enterprise solution for the dashboard needs of large organizations. Other tools such as Sonarsource [13], Logic Monitor [11] and RazorFlow [12] also provide solutions for dashboard visualization. Google Charts [6], Fusioncharts [5], Highcharts JS [7], Flot [4], jqPlot [9], jQuery Sparklines [10] and Chart.js [2] are some of the charting

DOI: 10.1145/2632434.2632444 http://doi.acm.org/10.1145/2632434.2632444

libraries and APIs that have been developed recently. These libraries and APIs are based on Javascript or jQuery.

The framework discussed in this paper, gives a higher degree of flexibility in the input data formats and charting options as compared to the existing solutions discussed above.

3. PROPOSED METHODOLOGY

In this section, we present a unique and efficient design solution to visualization, which is customizable and also data-agnostic. Our framework has five components (see Figure 1), namely:

- The Dashboard Portal
- Customizable Data-Visualization and Flexible Layout Configuration - XML File
- Wrapper API for data retrieval
- Charting Libraries and APIs
- The Core : Data-Retrieving API and Data-Visualization API

These components are explained in detail in the following sub-sections.

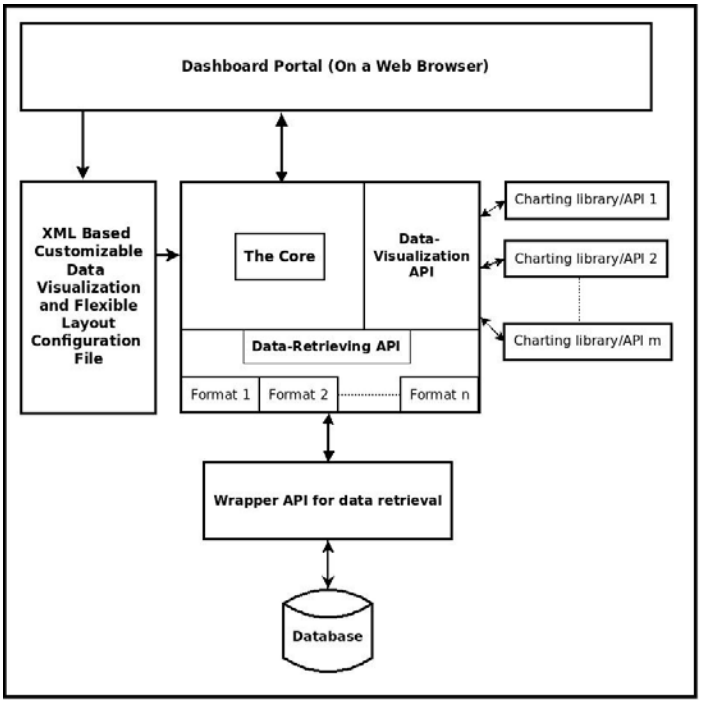

Figure 1. Structure of the Framework

3.1 The Dashboard Portal

The dashboard portal is the UI (User Interface) component of the framework, which is viewed on a web browser. It displays data in the form of charts, graphs and other such visual media, which enables the viewer to gain insights. These insights help the viewer in making meaningful and well-informed business decisions. The user can customize the entire dashboard including its layout and the types of charts and graphs used, through the GUI (Graphical User Interface) provided as a part of the dashboard portal. These customization choices are stored in the configuration XML file.

Sample dashboard portals generated using this framework, are shown in Figure 2.

