Hung Ho Ngoc

Single Page Web Application with Restful API and AngularJS

Best Practices with Verto Monitor

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Degree Programme in Media Engineering

Bachelor's Thesis

1 November 2014

Helsinki Metropolia University of Applied Sciences

Author(s) Title Number of Pages Date	Hung Ho Ngoc Single page web appplication with Restful API and AngularJS: best practices with Verto Monitor 48 pages + 5 appendices 1 November 2014
Degree	Bachelor of Engineering
Degree Programme	Media Engineering
Specialisation option	JAVA and .NET Application Development
Instructor(s)	Tero Parviainen, Project Manager Petri Vesikivi, Principal Lecturer

Nowadays, research services cover only patches of the fast growing digital market, and are, at many times, still heavily based on survey-based research or temporal measurement activities that have major shortcomings in all things digital. It is increasingly important to access hard data that provides analytics of the state of today's digital ecosystem across hardware and software.

Project Verto Monitor is targeted towards customers or companies which want to measure how many smart devices there are in the market place, what is being downloaded on those devices, how much money is spent in the ecosystem by different parties, and finally, how the devices, services and content are being used by the customers.

Recognizing that a vital aspect of measuring Internet devices and usage is to provide customers or companies a comprehensive measurement service that captures all devices usage throughout the day, this project results a measurement framework to achieve main objectives and allows: tracking of consumer engagement with different smart devices, measuring a long tail of properties and services and tracking both sales and the number of smart devices being used.

Receiving two secure fundings after one year is quite success. The ultimate goal of this project is providing internet media measurement services around smart digital devices. The final version of Verto Monitor can establish a strong track record in the digital measurement field and help customers to do tactical and strategic decisions, providing insights though hard measurement of data visualizations in the multi-screen digital industry.

Keywords	SPA, AngularJS, Javascript, web development, web application, HighchartJS

Contents

1 Introduction 6
2 Single Page Web Application 7
    2.1 Structure of SPA 7
    2.2 Model View Controller 8
3 SPA with AngularJS 9
    3.1 Introduction of AngularJS 9
    3.2 Architecture 9
    3.3 Basics of AngularJS 11
        3.3.1 Data Binding 11
        3.3.2 Module 11
        3.3.3 Scope 12
        3.3.4 Controllers 13
        3.3.5 Services 15
        3.3.6 Directives 17
        3.3.7 Events 18
    3.4 Communicating with server 20
        3.4.1 REST API 20
        3.4.2 $http Service 22
4 Case Project Verto Dashboard 23
    4.1 About Verto Analytics 23
    4.2 Digital Measurement Challenge 24
    4.3 Data Deliverables 25
        4.3.1 Conceptual overview 25
        4.3.2 Verto Monitor 26
        4.3.3 Data API 27
5 Apply SPA to Verto Dashboard 28
    5.1 Key Design and Development 28
    5.2 Dashboard Architecture 28
    5.3 Basic Layout and Primary Navigation 30
        5.3.1 Basic Layout 30
        5.3.2 Primary Navigation 31
    5.4 Configuration Component 32
        5.4.1 Time Period 32
        5.4.2 Day Filter 33
        5.4.3 Daypart Filter 34
        5.4.4 Resolution 34
        5.4.5 Target Group Filter 35
        5.4.6 Subject Selector 35
        5.4.7 Metrics 36
    5.5 Display Component 37
        5.5.1 Data Chart 37
        5.5.2 Data Table 38

5.6 Report 39
5.6.1 Ranking Report 39
5.6.2 Share Report 40
5.6.3 Usage Trend Report 42
6 Conclusion 45
References 46
Appendix 48

Abbreviations and Terms

SPA	Single Page Application
MVC	Model-View-Controller
REST	Representational State Transfer
CRUD	Create - Read - Update - Delete
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheets
JS	Javascript Language
JSON	Javascript Object Notation
API	Application Programming Interface
DOM	Domain Object Model
R&D	Research and Development
PNG	Portable Network Graphics
JPG	Photographic Experts Group
BMP	Bitmap Image

1 Introduction

The digital landscape is changing every day. New platforms, new devices, new content, new products, and new ways of monetizing their intersections mean that the industry must track new metrics, across new ecosystems by using new methods.

In order to compete in today's converged and accelerating environment, comprehensive measurement across devices, platforms, and ecosystems is the only way to understand the evolving marketplace, assess performance, identify market opportunities, and make informed decisions. Passive measurement of consumer behaviors, emerging trends, and daily usage patterns, is the key in producing quantified data and information for decision making, also it provides superior means to do consumer research on these topics, versus traditional survey or interview based research.

Verto Analytics is operating in a multi-screen digital media measurement service and providing business critical information for strategic (market insights, competitive analysis, consumer behavioural and purchasing trends) and tactical (media buy and sales, financial investments and product development) decision making to a global clients, being based in New York (USA) and Helsinki (Finland).

Verto reports on consumer behavior with digital content, performance of properties and publishers, platform and device diffusion, and digital device usage, across a multi-screen digital world, including smartphones, tablets, desktop and laptop computers, smart TVs etc.

Verto Monitor is described as a good example of a single page web application in this thesis. The main objective of company is a combination of background technical knowledge and data processing both in theory and in practice. First three chapters explain about how to work with popular open-source framework AngularJS; the following chapters presents how to apply to an example enterprise product. The first successful steps are also presented in the conclusion part of the thesis.

2 Single Page Web Application

2.1 Structure of SPA

A Single Page Application (SPA) is a web application fitted on a single page and does not reload the page during use to provide better user experience and performance.

When the web application moves to the browsers, the requirements for performance in server are quite important. The following figure illustrates how SPAs solve a performance problem with the business logic and HTML templates migrate from the server and the client.

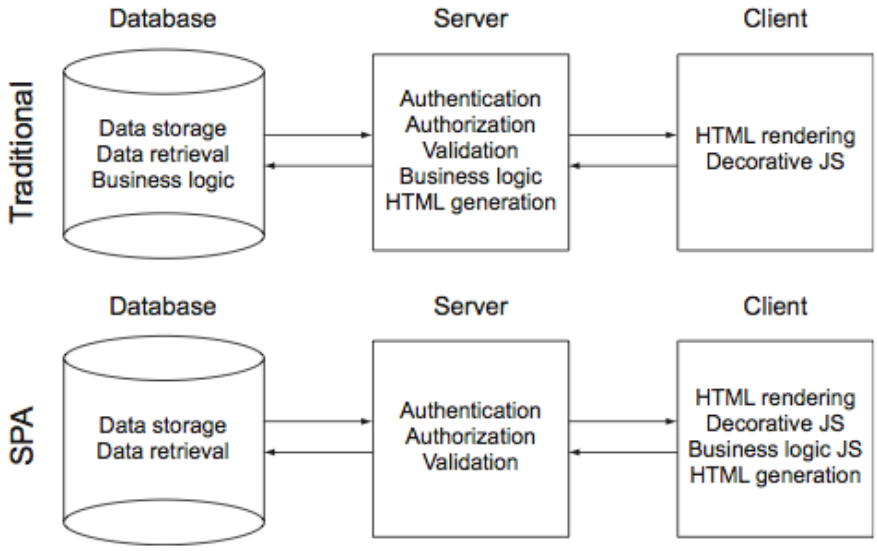


Figure 1: Responsibilities of database, server and client between traditional application and single page application. [1, 8]

As can see from figure, with SPA all business logics from database and server (traditional) are now moved to client side (SPA) in order to improve performance of the database and server. In comparison to a traditional method, each time a request goes to server, it receives the request, renders the response and sends it back to requester. This cycle would repeat for every request and take a lot of time for loading. [1, 9]

2.2 Model View Controller

The Model-View-Controller (MVC) is a popular design pattern used in developing web applications. In simple terms, it separates the user interface of the application from the underlying application business logic. The three parts of the pattern are illustrated as follows:

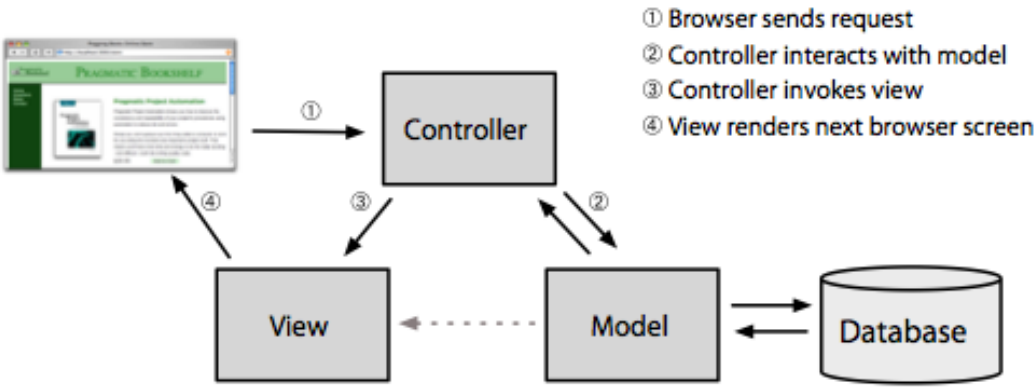


Figure 2: The Model-View-Controller architecture [2]

The Model manages the state of application. It can respond to data request or even notify the observer in application events when information changes. The model is just object data or some structure of objects and it enforces all the business rules to apply to that object data.

The View provides a use interface of application and know how to respond to user actions, normally based on data in the model.

Controllers act as an intermediary of the application. They receive events from the user's interactions (normally user input), make calls to the model and display an appropriate view to the user.

3 SPA with AngularJS

3.1 Introduction of AngularJS

AngularJS is a well-known open source JavaScript MV* (Model – View – Controller or Model - View - ViewModel) framework developed and maintained by Google. It is the next generation framework designed to give JavaScript developers a highly structured approach to developing cutting-edge web application.

Below are some reasons for AngularJS's considerable growth [3.]
Extendable: AngularJS simplify a complex AngularJS app works by splitting. application into MVC component and easily enhance applications with customised module.
Maintainable: AngularJS is supported by active open-source community
Testable: AngularJS supports unit and end-to-end testing that beats the traditional way of testing web apps by creating individual test pages.
Standardized: AngularJS helps to create standard web applications that use the latest features (such as HTML5 APIs) and popular tools and frameworks.

3.2 Architecture

AngularJS exists in the browser, which leads to a twist on the MVC pattern, as illustrated in the following figure

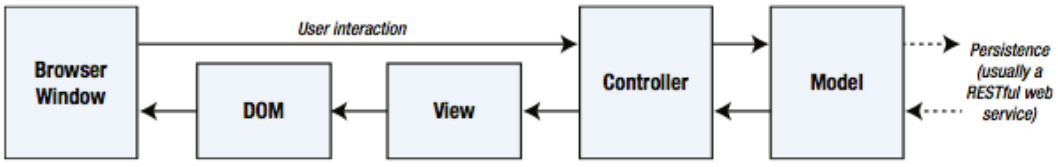


Figure 3: The flows of client-side MVC pattern [4. 48]

As can be seen from above figure, the MVC client-side implementation gets data from server-side API, usually via a RESTful web service. The goal of the controller and the view is to process data in the model in order to perform DOM manipulation so as to create and manage HTML elements that the user can interact with. [4, 51] Those

interactions are brought back to the controller, closing the loop to form an interactive application.

