**Data Visualization**

Presentations and report building based on real-time data from the MongoDB database. This development work is for the proof-of-concept purposes.

**Interactive Data Visualization**

* Quick Analytics
* Simple to use
* Changing view and variables on the fly
* Update automatically
* Dashing Dashboards

**Overview of the Application**

The given code is a Dash application that connects to a MongoDB database, retrieves a collection of documents, and displays the data of the collection in Plotly charts.

**The System Architecture**

The system architecture for the code consists of the following components:

* [**MongoDB database**](https://www.mongodb.com/): a NoSQL database that stores the data being plotted in the line chart.
* [**Pymongo**](https://pymongo.readthedocs.io/en/stable/): a Python library that allows the Dash application to connect to and query the MongoDB database.
* [**Pandas**](https://pandas.pydata.org/): a Python library that is used to store and manipulate the data retrieved from the MongoDB database in a DataFrame.
* [**Plotly**](https://plotly.com/python/): a data visualization library that is used to create the line chart and other types of charts in the Dash application.
* [**Dash**](https://plotly.com/dash/): a Python framework for building web applications. It is used to build the user interface for the application, which consists of different types of charts and the drop-down menu for selecting the options.
* An open-source library
* Framework for building web analytic applications
* Written on top of Flask, Pltly.js, and React.js
* Developed by Plotly
* A platform for managing apps in an enterprise environment
* Layout and Interactivity of the app

**Flow**

Data visualization (plotly)

Data processing (pymongo, pandas)

Dash app

Data source (MongoDB)

1. The client sends a request to the [Dash](https://plotly.com/dash/) application to display the report data for different selected options.
2. The Dash application connects to and queries the MongoDB database using the [Pymongo](https://pymongo.readthedocs.io/en/stable/) library and retrieves the related collection data.
3. The data is then stored in a [Pandas](https://pandas.pydata.org/) DataFrame and used to create the line chart using [Plotly](https://plotly.com/python/). The application processes the data using the pandas library, and creates a drop-down menu using the [dash](https://plotly.com/dash/) and [dcc](https://dash.plotly.com/dash-core-components) libraries.
4. The application then creates a graph using the plotly.graph\_objs library, and displays the graph on the web page using the html library.
5. The user can then select an option (e.g., experiment id, cell panel, etc.) from the dropdown menu, which will trigger the corresponding function through a callback. This function filters the data based on the selected cell panel and updates the graph accordingly.
6. The updated graph is then displayed on the web page for the user to view.

**Demo**