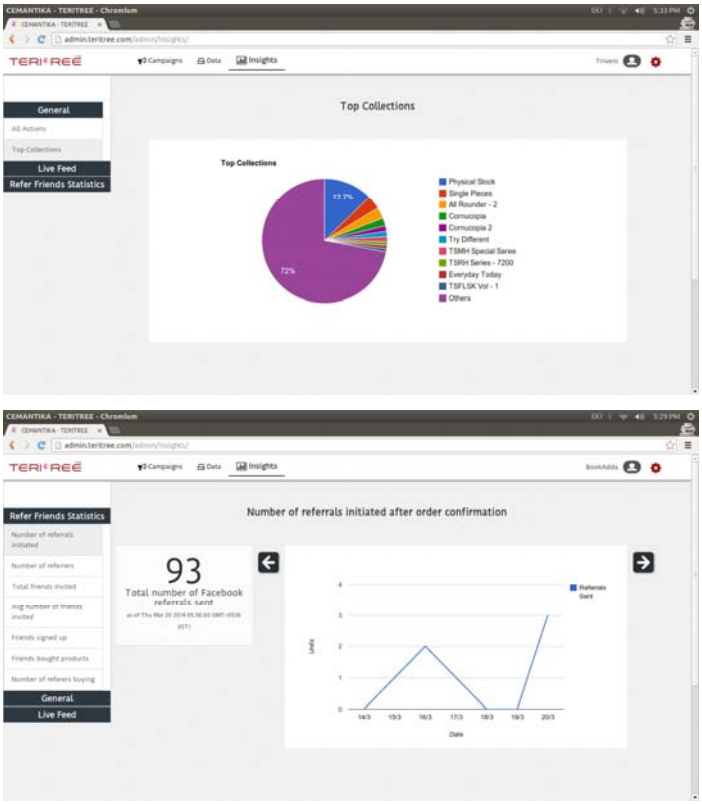

Figure 2. Sample dashboard portals built using the framework

3.2 Customizable Data-Visualization and Flexible Layout Configuration - XML File

Customizable data-visualization and flexible layout is supported in the framework through a unique XML configuration file. XML is a markup language that defines a set of rules for encoding documents in a specified format that is readable by both humans and machines. The different customization aspects supported by the framework are shown in Figure 3. These customizations are specified by the user through the use of special pre-defined tags of the framework.

A sample XML configuration file with specification of required customizations is shown in Figure 4 and the corresponding dashboard generated is shown in Figure 5.

Only tag names defined in Table 1 can be used in the configuration XML file. The configuration XML file supports specifications for grouping of charts into categories. Each of these groupings has its own individual layout specifications. A particular chart or graph can also be enabled or disabled through the use of proper tags and values in the configuration XML file. The various tag names supported in the configuration XML file and their respective functionalities are listed in Table 1.

3.3 Wrapper API for Data Retrieval

Vendors need to provide a wrapper API to the framework, for it to retrieve data from their databases. The wrapper API when called from the data-retrieving API of the framework fetches the required data from the vendor's database and serves it to the framework. Figures 1 and 6 show how the wrapper API is used in this framework. The API calls to be made to retrieve the data are specified in the configuration XML file.

The vendor’s database stores all the collected user-behavior data. This database could be running on any database platform like MySQL, MongoDB, SQLite etc.