Although AngularJS web applications use the MVC pattern, the underlying components rely on a wider range of building blocks. There are the headline components including the model, the views and the controllers. However, there are lots of other important parts in an AngularJS app as well, comprising of modules, directives, filters, factories, and services.

Those different types of AngularJS component are tightly integrated and illustrated in following summary table:

AngularJS component	**Description**
angular.module method	Method to create an AngularJS module
ng-app attribute	Set the scope of a module
Module.controller method	Method to define a controller
ng-controller attribute	Apply a controller to a view
$scope service	Pass data from a controller to a view
Module.directive method	Method to define a directive
Module.filter method	Method to define a filter
$filter service	Use a filter programmatically
Module.service / Module.factory / Module.provider method	Method to define a service
Module.value	Define a service from an existing object or value
Module.config and Module.run methods	Register functions that are called when modules are loaded

Table 3. Different components in AngularJS application

3.3 Basics of AngularJS

3.3.1 Data Binding

AngularJS takes a different approach for combining data from models and deliver to view. Instead of using traditional way to merge data into a template and then replace a DOM element, AngularJS creates a view by using live HTML templates. Each component of the views is interpolated dynamically. This feature is one of the most important features in AngularJS and can be seen clearly in figure 4. [5.]

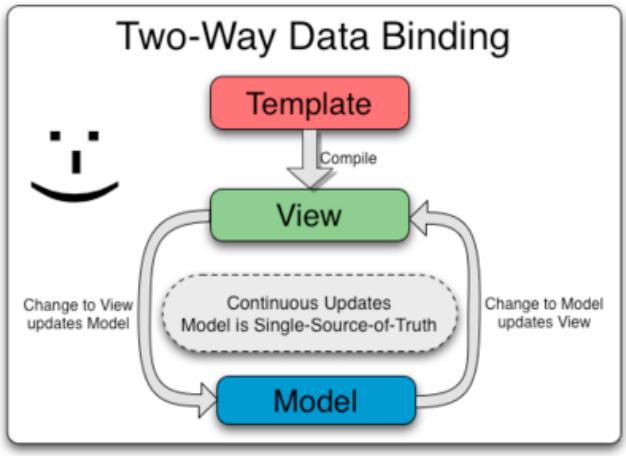


Figure 4: AngularJS two-way data binding model [6.]

As seen from figure 4, because AngularJS supports bi-directional or two-way data binding, there are two approaches happening at the same time: when the view changes or modifies the value data, the model observes changes by using its dirty checkin and when the model changes the values or states, the view also update with the change.

3.3.2 Module

In AngularJS, a module is the first step to define an AngularJS application. The app's module contains all application logic code. An AngularJS application can have one or many modules, each of them can manage specific functionality.

In addition, module also gives a lot of advantages which are keeping our global namespace clear, making tests easier to write and keep them clean, making it easy to

share code between application and allowing our app to load different parts of code in any order [7.]

AngularJS uses method angular.module() to declare a module. There are two parameters in this method. The first one is the name of the module and the second one is the list of dependencies. For example: *angular.module('myApp', [])*

Function angular.module returns an instance of a newly created module. By providing a value to the ng-app attribute to view, AngularJS application is activated as follows: *<body ng-app="myApp">*

3.3.3 Scope

Scopes are the core element of Angular application. The application model refers to the scopes. In addition, the $scope object is used to express the business logic of the application, the methods in controllers, and the properties in views. [8.]

Scopes serve as the middle layer between application controller and view. The view template connects to the scope before the application renders the view to the user. In addition, the application creates the DOM to notify Angular for changes in properties. Figure 5 is an example of using $scope as a glue between controller and view:

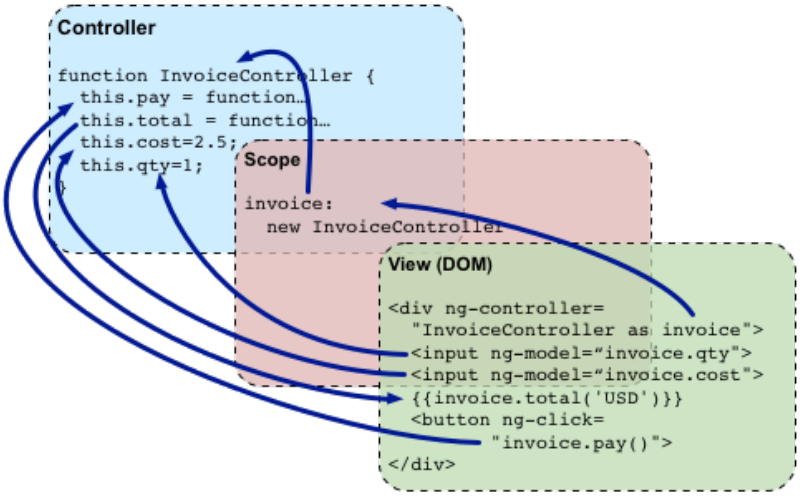


Figure 5: How scopes work as middle layer [9.]

Scopes are the primary elements for the application state. Because of this live binding, $scope can update value immediately when the view has been changed. In addition, the view can be updated when the $scope changes. It is a so-called two-way data binding.

In addition, scopes have some basic functions that are providing observers to watch for changes in model, creating the way to propagate model changes inside and outside applications to other components, being nested in order to split functionality and model properties easily and providing an execution environment in which expressions are evaluated.

3.3.4 Controllers

In AngularJS, controllers exist to the view of an AngularJS application. The controller is a function that creates business logic functionality to the scope of the view. When creating a new controller on a page, AngularJS initializes a new $scope. This new $scope is the state of the scope on controller. Listing 1 is an example of using controller in application. [10.]

Module

```
var app = angular.module('app', []);
app.controller('FirstController', function($scope) {
  $scope.message = "hello";
});
```

Controllers

```
app.controller('FirstController', function($scope) {
  $scope.counter = 0;
  $scope.add = function(amount) { $scope.counter += amount; };
  $scope.subtract = function(amount) { $scope.counter -= amount; };
});
```

View

```
<div ng-controller="FirstController">
  <h4>The simplest adding machine ever</h4>
  <button ng-click="add(1)" class="button">Add</button>
  <a ng-click="subtract(1)" class="button alert">Subtract</a>
  <h4>Current count: {{ counter }}</h4>
</div>
```

Listing 1. Example code of using controller in simple AngularJS application

From the code of listing 1, when the button is pressed, both the button and the link are bound to an action on the containing $scope. Add or subtract functions that are defined on the “FirstController” scope or parent $scope will be called and then they return counter result.

Controller Hierarchy (Scopes within Scopes)

AngularJS application has a default $rootScope and many parent scopes when the context is rendered. By default, if AngularJS cannot find current properties on a local scope, it will go to upper level in the parent scope and look for the properties or methods there.

In case AngularJS cannot find the properties in current scope, it will walk to that parent scope and upper levels until it reaches the $rootScope. If it does not find it on the $rootScope, it will stop updating the view.

The following listing 2 creates a ParentController that contains the user object and a ChildController that wants a reference of that object.

```
app.controller('ParentController', function($scope) {
  $scope.person = {greeted: false};
});

app.controller('ChildController', function($scope) {
  $scope.sayHello = function() {
    $scope.person.name = "Ari Lerner";
    $scope.person.greeted = true;
  }
});

<div ng-controller="ParentController">
  <div ng-controller="ChildController">
    <a ng-click="sayHello()">Say hello</a>
  </div>
  {{ person }}
</div>
```

Listing 2. An example of using controller hierarchy in AngularJS application.

Because of the prototypal behavior, data can be referenced from the code on the ParentController in which contains $scope on the child scope. Once link “Say Hello” is clicked, $scope.person value in the ChildController is referenced as a person object defined in the ChildController’s $scope object.

3.3.5 Services

Services are functions or singleton objects instantiated when an application component depends on it by using the $injector and lazy-loaded when needed. Services also provide an interface to keep methods related to a specific function or task. [11.]

As default, AngularJS comes with several built-in services. It is also useful to make customized services for any complex applications. In addition, AngularJS helps to create customized service in a easy way.

The following illustration shows an example of how to register and define a service:

```
angular.module('myApp.services', [])
  .factory('githubService', function($http) {
      // Our serviceInstance now has access to
      // the $http service in it's function definition
      var serviceInstance = {};
      return serviceInstance;
  });
```

Listing 3: How to define a simple service.

To use a service, one only needs to embed a dependency for the component where using it. It can be a controller, a directive, a filter or another service. At run time, Angular will carry on instantiating it and resolving dependencies shown in the following listing.

```
angular.module('myApp', ['myApp.services'])
.controller('ServiceController',
    function($scope, githubService) {
      // We can call the events function
      // on the object
      $scope.events =
        githubService.events('auser');
});
```

Listing 4: How to inject service into controller.

There are five different methods for creating services including: [12]
factory(): the factory() method is a popular way to create and configure service. It takes two arguments: the name of service and function runs when Angular creates the service.

service(): In order to register an instance of a service using a constructor function, using service() enables the app to register a constructor function for service object. It also takes two arguments: the name of service and constructor function that needs to be called to instantiate the instance. The service() function will instantiate the instance using the new keyword when creating the instance

constant(): In order to register an existing value as a service that can inject into other parts of application as a service. The constant() function takes two arguments: the name of service and the value to register as the constant.

value(): If the return value of the $get method in service is a constant, defining a full service with more complex methods is unnecessary. The major difference between the value() method and the constant() method is that constant() method injects a constant into a config function where as values() cannot do the same. The value() method accepts two arguments: name of registered service and value as injectable instance.

3.3.6 Directives

Directives are Angular's method on DOM elements which attach their own custom functionality or special behaviour to it. For instance, the following video tag is customized and works across all browsers:

```
<my-better-video my-href="/goofy-video.mp4">
Can even take text</my-better-video>
```

Listing 5. An example of using directive in AngularJS application.

As seen from listing 5, this custom element has custom open and closing tags, my-better-video, and a custom attribute my-href.

Built-in directives

AngularJS comes with serveral built-in directives. Some directives override built-in HTML elements, such as the <form> and <a> tags. In addition, there are some basic ng attribute directives. For example ng-href, ng-src, ng-disabled, ng-checked, ng-readonly, ng-selected, ng-class, ng-style and more.

Custom directives

A custom directive is simply a function that runs on a particular DOM element to provide supplemental functionality. It is defined by using the .directive() method, one of the many methods available on application's Angular module seen in listing 6.

```
angular.application('myApp', [])
.directive('myDirective', function() {
  // A directive definition object
  return {
    // directive definition is defined via options
    // which we'll override here
  };
});
```

Listing 6. A simple definition of custom directive.

As seen from listing 6, the directive() method takes two arguments: name (string) and a function returns an object that defines how the directive operates. In order to use in template, just put it as an attribute or element with snake-case name such as listing 7:

```
<div my-directive></div>
```

Listing 7. How to use that custom directive in practice.

