DESIGN PATTERNS?

<https://www.tutorialspoint.com/design_pattern>

JAVA?

-JAVA.DOCX

UI?

-not done

UNIX?

-not done

DB?

-not done

APPTITUDE?

-not done

PROJECTS?

CODING STRUCTURE OF BACKEND AND UI?

COMPANY WORK FLOW?

ITDA

Data-visualization

RIM

React

Redux

Javascript githup

npm

node

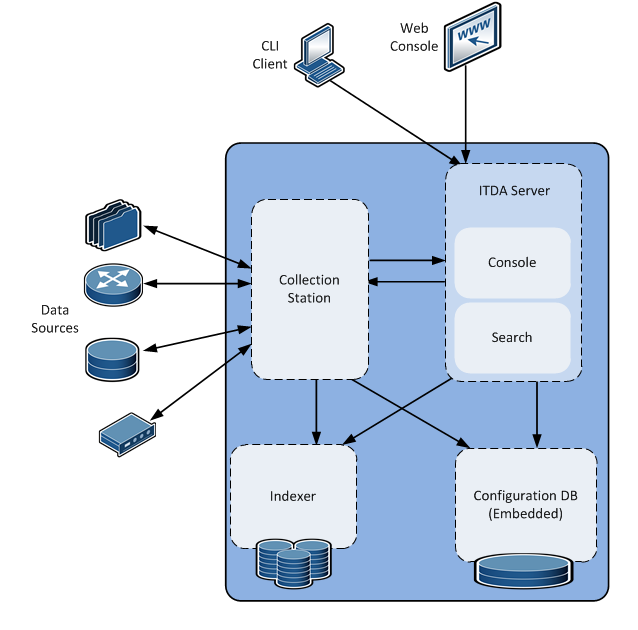
Resume go through

My Roles & Resonsibilies

ITDA

\*\*\*\*\*\* Architechture \*\*\*\*\*\*\*

The following diagram illustrates the overall architecture of the BMC TrueSight IT Data Analytics product.



The various components of the product are described as follows:

|  |  |
| --- | --- |
| **Component** | **Description** |
| Collection Station | The Collection Station is responsible for collecting and receiving data from target hosts and then indexing the data using the Indexer.  This component fetches data collector details from the Configuration Database and then depending on the data collector configuration fetches local (or remote) log file data. All data collectors created in the system translate into actions performed by the Collection Station.  All Collection Agents read data from the data sources based on the data collector configuration; the Collection Agents and then forward this data to the Collection Station for indexing.  This component interacts with the Indexer and the Configuration Database. All Collection Agents interact with the Collection Station for data collector configuration and sending the collected log data. |
| Console Server | This is the administrative console for the BMC TrueSight IT Data Analytics product.  The Console Server component serves as the user interface for performing all actions in the BMC TrueSight IT Data Analytics product. The Console Server also acts as the gatekeeper for all actions that can be performed by using the CLI. The authentication and authorization checks for all operations are performed by the Console Server.  It enables various functions such as, configuration of data collectors, searching across all the stored log data, collaboration by using the workspace, and creating saved searches for creating views and scheduling notifications.  The Console Server stores all the configuration details in the Configuration Database. This components interacts with all the other product components. |
| Search | The Search component is responsible for processing searches that are executed by the user. In addition, this component is responsible for executing the scheduled notifications and reports. The data displayed on the **Views** tab is processed by the Search component.  The Search component searches data available in the Indexer component. The primary function of this component is to understand the user’s search query and fetch required data from the Indexer by applying search commands.  This component can interact will multiple Indexer instances and aggregate the search results. This component interacts with the Indexer and Configuration Database. |
| Indexer | The Indexer component acts as an internal datastore used by BMC TrueSight IT Data Analytics, for storing all the data that is collected by using Collection Stations and Collection Agents.  This component provides the indexing capability for all the log data. The Indexer stores the data in a way that makes searching it easier. It also stores the raw log data that can be seen in the search results. All the data is stored in a time series way to enable faster searching. The Collection Station interacts with the Indexer to send all the configured log data. Primarily the Search and Collection Station components interact with the Indexer for searching and indexing data.  Each Indexer operates within the Indexer cluster, which is a group of multiple Indexers that can communicate with each other and help you collect and search data more effectively. Data collected and searched is distributed across all the Indexers present in the cluster. For more information, see [Indexer cluster](https://docs.bmc.com/docs/display/itda11/Indexer+cluster). |
| Configuration Database | All configurations of data collectors, views, and saved searches that are used in BMC TrueSight IT Data Analytics are stored in the Configuration Database.  The Configuration Database is a lightweight component in terms of resource usage.  It does not contain the data collected by using the Collection Stations and Collection Agents. |