DOI: 10.1145/2632434.2632444 http://doi.acm.org/10.1145/2632434.2632444

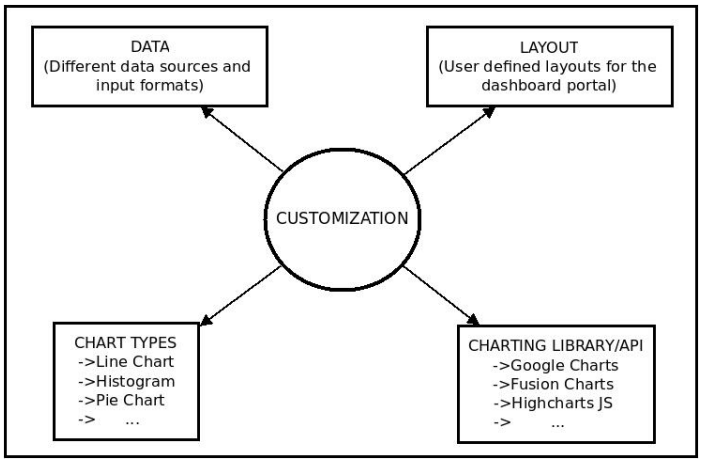

Figure 3. Aspects of customization in the dashboard

Table 1. XML tags and their functionality

Tag	Functionality
<screen>	Indicates the beginning of the layout description of a new screen. One screen can have many rows and each row can hold multiple charts.
<enable>	Indicates if the screen is to be rendered or not. The values it accepts are ON and OFF.
<name>	Specifies the name of the screen to be displayed on the dashboard portal.
<description>	Describes the screen that is to be displayed on the dashboard.
<row>	This tag indicates the beginning of a new row of charts on the screen.
<height>	It specifies the height of each row as a percentage of the total height of the screen.
<chart>	Indicates the beginning of description of the chart. It is followed by all the configuration details of a chart including chart type, source, width and plots to be displayed in the chart. A single chart may contain multiple plots.
<width>	Specifies the width of each chart as a percentage of the total width of the row containing it.
<chartsource>	Specifies the library or API from which the chart is to be rendered. For example, it could be Google Charts [6] or FusionCharts [5].
<charttype>	Specifies the type of the chart. For example, it could be a line chart or a pie chart.
<plot>	It refers to the data point to be plotted on the chart. One chart can have multiple plots on it.
<plotname>	It specifies the name of the plot that is to be drawn.
<apicall>	Specifies the API call to be made to retrieve the required data for the plot.

3.4 Charting APIs and Libraries

Effective visualization of the data happens through visual aids such as charts and graphs, which help in analyzing and generating insights. The process of rendering charts is supported by multiple charting libraries and third-party APIs. The data to be visualized and the type of chart to be rendered are passed to the charting library or third party APIs, which render the chart on the dashboard portal.

```
<screen>
 <enable>ON</enable>
 <name> All Actions </name>
 <description> User Actions in Total</description>
 <row>
  <height> 20 </height>
  <chart>
    <width> 80 </width>
    <chartsource> Google Charts </chartsource>
    <charttype> Line </charttype>
    <plot>
        <plotname> Buys </plotname>
        <apicall> http://api.in10do.com/place_order/?key=3a18328b8 </apicall>
    </plot>
    <plot>
        <name> Add To Cart </name>
        <apicall> http://api.in10do.com/add_to_cart/?key=3a18328b8 </apicall>
    </plot>
  </chart>
 </row>
</screen>
```

Figure 4. Snippet from the configuration XML File

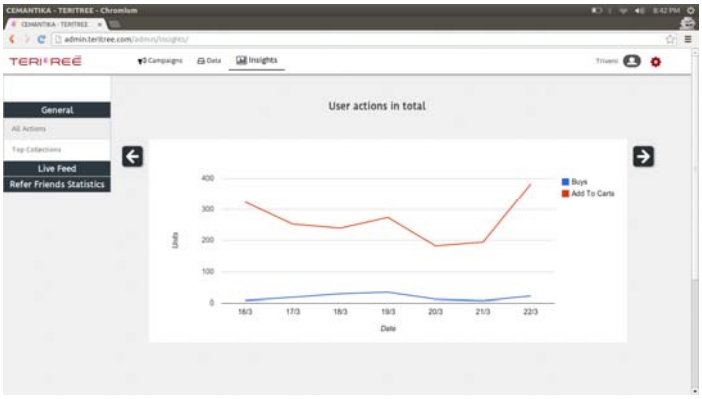

Figure 5. Dashboard created using the XML file shown in Figure 4

In our framework, there is an abundance of charting options as multiple charting libraries and APIs are supported. Each of the libraries or APIs may include a variety of chart types. The user may choose to render any type of chart from any of the supported libraries or APIs.

Following is the list of popular charting libraries and APIs that are supported in this framework:

- Google Charts [6]
- FusionCharts [5]
- Highcharts JS [7]
- Flot [4]
- jqPlot [9]
- jQuery Sparklines [10]
- Chart.js [2]

This is a scalable system, where new charting libraries or APIs can be supported with minimal coding effort as and when the need arises.

3.5 The Core of the Framework

APIs for retrieving and visualizing data form the core of the framework and they are explained in the following sub-sections.

3.5.1 Data-Retrieving API

The data-retrieving API of the framework is responsible for fetching the required data to be visualized, using the Wrapper API calls specified in the configuration XML file. When the user clicks a button to load a particular graph or chart from the dashboard, the framework reads the

DOI: 10.1145/2632434.2632444 http://doi.acm.org/10.1145/2632434.2632444

configuration XML file and gets all the required information. This includes information regarding the chart type, charting libraries or APIs and the Wrapper API calls to be made to retrieve the data of interest. The Wrapper API must serve data in one of the formats supported by the framework. Once the required data is fetched, it is passed on to the Data-Visualization API of the framework, which renders the required chart using the specified charting library or API.