3.3.7 Events

Events are information propagated across an application that generally contain information about what is happening inside of that application. [13.]

Event Propagation

Because scopes are in hierarchical structure, events can be transferred up or down the scope chain. There are two types of notification: for an entire event system using broadcast downwards and for a global module with passing an event upwards.

Bubbling an event up with $emit

$emit() function is used to dispatch an event to the scope chain from child scopes to parent scopes. $emit() is used to communicate the changes of state from within the application to the rest of the application as seen in listing 8:

```
// Send an event that our user logged in
// with the current user
scope.$emit('user:logged_in', scope.user);
```

Listing 8. An example of using emit to dispatch event.

In listing 8, when scope.$emit() event function call, the event travels up from the child scope to the parent scope. All the parent scopes of the current scope will listen to the event's notification. $emit() method takes two arguments: name (string) and args (a list of arguments are passed into the event listeners played as objects).

Sending an event down with $broadcast

$broadcast() is used to pass an event downwards from parent scopes to child scopes. $broadcast() method takes two parameters: name (string) and a list of arguments played as objects passed into the event listeners. Listing 9 is a good example to illustrate this:

```
// hold on, cart is checking out
// so all directives below should disable
// themselves while the cart is checking out
scope.$broadcast('cart:checking_out', scope.cart);
```

Listing 9. An example of using broadcast to send event.

By using $broadcast() method, all child scopes that registers a listener will receive this message. Then, this event is notified to all directives, current scope and calls every listener all the way down.

Events Listening

$on() method is used to listen for an event and call a listener for the event used with a particular name. The event name is just the event type fired in Angular same as listing 10:

```
scope.$on('$routeChangeStart',
  function(evt, next, current) {
    // A new route has been triggered
});
```

Listing 10. An example of using on method to listen for event

3.4 Communicating with server

3.4.1 Representational State Transfer (REST) API

Representational State Transfer (REST) is the stateless architecture based on HTTP protocol to represent the model of how the modern Web should work. REST uses existing technology and protocols of web to access and manipulate resources using verbs (HTTP request methods). [14.]

HTTP Verb	Action to take on resource	Database operation
POST	Create	Insert
GET	Retrieve	Select
PUT	Update	Update
DELETE	Delete	Delete

Table 1. How HTTP actions are relevant to database operation

There is a set of verbs for RESTful approach such as GET, POST, PUT and DELETE to manipulate data in database. Those verbs correspond to traditional CRUD (Create, Read, Update, Delete) operations in a database or object-relational management (ORM) system. In addition, REST also has 5 categories of responses: general information, a successful request, redirect, error on the client side and error on the server side. The response depends on the HTTP request type, as indicated in following table:

First Digit	Meaning
1xx	Information
2xx	Success
3xx	Redirection
4xx	Client Error
5xx	Server Error

Table 2. REST responses' common errors [15.]

GET: GET request is used by web client in order to ask for a representation of a resource, identified by a URL. For example, the client asks for a list of metric branches, the server then returns it in application as JSON format as shown in the following picture:

Figure 6: Example of success GET request and response from server

POST: The web client send a POST to request for creating a new resource based on a given representation. The most common response codes to a POST request are 200 (OK) or 201 (Created) and 202 (Accepted). The first two ones let the client know that a new resource was created and the latter means that the server intends to create a new resource based on the given representation, but has not actually created it yet. This is demonstrated in figure 7:

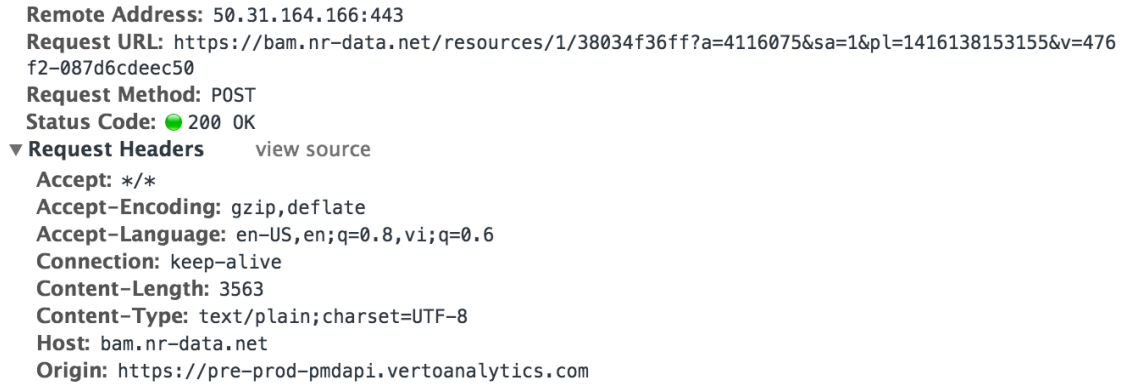


Figure 7: Example of success POST request and response from server

PUT: A PUT request is a request to modify resource data or information. If the server decides to accept a PUT request and then modified data in the server, it will change the

resource state to match what the client says in data representation, and often return 200 (OK) or 204 (No Content) response code.

Figure 8: Example of success PUT request and response from server.

DELETE: When client wants a resource to go away, it sends a DELETE request. Server will destroy resource and never refer to it again. If a DELETE request succeeds, the possible status code are 204 (No content) or 200 (OK) or 202 (Accepted).

Figure 9: Example of success DELETE request and response from server

3.4.2 $http service

The $http service is a more Angular service wrapping around the browser's XMLHttpRequest object to communicate with remote servers. $http service function takes only a single argument: a configuration object used to create a HTTP request. The function returns a promise that has two helper methods: success and error. Listing 11 is a simple code to show how to use $http service:

```
var promise = $http({
  method: 'GET',
  url: '/api/users.json'
});
```

Listing 11. An example of using $http service.

Since $http method returns a promise object, then() method or success() and error() can be used to handle the callback when the response is ready. If the response status code has a number value which ranges from 200 and 299, the response is considered successful, and the success callback will be called. Otherwise, the error callback will be invoked.

```
promise.then(function(resp) {
  // resp is a response object
}, function(resp) {
    // resp with error
});
// OR we can use the success/error methods
promise.success(function(data, status, headers, config) {
  // Handle successful responses
});
// error handling
promise.error(function(data, status, headers, config) {
  // Handle non-successful responses
});
```

Listing 12: Two ways of handling the callback when the response is ready.

4 Case Project with Verto Dashboard

4.1 About Verto Analytics

Verto Analytics (Verto is Latin, and means interpretation and change) is a global leader in providing Internet media measurement services around smart digital devices. Verto's management team includes senior executives from the market research industry, and a strong track record in creating high quality media research services and products, which are based on the industry best practices. They acknowledge the ever increasing need to account for new shifts in consumer engagement with both traditional and new forms of digital media, thereby designing, building and maintaining measurement technologies.

Verto Analytics works with the clients and provides its online measurement services to a global clientele in technology, finance, media, and entertainment industries.

Verto Pulse digital measurement services track the quickly evolving trends in digital usage and Internet media consumption, and the consumers' use of multiple devices and potentially simultaneous and interrelated behaviors across devices. Services and content are increasingly delivered to these devices in a variety of ways, ranging from native apps to web sites, and widgets to multimedia streaming. Furthermore, measurement of the related revenues is vital, in addition to mere engagement and other media related metrics. The increasing demand for insights poses a need for a next-generation media measurement provider that cracks the online measurements of the next decade.

Verto's core team has been building and providing its services in digital media measurements over the past 12 years to a number of customers in Europe, Asia and the US, and is open for mutually beneficial projects and commercial relationships based on its core competencies. Verto Analytics is based in New York, USA, with research and development and operational centers in Europe.

4.2 Digital Measurement Challenge

Nowadays, mobile usage data is characterised by three significant and accelerating trends:

Device Fragmentation: The environment used to be simple: radios, telephones, televisions – each device had limited functionality, and a short list of brands in an easily-defined category. Today's world is very different: devices support multiple functions, consumers own and use many devices, often for overlapping purposes. Understanding any one – or even any two or three – categories or brands is no longer enough for a complete picture.

Function Convergence: In the past, devices were function-specific: One could watch video content on a television, listen to audio content on a radio, run software on a

desktop computer, and make phone calls with a telephone. Today, any one of these devices (and others) can perform any of these functions. Understanding how consumers use any one device type is no longer enough: Consumers' real experiences and content/ software-producers' audiences are shaped across multiple devices, device types, and networks.

Audience Fragmentation: The digital revolution has seen an explosion in content, software, and services. This leads the audience to become fragmented in the marketplace. Understanding who these competitors are, and collecting data from large enough samples to report about their usage, are essential to understanding today's diverse ecosystem.

4.3 Data Deliverables

4.3.1 Conceptual overview

The Verto's Data Deliverables makes data available though two different forms

Component	Explanation
Verto Monitor	The Verto Monitor is a web-based tool that allows clients to visually and dynamically interact with the data collected through the Verto Smart Measurement Platform. Clients can analyze data across all measured devices, panelists, web sites, apps, categories, platforms, etc. The Verto Monitor makes available 100+ different metrics, each analyzable by target group, and over time.
Data API	The Data API is a RESTful API which accepts requests for particular analyses, extracts relevant data from the Session-level DB, and calculates the metrics indicated in the original request. It represents a programmatic way to gain access to the 100+ metrics that the Verto Smart Measurement Platform can calculate.

Table 4. Two basic components of Verto Deliverables

4.3.2 Verto Monitor

The Verto Monitor and Data API supports the calculation of 100+ metrics calculated based on data collected through the Smart Meters. These metrics can be calculated at hourly, daily, weekly, or monthly resolutions and filtered based on Time period, and/or; Day(s) of the week, and/or; Hour(s) of the day, and/or; Particular web sites, and/or; Particular apps, and/or; Particular demographic characteristics.