\*\*\*\*\*\* Search \*\*\*\*\*\*\*

Speed up MTTR by using advanced analytics with your log data

=>Compare data

-Focus on whats not normal by comparing to a known normal(good)

-Visually identify the needle in the haystack in seconds

-Drill down on differences with a single click

=>Coalesce data (classify data based on pattern)

-Reduce millions of log messages to dozens

-Identify "top talkers"

-Expose all messages within a pattern with a single click

=>Anomalies

-Takes coalescs to the next level

-Is there anything unsual in the log data? More of a particular type of mesage? Less?

-Anything I have never, or have rarely, seen before?

\*\*\*\*\*\* Collection \*\*\*\*\*\*\*

Monitor File on Collection Agent

Monitor File over SSH

Monitor File over Windows Share

Monitor Script Output over SSH

Monitor Script Output on Collection Agent

Monitor Remote Windows Events

Monitor Local Windows Events

Monitor using External Configuration

Receive over TCP/UDP

Receive over HTTP/HTTPS

Upload File

Unfortunately, solution is incomplete. I tried to calculate office's mood based on coffee stock, amount of fruit in fridge and current temperature. Below is the assumption I made.

Coffee stock range: 2 to 10  
Temperature: -5 to 45  
Fruit content in fridge: 0 to 16   
Office mood index: 1 to 5

Mood calculation logic is implemented in changeOfficeMood() method.

# Office Monitor Challenge

At **EDITED** data is our lifeblood — our systems need to have a clean and constant stream of it. To make sure of this, we monitor almost everything. We've decided to build a real-time dashboard to monitor an equally important part of our company — the happiness of our staff. You are in charge of building its frontend.

Time limit: **2 hours** Language: **JavaScript** Difficulty: **Easy for you, Vishal**

### The Challenge

You are given an archive (found [here](https://s3-eu-west-1.amazonaws.com/editd-challenges/js_dashboard_challenge.zip)) that contains three files: index.html, styles.css and reset.css. The interface for the dashboard is in index.html - note: there is no need to change the markup in order to complete the task.

The index.html also defines a set of possible events that the browser will receive from a server. The specifics of the server implementation are out of scope, so you should create a method that simulates these events being periodically received.

You need to **listen** and **react** to the fired events by changing the interface appropriately.

You also need to remember to reflect the office's mood depending on certain conditions. Our staff are known to get unhappy when we are low on coffee or there's nothing in the fridge. They also like warm climates - weird, we know, for people living in the UK.

It is up to you to define the office conditions that result in each mood.

You can use these values for your initial state:

Coffee: 6

Temperature: 22.5

Fridge: 8

### Recap

1. Build a method that simulates events coming from the server.
2. Build a mechanism to listen, react to those events and update the dashboard widgets appropriately.
3. If you have time, you can try and come up with a better algorithm for changing the office mood or simulating the flow of events.
4. Provide a write-up describing your approach and how you might improve your code given more time.
5. Have fun!

### What we look for

1. Clean, nicely structured and maintainable code.
2. JavaScript best practices.
3. Thoughtful design. Think about structure - how does your code scale as the number of things we monitor increases?
4. Avoid using third-party libraries (angular/backbone/react/knockout etc.), **apart from jQuery and lodash, which are already provided**. Grab the chance to show off your skills, rather than demonstrating a framework's features.

### Time constraint: 2 hours

We're not looking for a totally complete solution, so don't worry if you don't finish everything in the time allowed - we want to see evidence of good practices! Document your code and illustrate your thought process with a small writeup in a README.md file so that we can see what you were thinking as you went along.

Please [submit your solution](https://edited.com/challenges/dda76b35bb124a8f9542671b5f657451/submit), without forgetting to add a writeup in a README.md file, at the end of your allotted time, but feel free to carry on if you're enjoying yourself.

Good luck warrior,   
— The **EDITED** Geeks