**CHAPTER 6**

**DASHBOARDS**

**6.1 Introduction**

A dashboard is a data visualization tool that tracks, analyses, and displays KPIs, metrics, and critical data points. Dashboards empower both technical and non-technical users to understand and leverage business intelligence to make more informed decisions. Users actively participate in the analytics process by compiling data and visualizing trends or occurrences and uncovering an objective view of performance metrics that can be immediately understood.

Dashboards feature visualized data via charts, tables, and gauges. Viewers use these visualizations to monitor the health of the organization against established goals and industry benchmarks.

* **Dashboards are fuelled by Business Questions.**
* **Dashboards can focus on presenting operational and analytical data.**
* **Dashboards present interactive data visualizations.**

**6.2 Dash by Plotly**

Dash gives a point-&-click interface to models written in Python, R, and Julia - vastly expanding the notion of what's possible in a traditional "dashboard." With Dash apps, data scientists and engineers put complex Python analytics in the hands of business decision makers and operators.

Dash will help you build dashboards quickly. If you’re used to analyzing data or building data visualizations using Python, then Dash will be a useful addition to your toolbox.

Three technologies constitute the core of Dash:

1. **Flask** supplies the web server functionality.
2. **React.js** renders the user interface of the web page.
3. **Plotly.js** generates the charts used in your application.

Written on top of Plotly.js and React.js, Dash is ideal for building and deploying data apps with customized user interfaces in pure Python, R, or Julia. It's particularly suited for anyone who works with data.

Through a couple of simple patterns, Dash abstracts away all of the technologies and protocols that are required to build a full-stack web app with interactive data visualization.

Dash is simple enough that you can bind a user interface to your Python, R, or Julia code in less than 10 minutes.

Dash apps are rendered in the web browser. You can deploy your apps to VMs or Kubernetes clusters and then share them through URLs. Since Dash apps are viewed in the web browser, Dash is inherently cross-platform and mobile ready.

There is a lot behind the framework. To learn more about how it is built and what motivated Dash, read our post Dash is React for Python.

Dash is an open source library released under the permissive MIT license. Plotly develops Dash and also offers a platform for writing and deploying Dash apps in an enterprise environment.

**6.3 Dash Architecture:**

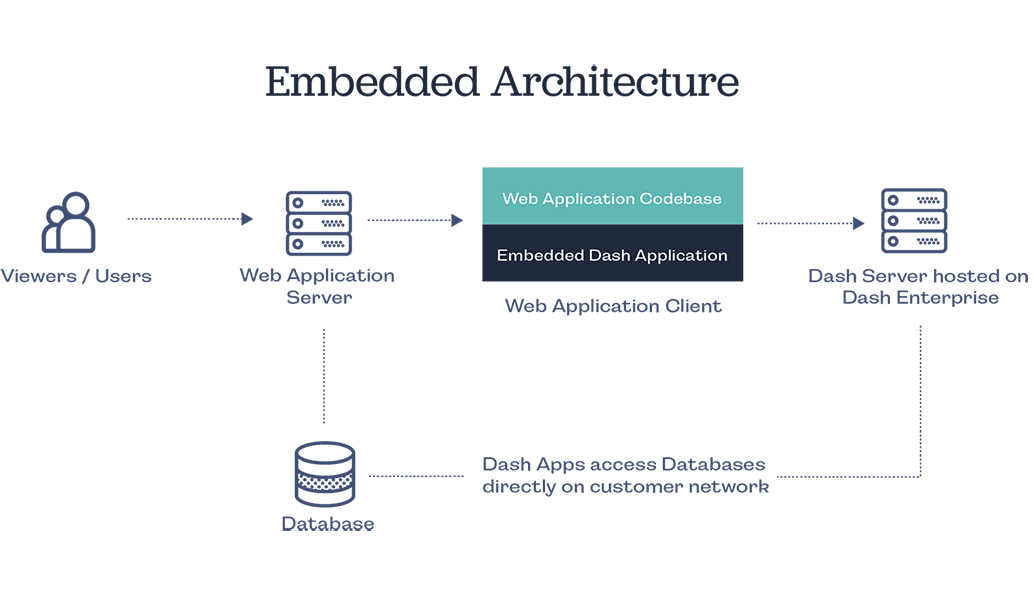
Flask and React

Dash applications are web servers running Flask and communicating JSON packets over HTTP requests. Dash’s frontend renders components using React.js, the Javascript user-interface library written and maintained by Facebook.

Flask is great. It’s widely adopted by the Python community and deployed in production environments everywhere. The underlying instance of Flask and all of its configurable properties is accessible to Dash app developers. For advanced developers, Dash apps can be extended through the rich set of Flask Plugins as well.

React is fantastic too. At Plotly, we’ve rewritten our entire web-platform and our online chart editor with React. One of the incredible things about React is how prolific and talented the community is. The open source React community has published thousands of high quality interactive components, from Dropdowns to Sliders to Calendar Pickers to Interactive Tables.

Dash leverages the power of Flask and React, putting them to work for Python data scientists who may not be expert Web programmers.