The following list contains the metrics that are calculated / available to clients through either the Verto Monitor or Data API:

Basic Metrics	Engagement Metrics	
+ Reach/Penetration (%) + Users (#) + Devices (#) + Devices per User (#) + Time Spent (hh:mm:ss) + Time Spent per User (hh:mm:ss) + Time Spent per Device (hh:mm:ss)	+ Sessions (#) + Sessions per User (#) + Sessions per Device (#) + Avg Session Duration (hh:mm:ss) + Avg Session Interval (hh:mm:ss) + Multi-screen Sessions (#) + Multi-screen Usage Rate (#) + Multi-screen Sessions per user (#) + Devices per Multi-screen Session (#)	+ Interactions (#) + Interactions per User (#) + Interactions per Device (#) + Interactions per Session (#) + Interactions per Minute of Use (#) + Avg Interaction Duraction (hh:mm:ss) + Bounces (#) + Bounce Rate (#) + Focus-Outs (#) + Foucs-Outs per User (#) + Foucs-Outs per Minute of Use (#)
Device/Telecom Metrics		
+ Telephone Calls (#) + Inbound Telephone Calls (#) + Outbound Telephone Calls (#) + Missed Telephone Calls (#) + Avg Calls per User (#) + Avg Inbound Calls per User (#) + Avg Outbound Calls per User (#) + Avg Missed Calls per User (#) …..	+ Avg Inbound Call Time per Device (hh:mm:ss) + Avg Outbound Call Time per Device (hh:mm:ss) + Avg Call Duration (hh:mm:ss) + Avg Inbound Call Duration (hh:mm:ss) + Avg Outbound Call Duration (hh:mm:ss) + Total Data Transfer (kB) + Inbound Data Transfer (kB) + Outbound Data Transfer (kB)	+ Wifi Connectivity Uptime (%) + Wifi Connectivity Uptime per User (hh:mm:ss) + Wifi Connectivity Uptime per User (%) + Wifi Connectivity Uptime per Device (hh:mm:ss) + Wifi Connectivity Uptime per Device (%) + Total Time Roaring (hh:mm:ss)

Table 5. List of metrics used in Verto Monitor (Dashboard)

4.3.3 Data API

The Data API accepts either a singleton or a batch of JSON objects, each of which represents a data request being asked. One can think of a data request as the "question" being asked, and the resulting data response as the "answer" being given. Data requests can specifically request any of the metrics provided in the Verto Monitor.

Data Request: While a detailed set of documentation will be provided, the following represents a complete data request in JSON format

Data Response: The following snippet of code represents a complete represents a complete data response in JSON format

```
{ "data_response": {
   "id":"1",
   "days":"1,2,3,4,5,6,7",
   "dayparts":"1,2,3,4,5",
   "resolution":"monthly",
   "target_group_filters":[ ... ],
   "request_type":"active",
   "display":"CHART-AND-TABLE",
   "source":"report template name",
   "data_series":[
          {
                 "id":"1",
                 "title":"Angry Birds",
                 "benchmark":false,
                 "subject":{
                        "title":"Angry Birds",
                        "description":"Angry Birds game by Rovio.",
                        "tooltip":"Angry Birds",
                        "logo_url":"...",
                        "icon_url":"..."
                 },
                 "data_point_format":{
                        "decimal_separator":".",
                        "min_decimal_digits":0,
                        "max_decimal_digits":0,
                        "thousand_separator":",",
                        "prefix":null,
                        "suffix":null
                 },
                 "yaxis":{
                        "value_type":"nominal",
                        "title":"Users (#)",
                        "oom":6,
                        "lowest_value":123456789,
                        "highest_value":987654321,
                        "logarithmic":false,
                        "force_whole_numbers":true,
                        "decimal_separator":".",
                        "min_decimal_digits":"0",
                        "max_decimal_digits":"0",
                        "thousand_seaprator":",",
                        "prefix":null,
                        "suffix":null,
                        "min_value":0,
                        "max_value":null
                 },
                 "data_points":[
                        {
                               "x":"2013-01-01 00:00:00.0000",
                               "y":123456789,
                               "n":2345,
                               "preliminary":true
                        },
                        { ... },
                 ]
   ]
}
```

Listing 13. An example of data response used in Verto Dashboard

5 Apply SPA to Verto Dashboard

5.1 Key Requirements for Design and Development

The Verto Dashboard will be a single page web-based application using HTML5, CSS3 and Javascript. Aaccessing to the Verto Dashboard's functionality will be strictly limited to authenticated users. User authentication will occur based on email address and password.

Dashboard is also be designed with a responsive layout using CSS3 media queries to be functional across devices with varying screen resolutions. Despite its responsive layout, the Verto Dashboard should be optimised for a screen resolution of 1280x800 with graceful transformation down to 1024x768 and up to 1920x1080.

In order to address 96% of the potential user base, the Verto Dashboard should support Microsoft Internet Explorer 8+, Google Chrome 25+, Firefox 5+, Safari 5.1+, Safari for iPad, Android 4+, Opera 12.1+. In addition, data displayed in charts and data tables should be refreshed or populated using AJAX so as to not refresh the entire page

5.2 Dashboard Architecture

The Verto Dashboard must simultaneously provide users with a passive ("consumption") model for interacting with the data and a active ("exploration") model for interacting with the data.

As seen from figure 10, Verto Monitor creates a significant development challenge, in that user experience (the view) and system logic (controllers) for each of these two models are very different by nature. Rather than support two complex and contradictory models simultaneously, it makes more sense to create an abstraction layer which both models can leverage:

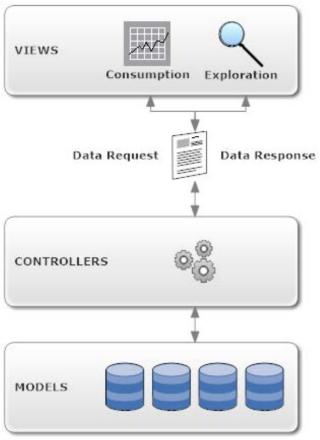


Figure 10: Basic Verto Monitor architecture

As the diagram above suggests, both the data consumption and data exploration process with create Data Requests and consume (display) Data Responses. From the controllers' perspective, Data Request will be identically structured. The only difference will exist in the view.

Data Requests produced by the "data consumption" approach will be embedded into the view as widgets. The user will be able to either turn their display on or turn their display off, but the configuration of these Data Request will not be possible.

Data Request produced by the "data exploration" approach will be dynamically created and configurable by the user. In other words, they may apply certain default values, but they are dynamically created and interpreted on the fly.

5.3 Basic Layout and Primary Navigation

5.3.1 Basic Layout

Once the user has been logged in, they should be presented in the standard layout of the Verto Dashboard. In general, this layout should have two main content areas:

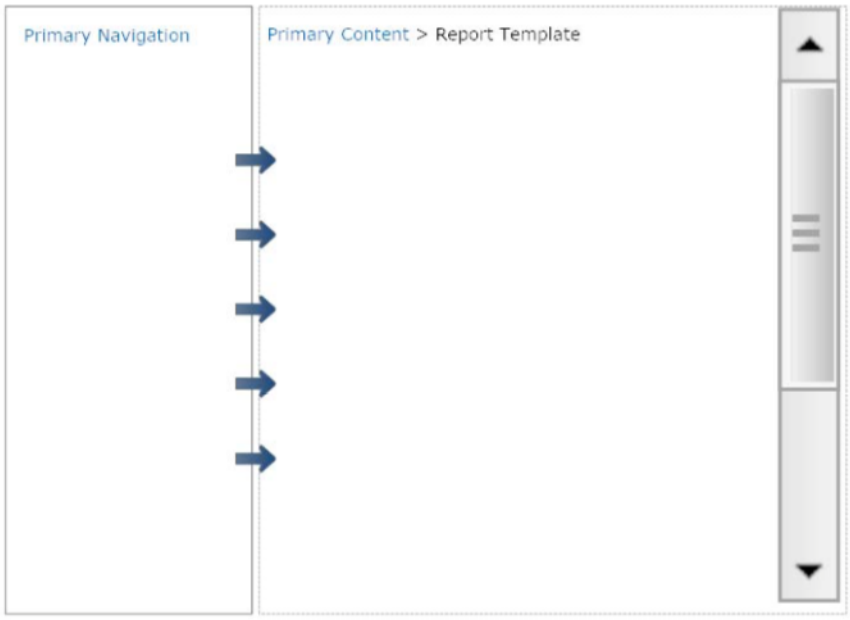


Figure 11: Basic 2-column layout of dashboard

The Primary Navigation area allows user to navigate throughout the Verto Dashboard. In particular, it should enable user to

- Select the Dashboard
- Browse available Report Templates
- Select the Report Template user wishes to work with
- Navigate to Account Management functionality

The Primary Content area displays functional content to the user, arranged into a two-column wide layout with visual elements either 1-column wide or 2-columns wide. The content displayed may be one of three types:

Dashboard (default upon login): If the Dashboard has been selected, then the user should be shown those Data Responses which have been pinned to his dashboard. These Data Responses cannot be configured by the user, but they should be contained in "widget-like" components that can be repositioned (clicked and dragged), minimised and "unpinned" (deleted from the Dashboard). Each widget also has "Detail" link which brings the user to a user-configurable version of the originating Report Template

Report Template: If a Report Template has either a) been selected in the Primary Navigation area, or b) clicked through from the Dashboard, then the content displayed in the Primary Content area should be that report. Each such Report Template should present the user with user-configurable Configuration Components and the resulting Display Components

Account Management: If one of the Account Management views has been selected in the Primary Navigation area, the content displayed should be that Account Management screen.

5.3.2 Primary Navigation

The primary Navigation area can be considered the "table of contents" to the Verto Dashboard. It should provide the user with a collapsible tree with textual links to access different Report Templates.

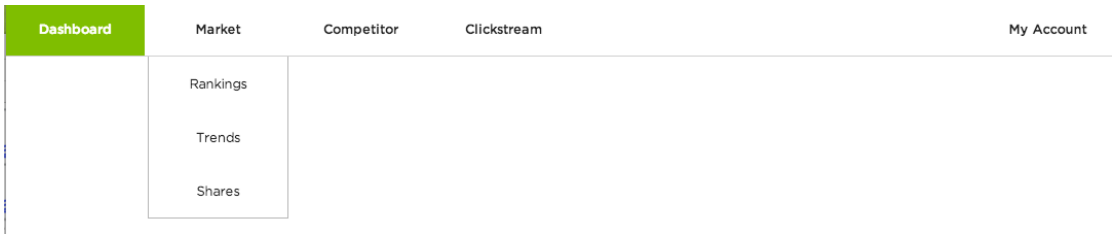


Figure 12: Primary Navigation of Dashboard

The table below provides a detailed list of the nodes that should be present in the Primary Navigation area, along with their relevant context-sensitive documentation and the report/functionality which they link to:

Node Label	Contextual Help
Dashboard	Quick view the most recent data about marketplace
Market	Explore how the industry is performing in the marketplace
+ Ranking	Sort competitors according to their performance
+ Trends	Explore trends in usage/behavior across competitors
+ Shares	Explore competitors market share
Competitor	Explore an individual competitor's performance
+ Overview	Wide-ranging analysis of an individual competitor's performance
+ Audience	Breakdown of a competitor's audience
+ Peer Comparison	Compare one competitor against a peer-group
Clickstream	
+ Usage Context	Explore what users do simultaneously to using this competitor
+ Usage Sequence	Explore how users interact with a competitor
My Account	Change user password and other account settings
+ Account Settings	Change user password and other preferences
+ Manage Users	Manage the users authorized to use account

Table 6. Detailed list of nodes used in primary navigation

5.4 Configuration Components

5.4.1 Time Period

The Time Period component is used to get the range of time for which data is reported to the user. It features two configurable positions: Start Date and End Date

Both of these positions should be expressed as dates formatted as MM dd, YYYY.

In order to converse screen real estate, the Time Period is generally displayed as textual information for the user. However, when the user clicks on either the Start Date or the End Date, a small calendar control should appear below the clicked position. This calendar control should allow the user to select the value for the position.