3.5.2 Data-Visualization API

The Data-Visualization API of the framework is responsible for visualizing the data fetched by the Data-Retrieving API. It gets information about the type of chart to be rendered and its source library or API from the configuration XML file. Each charting library or API has its own pre-defined format for data-input. The Data-Visualization API converts the fetched data into the required data-input format and calls the appropriate charting library or charting API functions to render the chart or graph on the dashboard portal.

The UML sequence diagram for rendering a chart using this framework is shown in Figure 6.

4. RESULTS AND DISCUSSION

This framework has been successfully implemented and is currently being used in the CEMANTIKA platform [1] of Teritree Technologies Pvt Ltd [15]. Dashboards for multiple vendors are being supported on this website. The feedback received from these vendors is quite encouraging. The cycle time for creating a dashboard for a new user has been drastically reduced from a few days to a few hours.

Before this framework was deployed, it used to take four days of development effort to create a dashboard to suit a new customer's requirements at Teritree Technologies Pvt Ltd. The process of creating dashboards for new users has been considerably simplified by this framework and a data analytics dashboard can be created within the short span of one hour.

This framework currently supports the dashboards of four e-commerce vendors, and this involved extensive use of the customization features. Additional charting libraries or APIs and more data-input formats can be supported in this framework with minimal coding effort.

5. CONCLUSION

In this paper, a framework for data-agnostic dashboard visualization with customizable layouts has been discussed. This framework has been successfully developed and deployed at Teritree Technologies Pvt Ltd [15]. The dashboards of multiple e-commerce vendors have been successfully created using this framework and the feedback from these users has been very good.

The possible future enhancements to this framework would include extending the support to more charting libraries and APIs, supporting more data-input formats and implementing a drag-and-drop interface to improve the user experience in customizing layouts.

6. ACKNOWLEDGEMENTS

We would like to thank Mr. Ravikiran Annaswamy, Mrs. Tina Mani and Mr. Milind Utsav of Teritree Technologies Pvt Ltd for their support in the conceptualization and development of this framework.

7. REFERENCES

[1] (2014) CEMANTIKA [Online].

Available: http://www.admin.teritree.com/

[2] (2014) Chart.js [Online]. Available: http://www.chartjs.org/

[3] Dattatray V. Bhate and M. Yaseen Pasha . 2014. Analyzing Target Consumer Behavior Using Data Mining Techniques for E-Commerce Data. *International Journal of Innovative Research in Computer Science and Technology*. 2,1 (Jan 2014).

[4] (2014) Flot. [Online]. Available: http://www.flotcharts.org/

[5] (2014) FusionCharts. [Online].

Available: http://www.fusioncharts.com/explore/charts/

[6] (2014) Google Charts. [Online].

Available: https://developers.google.com/chart/

[7] (2014) HighCharts JS [Online].

Available: http://www.highcharts.com/products/highcharts

[8] Jaideep Srivastava, Robert Cooley, Mukund Deshpande and Pang-Ning Tan. 2000. Web usage mining: discovery and applications of usage patterns from Web data. *ACM SIGKDD Explorations Newsletter*, 1,2 (Jan 2000). DOI=http://doi.acm.org/10.1145/846183.846188

[9] (2014) jqPlot. [Online]. Available: http://www.jqplot.com/

[10] (2014) jQuery Sparklines. [Online].

Available: http://omnipotent.net/jquery.sparkline/#s-about

[11] (2014) Logic Monitor. [Online].

Available: http://www.logicmonitor.com/quick-tour/easily-customizable-dashboards/

[12] (2014) RazorFlow. [Online]. Available: http://razorflow.com/

[13] (2014) Sonarsource. [Online]. Available: http://www.sonarsource.com/products/features/customizable-dashboards/

[14] (2014) Splunk Enterprise Framework. [Online]. Available: http://docs.splunk.com/Documentation/Splunk/6.0.1/Viz/Aboutthismanual

[15] (2014) Teritree Technologies Pvt Ltd [Online].