****

**Figure 6.1 Embedded Architecture of Dash**

**Source: https://plotly.com/dash/embedding/**

Diagram

Description automatically generated

**Figure 6.2 Dash Architecture**

**Source: https://towardsdatascience.com/plotly-dash-or-react-js-plotly-js-b491b3615512**

**6.4 Dash App Creation:**

Dash is Python framework for building web applications. It built on top of Flask, Plotly.js, React and React Js. It enables you to build dashboards using pure Python. Dash is open source, and its apps run on the web browser. In this tutorial, we introduce the reader to Dash fundamentals and assume that they have prior experience with Plotly.

Dash Installation

In order to start using Dash, we have to install several packages.

1. The core dash backend.
2. Dash front-end
3. Dash HTML components
4. Dash core components
5. Plotly

**6.4.1 Dash App Layout**

A Dash application is usually composed of two parts. The first part is the layout and describes how the app will look like and the second part describes the interactivity of the application. Dash provides HTML classes that enable us to generate HTML content with Python. To use these classes, we need to import dash\_core\_components and dash\_html\_components. You can also create your own custom components using Javascript and React Js.

**HTML Components**

Just like in Flask we initialize Dash by calling the Dash class of dash. Once that is done we can create the layout for our application. We use the Div class from the dash\_html\_components to create an HTML Div. We then use the HTML components to generate HTML components such as H1, H2 etc. dash\_html\_components has all HTML tags. In order to create a graph on our layout, we use the Graph class from dash\_core\_components. Graph renders interactive data visualizations using plotly.js. The Graph class expects a figure object with the data to be plotted and the layout details. Dash also allows you to do stylings such as changing the background color and text color. You can change the background by using the style attribute and passing an object with your specific color.

In our case, we have defined a color dictionary with the background and text color we would like. Similarly, we can change the layout background using the plot\_bgcolor attribute.

In HTML the style property is specified using a semicolon, but in Dash, a dictionary is supplied. The keys in the dictionary are camelCased e.g text-align is textAlign. Instead of using classes like in HTML, className is used in Dash.

**Generating Scatter Plots**

In order to plot a scatter plot, we import the normal dash components as previously done. We also need to import Plotly graph\_objs in order to plot the scatter plot. As mentioned previously we use the Div class and Graph components from Dash in order to accomplish this. The Graph component takes a figure object which has the data and the layout description. We plot the scatter plot using graph\_objs scatter property. In order to make sure the plot is a scatter plot we pass a mode attribute and set it as markers. Otherwise, we would have lines on the graph.

**Core Components**

Next, let's look at some dash\_core\_components that you will encounter when using Dash. You can generate a drop down as shown below. You do by calling Dropdown off dash\_core\_components and passing the options as a list of dictionaries. You can set the default value using the values attribute and passing in the default option.

**Authentication**

Dash provides authentication through a separate package called dash-auth. It provides two modes of authentication HTTP Basic Auth and Plotly OAuth. In Basic Auth, you hardcode a set of usernames and passwords in your application. This method has some challenges such as users cannot log out of your application, users cannot create accounts or change passwords, and you are responsible for safely storing the usernames and passwords in your code. Plotly OAuth provides authentication through your online Plotly account, and it's not free.

**Dash Bootstrap Components**

dash-bootstrap-components is a library of Bootstrap components for Plotly Dash, that makes it easier to build consistently styled apps with complex, responsive layouts. Dash Bootstrap Components for Python can be easily installed with pip or conda.

Once installed, just link a Bootstrap stylesheet and start using the components exactly like you would use other Dash component libraries.

Bootstrap components are available as native Dash components to let you easily incorporate them into your Dash apps. Each component exposes a number of props to let you control the behaviour with Dash call-backs.

Dash Bootstrap Components is compatible with any Bootstrap v4 stylesheet of your choice. Learn how to customise the look of your app using the bundled themes or your own custom themes.

**Use Case Diagram of Dashboard:**

Current Trending Technologies

Demographics

In Future Trending

Technologies

User

**Figure 6.3 Use Case Diagram**

**6.4.2 Graphical User Interface (GUI)**

Graphical user interfaces would become the standard of user-cantered design in software application programming, providing users the capability to intuitively operate computers and other electronic devices through the direct manipulation of graphical icons such as buttons, scroll bars, windows, tabs, menus, cursors, and the mouse pointing device. Many modern graphical user interfaces feature touchscreen and voice-command interaction capabilities.

Graphical user interface design principles conform to the model–view–controller software pattern, which separates internal representations of information from the manner in which information is presented to the user, resulting in a platform where users are shown which functions are possible rather than requiring the input of command codes. Users interact with information by manipulating visual widgets, which are designed to respond in accordance with the type of data they hold and support the actions necessary to complete the user’s task.

**Authentication:**

**Graphical user interface, application

Description automatically generated**

**Figure 6.4 Authentication Page**

**Current Trending Technologies:**

**Chart, bar chart

Description automatically generated**

**Figure 6.5 Current trending tech. page**

**In Future Trending Technologies:**

**Graphical user interface, chart, application, bar chart

Description automatically generated**

**Figure 6.6 Future Trending tech. page**

**Demographics:**

**Graphical user interface, chart, application

Description automatically generated**

**Figure 6.7 Demographics page**