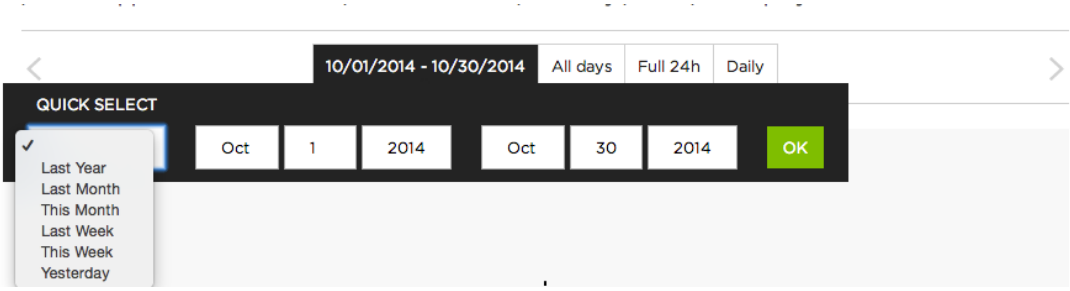


Figure 13: Layout of Time Period selection

Note that above the time period is a series of several "quick selection" links. When clicked, these links automatically adjust the Start Date and End Date positions based on the current date. These "quick selection" position should be as follows: Last Year, Last Month, This Month, Last Week, This Week and Yesterday.

5.4.2 Day Filter

The Day Filter component is used to select which days of the week should be included in the Report. This component should be a "toggle" control, meaning that when user clicks on a give day of the week that day should either be togged "ON" or "OFF" according to its state:

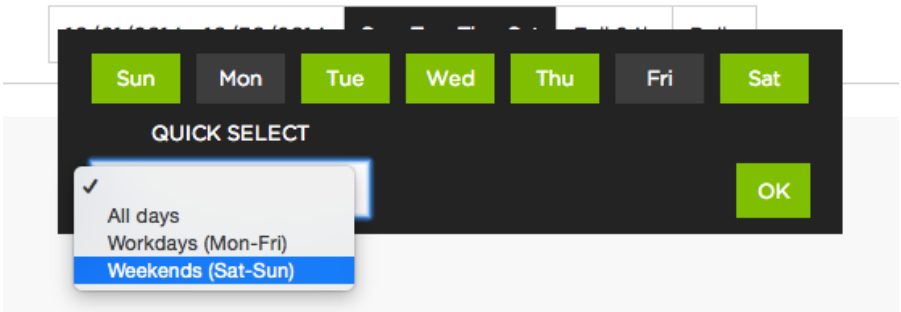


Figure 14: Layout of Day Filter

5.4.3 Daypart Filter

The Daypart Filter component is used to select which dayparts (range of hours) should be included in the Report. This component should combine both a “toggle” control (when the user clicks on a give hour, it should either be toggled “ON” or “OFF” according to its state)

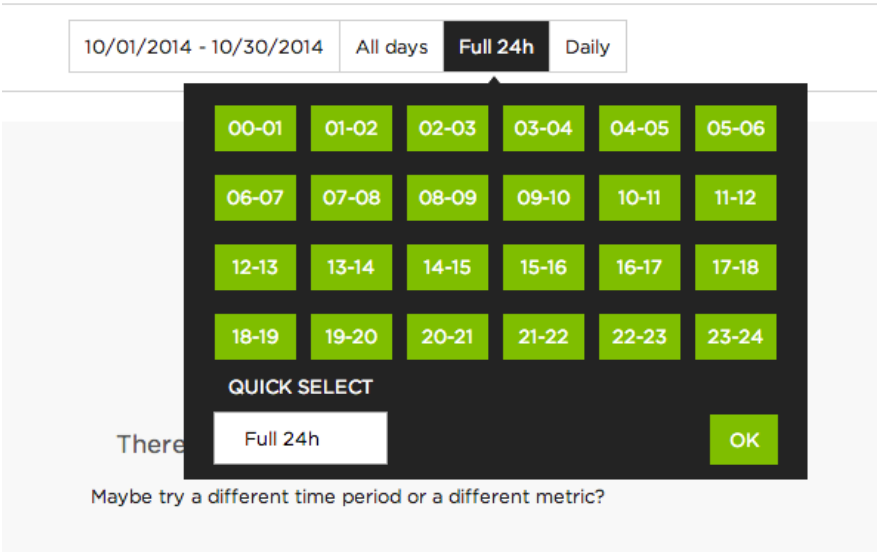


Figure 15: Daypart Filter component

5.4.4 Resolution

The Resolution component is used to determine at what level of granularity (over time) data should be displayed. While Resolution is a Configuration Component, it should more visually represent a navigation element, a “tab” displayed above the Display Components.

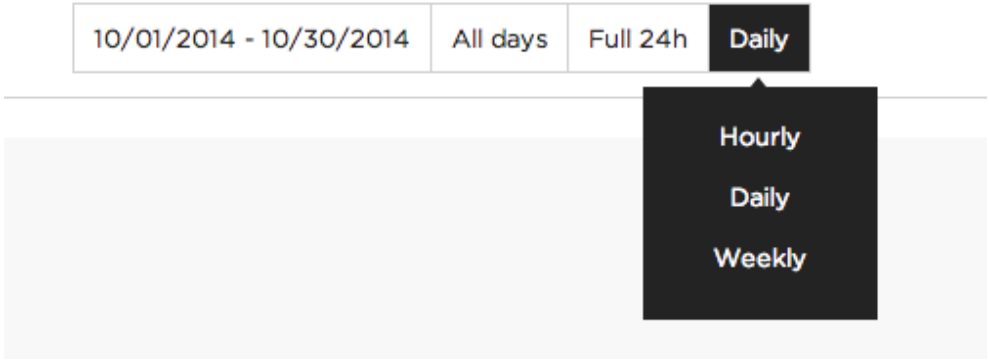


Figure 16: Resolution Component

The Resolution component can receive any one of the following values: Hours, Days, Weeks, Months, Quarters and Annuals.

5.4.5 Target Group Filter

The Target Group Filter component is used to select what demographic or behavioural criteria should be applied to the Report. When the user clicks on the target group name, a box containing a list of the user's defined target groups should be opened.

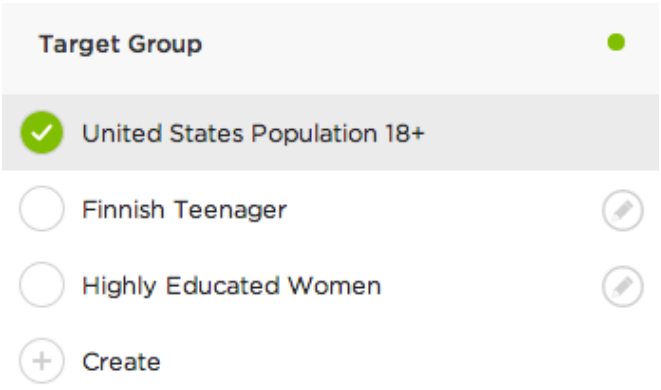


Figure 17: Target Group Filter

This Target Group Filter box contains "Create New Group" link which when clicked opens a modal dialog box containing the Edit Target Group view and "List of Target Groups" which are a list of the user's defined target groups, starting with the default "United States Population 18+" (no target group filter). With the exception of the first (default) target group, each group in this list have "Edit" link in which clicked opens a dialog box containing Edit Target Group view and allowing the user to edit the selected target group.

5.4.6 Subject Selector

The Subject Selector component is used to indicate which entities the user wishes to include in their analysis. The Subject Selector is a relatively complicated component, in that it receives its values from the Meta-data Platform, should be organised in a tree-like hierarchy and organized into several different hierarchies.

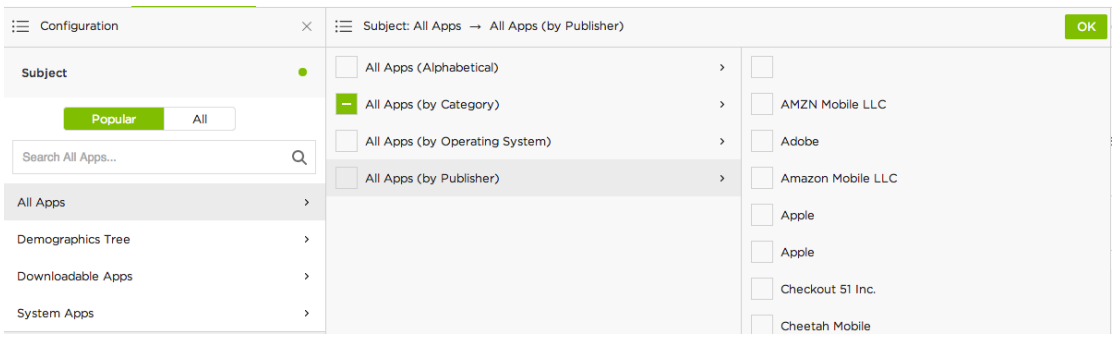


Figure 18: Subject Selector

Note that the Subject Selector diagrammed above features some distinct elements Quick Search and Subject Tree. The former is a search box which when the user types a value into it, filters the Subject Tree to display those subjects which match the user's search string. The latter is the hierarchical tree list of subjects. The structure of the subject tree is determined based on the organization schema selected by the user and the contents and structure are themselves defined in the Meta-data Platform.

5.4.7 Metrics

The Metrics component is actually composed of other components which are closely related and displayed alongside of each other. The contents of each of these components are highly dependent on the Report Template currently active and the metric(s) selected in the Metric Selector.

In general, the Metrics component (including all of its sub-components) can be visualized as follows:

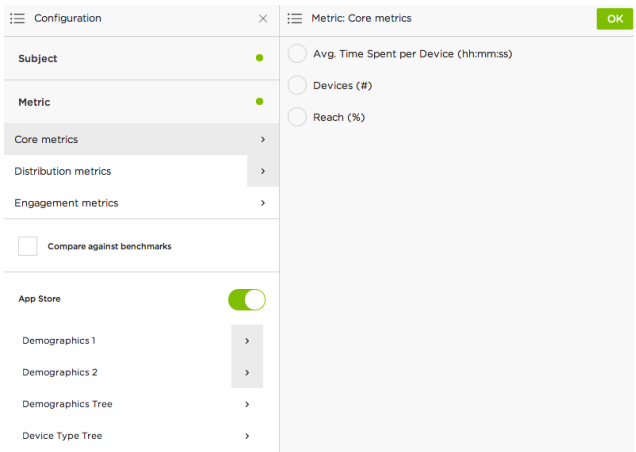


Figure 19: Metrics Component

The sub-components which may be included in the Metrics component are:
Metric Selector: This is the component used to actually select which metric(s) the user wishes to include in their Data Response.
Metric Parameters: This component is used to configure specialised parameters (particular to the metrics selected in the Metric Selector) to further narrow the data included in the Data Response.
Breakdown Criteria: This component is used to select the criteria by which the data should be broken out or organised in the Data Response. For example “Gender”, “Geography”, “Device Type”.
Benchmarks: This component is used to toggle whether a category benchmark for the metric(s) selected in the Metric Selector in returned in the Data Response.