Available: http://www.teritree.com/

[16] Yuantao Jiang and Siqin Yu. 2008. Mining E-Commerce Data to Analyze the Target Customer Behavior, *First International Workshop on Knowledge Discovery and Data Mining. WKDD*. 406-409. (Jan 2008). DOI = http://doi.acm.org/10.1109/WKDD.2008.90

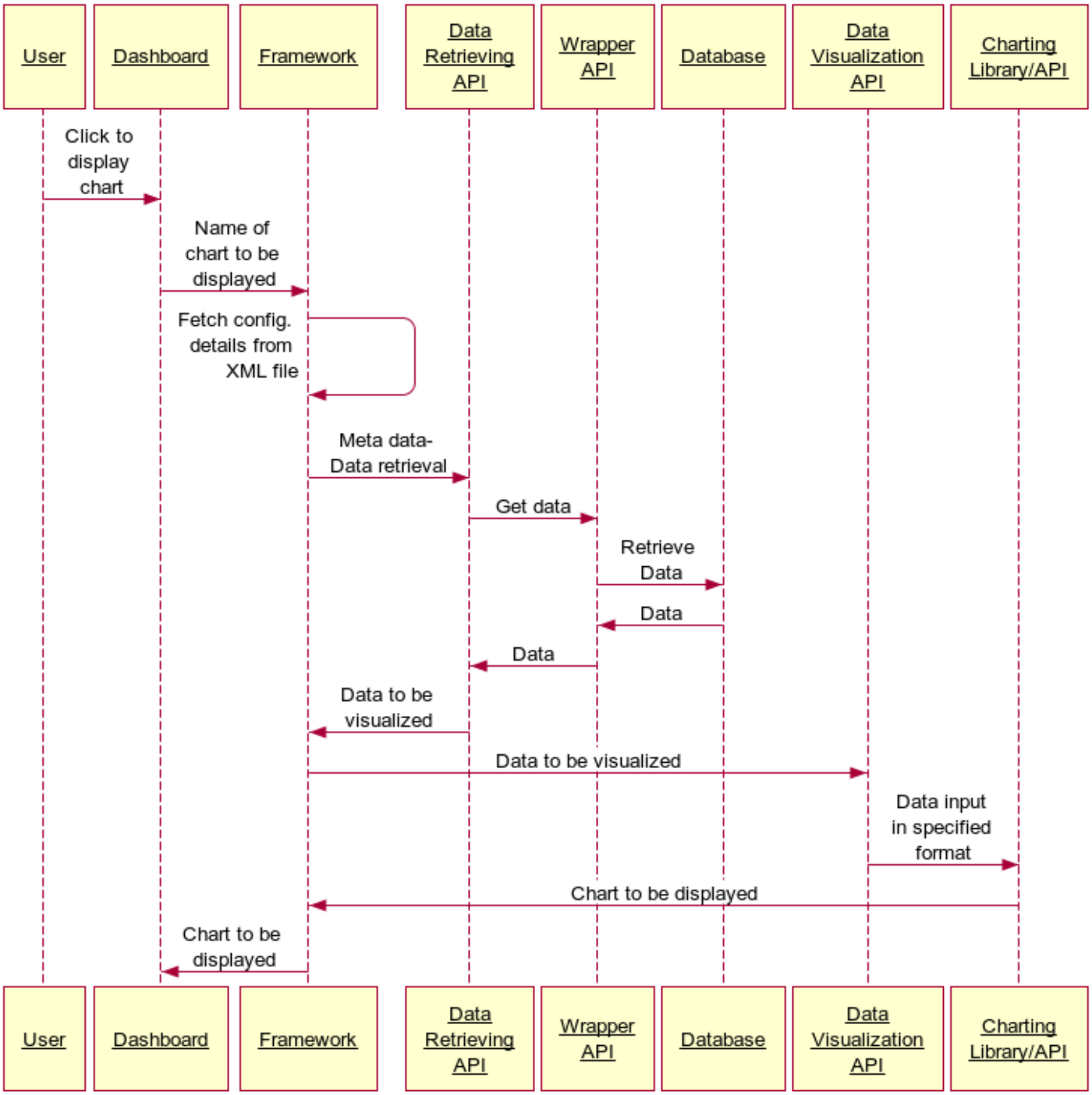

Figure 6. UML Sequence diagram showing the interactions involved while loading a chart

DOI: 10.1145/2632434.2632444 http://doi.acm.org/10.1145/2632434.2632444