5.5 Display Components

5.5.1 Data Chart

The Data Chart component is used to visually display a chart of the Data Response back to the user. The logic underlying each view should be able to take the data received in the Data Response and appropriately parse it to provide relevant instruction to the Data Chart component.

The basic functionality requirements of the Data Chart component should be as follows:

- Support Stacked Column Charts, Single-Column Charts, Line Charts, and Pie Charts
- Support up to 20 individual Data Series on a single chart
- Display dynamic ToolTips for individual data points communicating: metric name, series name, y-axis value and x-axis point
- Support the dynamic display of a legend alongside the chart
- Support export of the chart as the image (PNG, JPG, BMP)

HighchartsJS is used as as main approach for basic 2D charts. It offers very appealing and professional looking charts in the markets. Although HighchartsJS is built with the Javascript framework library, it is implemented in each a way that it does not totally on on particular framework. HighchartsJS is packaged with adapters, to make its interfaces to framework pluggable.

In general, this approach also has different advantages and disadvantages:

Advantages	**Disadvantages**
+ Ability to fully customize the look and feel of data visualization tools + Because library is provided as JS source code, it can be customised as necessary + Because query logic is separate, query performance can be optimised independently of the rendering tool	- Implementation requires actual development work, rather than configuration

Table 7. Advantages and disadvantages of using HighchartsJS

5.5.2 Data Table

The Data Table component is really only a logical component of the Verto Monitor. It relies on parsing the contents of the Data Response and rendering a tabular form of the resulting data. The Data Table component must adopt certain basic styling rules that will be applied throughout the design.

Position	All Apps (by Category)	Installations (#)	Previous	Change	Benchmark	Benchmark Previous	vs. Benchmark
1. -	UNKNOWN CATEGORY	2,177 preliminary	3,056 preliminary	↓ 879 preliminary	177 *	252 *	11 *
2. - *	GAMES AND KIDS	299 * preliminary	449 preliminary	↓ 150 * preliminary	177 *	252 *	1 *
3. - *	LIFESTYLE AND SHOPPING	296 * preliminary	387 preliminary	↓ 91 * preliminary	177 *	252 *	1 *
4. - *	SOCIAL NETWORKING	246 * preliminary	339 preliminary	↓ 93 * preliminary	177 *	252 *	0 *
5. ↑1 *	UTILITIES	107 * preliminary	154 * preliminary	↓ 47 * preliminary	177 *	252 *	0 *
6. ↓1 *	ENTERTAINMENT	103 * preliminary	160 * preliminary	↓ 57 * preliminary	177 *	252 *	0 *
7. - *	PHOTO AND VIDEO	55 * preliminary	88 * preliminary	↓ 33 * preliminary	177 *	252 *	-1 *
8. - *	PRODUCTIVITY	46 * preliminary	68 * preliminary	↓ 22 * preliminary	177 *	252 *	-1 *
9. - *	BUSINESS	43 * preliminary	65 * preliminary	↓ 22 * preliminary	177 *	252 *	-1 *
10. - *	FINANCE	42 * preliminary	62 * preliminary	↓ 20 * preliminary	177 *	252 *	-1 *

Figure 20: Basic Data Table

5.6 Report

5.6.1 Ranking Report

Overview

The purpose of the Ranking Report is to provide the user with a sorted list of Subjects, sorted according to the metric that the user selected.

Configuration Components

The Ranking Report should have the following Configuration Components:

Configuration Component	Default Value
Time Period	Last month
Day Filter	All Days
Daypart Filter	All Dayparts
Target Group Filter	US Population 18+
Subject Selector	
Metric Selector	Users (#)
Metric Parameter	
Benchmarks	TRUE
Add Comparison	

Table 8. Ranking report's default configuration component

Display Components

The Ranking Report should only feature a Data Table as its display component. This Data Table should be sortable in ascending / descending order by any of its columns.

Pos.	(+/-)	Subject	Avg. Sessions / User (#)	Change (%)	Cat. Benchmark	Change (%)
1	↑ 3	Facebook	12.3	↑ 10%	7.6	↑ 7%
2	↓ 1	Twitter	12.3	↑ 5%	7.6	↑ 7%
3	--	Tumblr	12.3	↓ 3%	7.6	↑ 7%
4	↑ 1	Instagram	12.3	↑ 6%	7.6	↑ 7%
5	↑ 1	YouTube	12.3	↑ 2%	7.6	↑ 7%

Figure 21: Ranking Data Table with up and down arrow

Layout

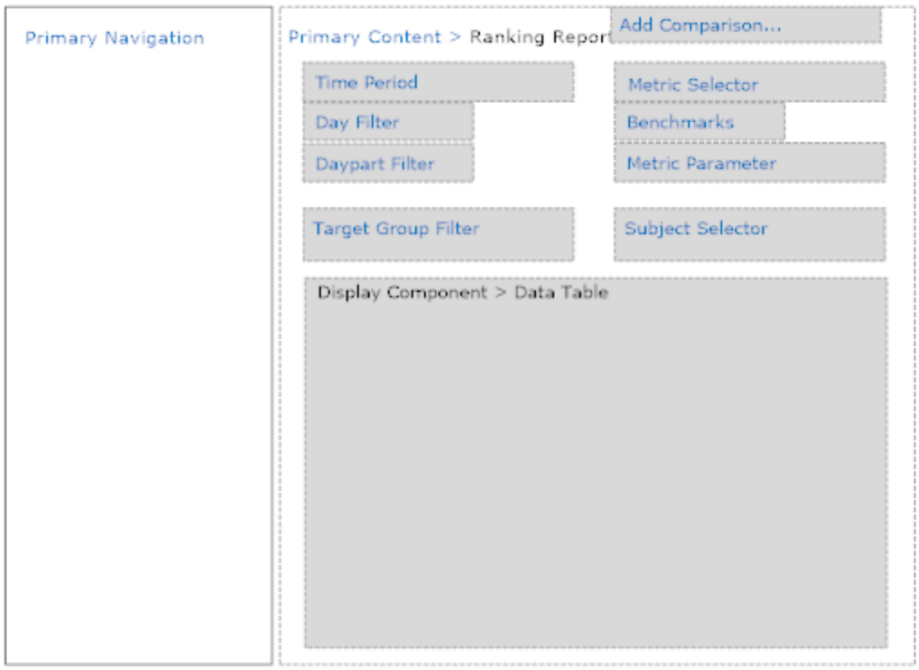


Figure 22: Basic layout of ranking report

Ranking report is a special report without any data chart, but only data table. It also includes some basic components for configurations.

5.6.2 Share Report

Overview

The purpose of the Share Report is to show the performance of various Subjects in comparison to their competitors

Configuration Components

The Share Report should have the following Configuration Components:

Configuration Component	**Default Value**
Time Period	Last month
Day Filter	All Days
Daypart Filter	All Dayparts

Target Group Filter	US Population 18+
Subject Selector	
Metric Selector	Users (#)
Breakdown Criteria	None
Metric Parameter	
Benchmarks	Unchecked
Add Comparison	
Resolution	Weekly

Table 9. Share report's default configuration component

Display Components

The Share Report should display two Data Charts and Data Tables as its display components.

The Data Chart to display will depend on the Resolution and Time Period selected in the Configuration Components. In addition, two Data Tables should display be displayed, each listing Subjects in rows and time along columns.

Two Data Tables should display each listing Subjects in rows and time along columns. It should resemble the following:

Users (#)	Jan-2013	Feb-2013	Mar-2013	Apr-2013	May-2013	Jun-2013
Facebook	15%	38%	23%	8%	32%	29%
Twitter	46%	57%	66%	84%	42%	52%
Tumblr	39%	5%	11%	8%	26%	19%

Users (#)	Jan-2013	Feb-2013	Mar-2013	Apr-2013	May-2013	Jun-2013
Facebook	123,458	123,458	123,458	123,458	123,458	123,458
Twitter	123,458	123,458	123,458	123,458	123,458	123,458
Tumblr	123,458	123,458	123,458	123,458	123,458	123,458

Figure 23: An example of two table in share report

Layout

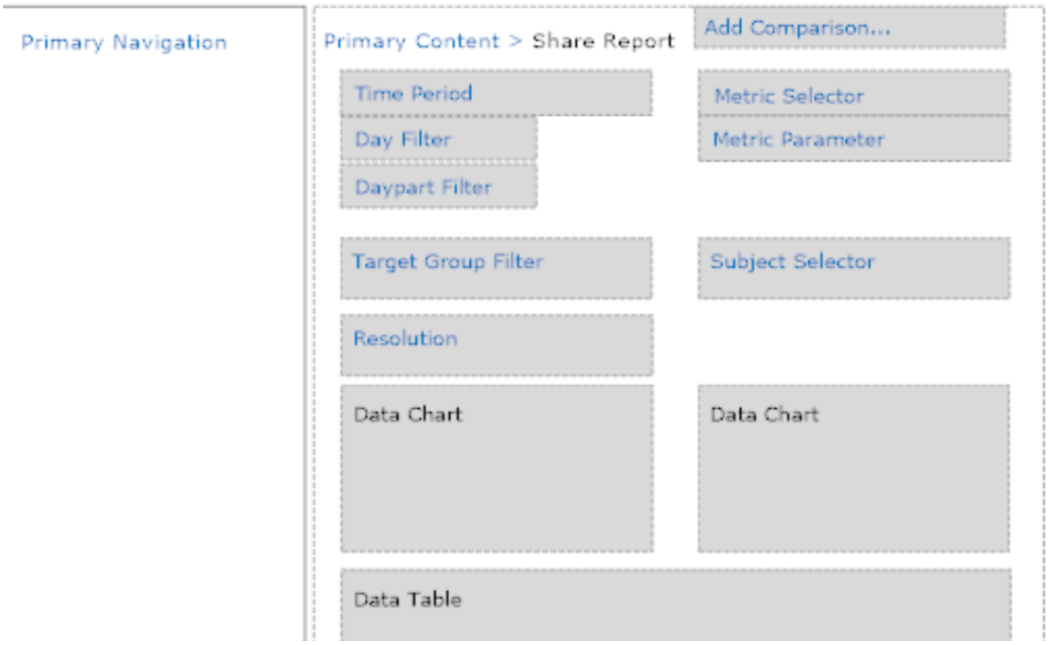


Figure 24: Basic layout of share report

Layout in Share Report comprises of some basic components and two data charts and two tables. Stacked column and column chart are used mainly in Usage Trend Report.

5.6.3 Usage Trend Report

Overview

The Usage Trend Report is used to show nominal usage statistics over time for selected Subjects and populations

Configuration Components

Configuration Component	Default Value
Time Period	Last month
Day Filter	All Days
Daypart Filter	All Dayparts

Target Group Filter	US Population 18+
Subject Selector	
Metric Selector	Users (#)
Metric Parameter	
Benchmarks	TRUE
Add Comparison	
Resolution	Monthly

Table 10. Usage Trend report's default configuration component

Display Components

The Usage Trend Report should display the resulting data in both a single Data Chart and a single Data Table. Data Chart should either be displayed as column chart, a stacked column chart, or a line chart. The decision between these chart types should be made based on following logic in table 11:

Condition	**Display**
The Resolution >= Time Period and Breakdown Criteria == None	Column Chart with Subjects on X-Axis
The Resolution >= Time Period and Breakdown Criteria <> None	Stacked Column Chart with Subjects on X-Axis and Breakdown Criteria as grouping
The Resolution < Time Period and Breakdown Criteria <> None	Stacked Column Chart with Subjects on X-Axis and Subjects as grouping
The Resolution < Time Period And Breakdown Criteria == None And Number of Subjects Selected == 1	Column Chart with Time on X-Axis
The Resolution < Time Period And Breakdown Criteria == None And Number of Subjects Selected > 1	Line Chart with Time on X-Axis with Subject as grouping
In each case, if Benchmarks are enabled, the Benchmark data series should be visualised as a line chart.	

Table 11. Conditions to display appropriate Data Chart

Data Table

The Data Table should display the Subjects in rows with a breakdown according to the Breakdown Criteria (if specified), with the individual data points in columns grouped according to Date/Time

If Benchmarks are enabled, then the first and last rows of the Data Table should display a subtotal row as the benchmark:

	Jan-2013	Feb-2013	Mar-2013	Apr-2013	May-2013	Jun-2013
Cat. Benchmark						
Men	123,456	123,456	123,456	123,456	123,456	123,456
Women	654,321	654,321	654,321	654,321	654,321	654,321
Facebook						
Men	123,456	123,456	123,456	123,456	123,456	123,456
Women	654,321	654,321	654,321	654,321	654,321	654,321
Twitter						
Men	123,456	123,456	123,456	123,456	123,456	123,456
Women	654,321	654,321	654,321	654,321	654,321	654,321
Cat. Benchmark						
Men	123,456	123,456	123,456	123,456	123,456	123,456
Women	654,321	654,321	654,321	654,321	654,321	654,321

Figure 25: An example of data table in Usage Trend report

Layout

Layout in Usage Trend Report has some basic components and only one data chart and one table. Column chart is used mainly in Usage Trend Report.

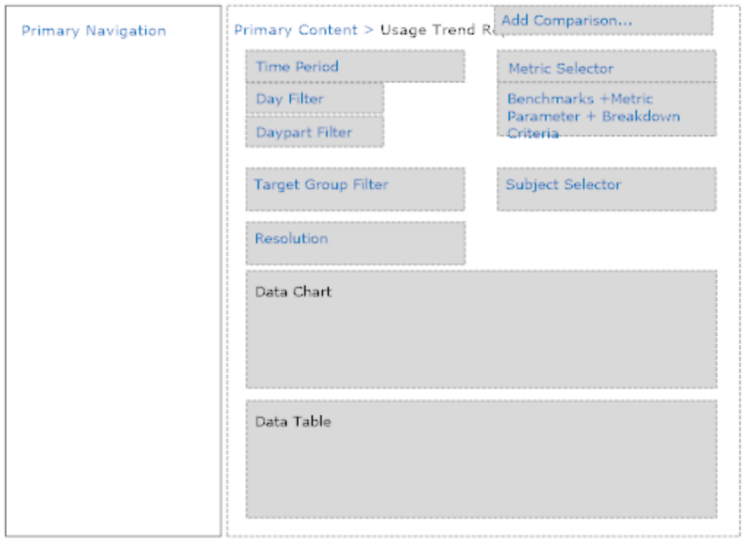


Figure 26: Basic layout of Usage Trend Report

6 Conclusion

The web is now far different from some years ago since single-page web application has become a new trend for enterprise applications. As more and more people use web applications, a need for an interactive experience and realtime, experience is needed from a technical perspective, a single-page web application has emerged and to meet those requirements. Digital media measurement services are part of this trend. Verto Dashboard is the main product of Verto Analytics and it offers cross platform digital media measurement and audience usage research. It received secure funding $5.4M on April 2014 [16] and $2.5M on September 2014 [17.]

Dashboard project was started on June 2013 and it continued in development stage. Receiving two secure fundings after one year is quite success. The ultimate goal of this project is providing internet media measurement services around smart digital devices. Verto Monitor is the first product to publish a digital media measurement service that is designed, and built, to be multi-screen by nature, reporting on the significant subjects of both the platform (devices, carries, operating systems) and media domains (apps, sites, publishers, properties), on both the distribution and engagement layer, through one integrated methodology and service.

The final version of Verto Monitor can establish a strong track record in the digital measurement field and help customers to do tactical and strategic decisions, providing insights though hard measurement of data visualizations in the multi-screen digital industry comprising of audience acquisition, media sales and audience research, product development and consumer insights, competitive insights, market trend analyses, app store analytics, conversion modeling and Investment decisions.

All background knowledge and experience will play a vital role in the future and challenges of the project.

References

[1] Mikowski M, Powell J. Single Page Web Applications. NY, USA: Manning Publications; 2013. p.8-9

[2] Ruby S, Thomas D, Hansson D. Agile Web Development with Rails. 4th ed. USA: Pragmatic Programmers, LLC; 2012. p.30

[3] Why AngularJS for Web Apps? Why Now? [online]. Alex Castillo; 2014
URL: http://www.business2community.com/tech-gadgets/angularjs-web-apps-now-0932000. Accessed 25 October 2014

[4] Freeman A. Pro AngularJS: Learn to harness the power of modern web browsers from within your application's code. In: Putting AngularJS in Context.USA: Apress Publisher; 2014. p.45-54

[5] Lerner A. ng-book: The complete book on AngularJS. In: Introducing Data Binding in AngularJS. USA; 2013. p11-12

[6] AngularJS Developer Guide on Data Binding [online]
URL: https://docs.angularjs.org/guide/databinding. Accessed 25 October 2014.

[7] Lerner A. ng-book: The complete book on AngularJS. In: Modules. USA; 2013. p18-19

[8] Lerner A. ng-book: The complete book on AngularJS. In: Scopes. USA; 2013. p20-24

[9] AngularJS Developer Guide on Conceptual Overview[online]
URL: https://docs.angularjs.org/guide/concepts. Accessed 25 October 2014.

[10] Understanding Controllers [online]
URL: https://docs.angularjs.org/guide/controller. Accessed 25 October 2014.

[11] Services [online]
URL: https://docs.angularjs.org/guide/services. Accessed 25 October 2014.

[12] Lerner A. ng-book: The complete book on AngularJS. In: Services. USA; 2013. p157-172

[13] Lerner A. ng-book: The complete book on AngularJS. In: Events. USA; 2013. p373-378

[14] "Representational state transfer" Wikipedia, The Free Encyclopedia. Wikimedia Foundation, Inc; 25 October 2014.

URL: http://en.wikipedia.org/wiki/Representational_state_transfer. Accessed 25 October 2014.

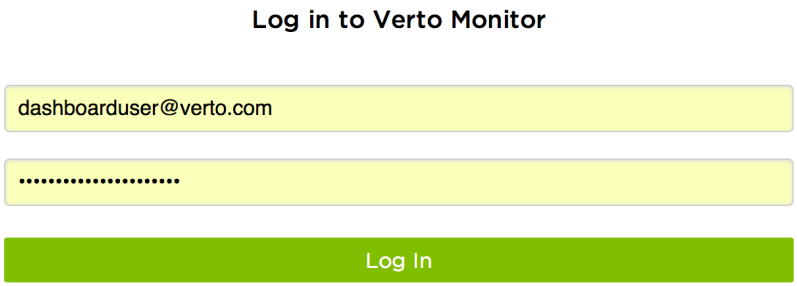
[15] W3Schools. HTTP status messages [online].
URL: http://www.w3schools.com/tags/ref_httpmessages.asp. Accessed 25 October 2014.

[16] Verto Analytics raises $5.4M to launch comprehensive syndicated digital measurement service [online]
URL: http://vertoanalytics.com/news-april-24-1.html. Accessed 26 October 2014

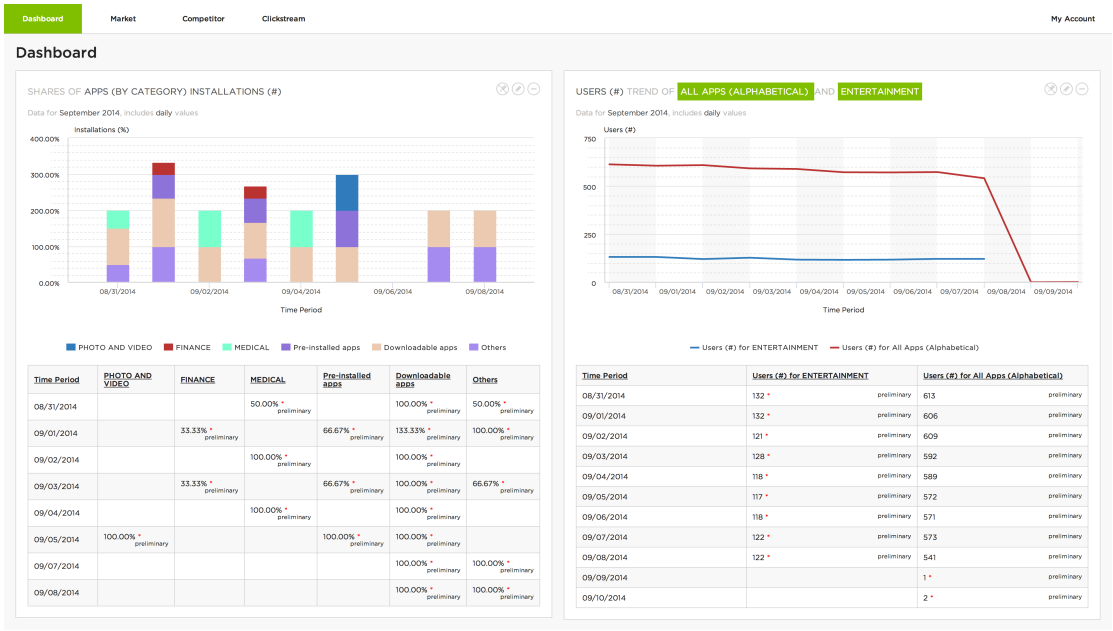
[17] Mysql investor backs Verto Analytics to revolutionize the digital media measurement industry with $2.4M funding [online]
URL: http://vertoanalytics.com/news-september-16-1.html. Accessed 26 October 2014

Appendix

Appendix 1: Login Page



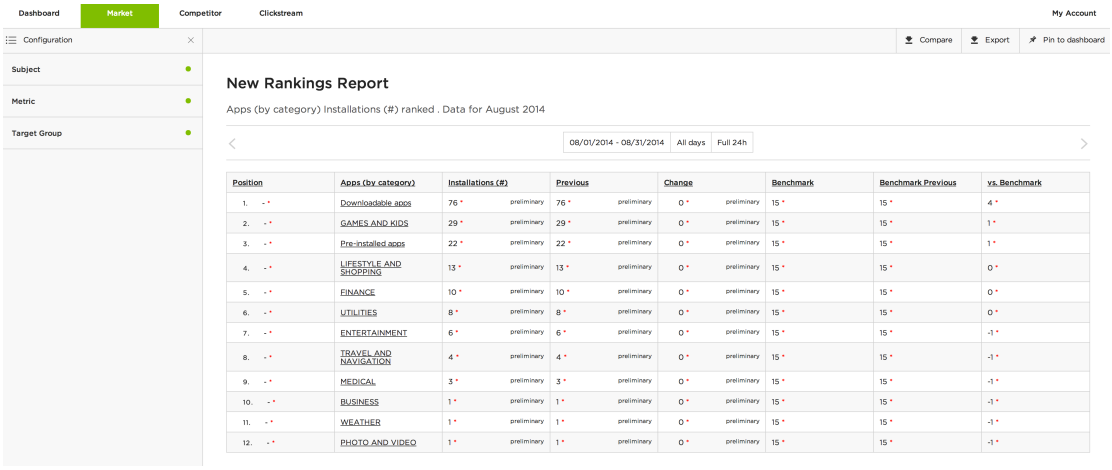
Appendix 2: Dashboard



Time Period	PHOTO AND VIDEO	FINANCE	MEDICAL	Pre-installed apps	Downloadable apps	Others
08/31/2014			50.00% * preliminary		100.00% * preliminary	50.00% * preliminary
09/01/2014		33.33% * preliminary		66.67% * preliminary	133.33% * preliminary	100.00% * preliminary
09/02/2014			100.00% * preliminary		100.00% * preliminary	
09/03/2014		33.33% * preliminary		66.67% * preliminary	100.00% * preliminary	66.67% * preliminary
09/04/2014			100.00% * preliminary		100.00% * preliminary	
09/05/2014	100.00% * preliminary			100.00% * preliminary	100.00% * preliminary	
09/07/2014					100.00% * preliminary	100.00% * preliminary
09/08/2014					100.00% * preliminary	100.00% * preliminary

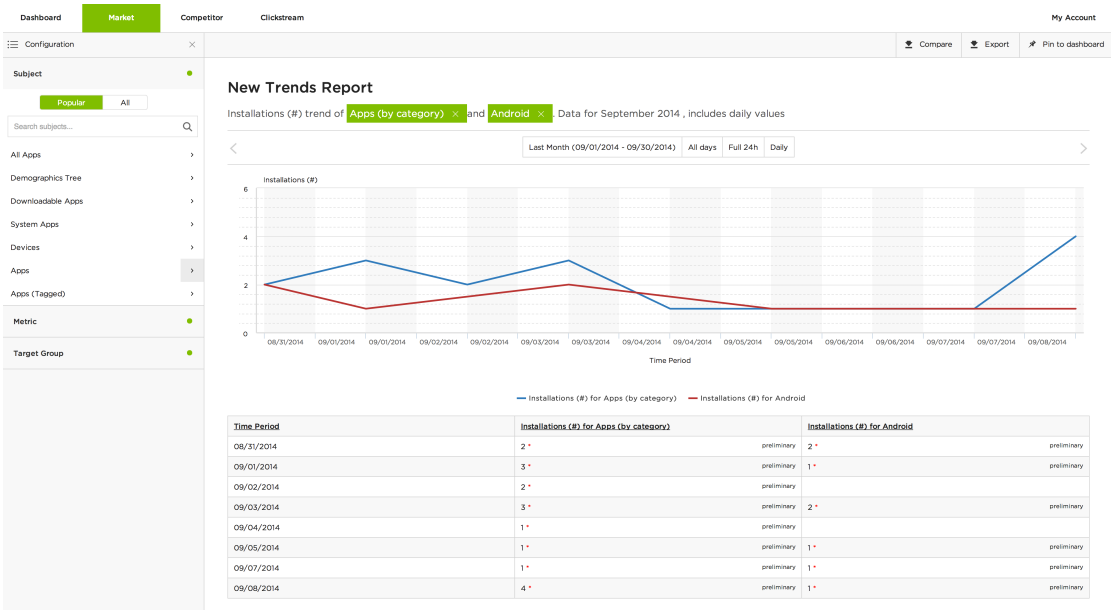
Time Period	Users (#) for ENTERTAINMENT	Users (#) for All Apps (Alphabetical)
08/31/2014	132 * preliminary	613 preliminary
09/01/2014	132 * preliminary	606 preliminary
09/02/2014	121 * preliminary	609 preliminary
09/03/2014	128 * preliminary	592 preliminary
09/04/2014	118 * preliminary	589 preliminary
09/05/2014	117 * preliminary	572 preliminary
09/06/2014	118 * preliminary	571 preliminary
09/07/2014	122 * preliminary	573 preliminary
09/08/2014	122 * preliminary	541 preliminary
09/09/2014		1 * preliminary
09/10/2014		2 * preliminary

Appendix 3: Ranking Report



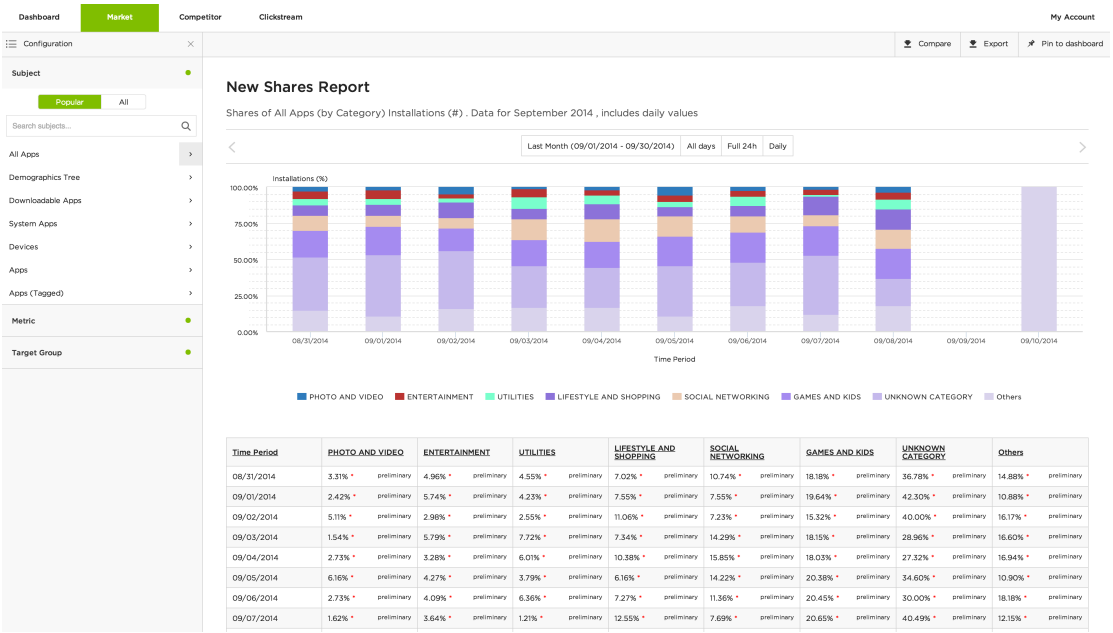
Position	Apps (by category)	Installations (#)	Previous	Change	Benchmark	Benchmark Previous	vs. Benchmark
1. - *	Downloadable apps	76 * preliminary	76 * preliminary	0 * preliminary	15 *	15 *	4 *
2. - *	GAMES AND KIDS	29 * preliminary	29 * preliminary	0 * preliminary	15 *	15 *	1 *
3. - *	Pre-installed apps	22 * preliminary	22 * preliminary	0 * preliminary	15 *	15 *	1 *
4. - *	LIFESTYLE AND SHOPPING	13 * preliminary	13 * preliminary	0 * preliminary	15 *	15 *	0 *
5. - *	FINANCE	10 * preliminary	10 * preliminary	0 * preliminary	15 *	15 *	0 *
6. - *	UTILITIES	8 * preliminary	8 * preliminary	0 * preliminary	15 *	15 *	0 *
7. - *	ENTERTAINMENT	6 * preliminary	6 * preliminary	0 * preliminary	15 *	15 *	-1 *
8. - *	TRAVEL AND NAVIGATION	4 * preliminary	4 * preliminary	0 * preliminary	15 *	15 *	-1 *
9. - *	MEDICAL	3 * preliminary	3 * preliminary	0 * preliminary	15 *	15 *	-1 *
10. - *	BUSINESS	1 * preliminary	1 * preliminary	0 * preliminary	15 *	15 *	-1 *
11. - *	WEATHER	1 * preliminary	1 * preliminary	0 * preliminary	15 *	15 *	-1 *
12. - *	PHOTO AND VIDEO	1 * preliminary	1 * preliminary	0 * preliminary	15 *	15 *	-1 *

Appendix 4: Usage Trend Report



Time Period	Installations (#) for Apps (by category)	Installations (#) for Android
08/31/2014	2 * preliminary	2 * preliminary
09/01/2014	3 * preliminary	1 * preliminary
09/02/2014	2 * preliminary	
09/03/2014	3 * preliminary	2 * preliminary
09/04/2014	1 * preliminary	
09/05/2014	1 * preliminary	1 * preliminary
09/07/2014	1 * preliminary	1 * preliminary
09/08/2014	4 * preliminary	1 * preliminary

Appendix 5: Share Report



Time Period	PHOTO AND VIDEO	ENTERTAINMENT	UTILITIES	LIFESTYLE AND SHOPPING	SOCIAL NETWORKING	GAMES AND KIDS	UNKNOWN CATEGORY	Others
08/31/2014	3.31% * preliminary	4.96% * preliminary	4.55% * preliminary	7.02% * preliminary	10.74% * preliminary	18.18% * preliminary	36.78% * preliminary	14.88% * preliminary
09/01/2014	2.42% * preliminary	5.74% * preliminary	4.23% * preliminary	7.55% * preliminary	7.55% * preliminary	19.64% * preliminary	42.30% * preliminary	10.88% * preliminary
09/02/2014	5.11% * preliminary	2.98% * preliminary	2.55% * preliminary	11.06% * preliminary	7.23% * preliminary	15.32% * preliminary	40.00% * preliminary	16.17% * preliminary
09/03/2014	1.54% * preliminary	5.79% * preliminary	7.72% * preliminary	7.34% * preliminary	14.29% * preliminary	18.15% * preliminary	28.96% * preliminary	16.60% * preliminary
09/04/2014	2.73% * preliminary	3.28% * preliminary	6.01% * preliminary	10.38% * preliminary	15.85% * preliminary	18.03% * preliminary	27.32% * preliminary	16.94% * preliminary
09/05/2014	6.16% * preliminary	4.27% * preliminary	3.79% * preliminary	6.16% * preliminary	14.22% * preliminary	20.38% * preliminary	34.60% * preliminary	10.90% * preliminary
09/06/2014	2.73% * preliminary	4.09% * preliminary	6.36% * preliminary	7.27% * preliminary	11.36% * preliminary	20.45% * preliminary	30.00% * preliminary	18.18% * preliminary
09/07/2014	1.62% * preliminary	3.64% * preliminary	1.21% * preliminary	12.55% * preliminary	7.69% * preliminary	20.65% * preliminary	40.49% * preliminary	12.15% * preliminary