**Event Driven Framework for BBSSl**

***Introduction***

1. What is Event Driven Architecture?

"Event-driven architecture (EDA) is a software architecture pattern promoting the production, detection, consumption of, and reaction to events."  
  
Broadly speaking it is a very loosely coupled architecture where events are fired off by producers and consumed by consumers without either of them necessarily having to know about each other. This allows for new consumers to start processing events without the producer changing. It also allows for multiple consumers to process events without any changes on the producer side.

1. What is Event driven Framework?

Event driven frameworks (EDF) typically mean frameworks that in one way or another employ the actor model of programming. This could mean using Akka in Scala, it could mean simply using Erlang, or it could mean using a message bus such as ActiveMQ or RabbitMQ and architecting your software into small, message driven components.  
  
Again the benefit is loose coupling. Actors need to be able to send each other messages (instead of blindly broadcasting events in EDA) but that is it. This pattern is very useful for modeling complex concurrent behavior in a way that is thread safe and can greatly simplify programming distributed systems. It also handles load spikes well due to the queues between actors and allows for actors to be scaled independently of each other which helps manage bottlenecks.

1. Why are we using Event Driven Framework?

Reason behind using Event Driven Framework is the benefit of loose coupling. Each event trigger will take care of its own functions. Polling for an event to be occurred is an important drawback which has been removed by this framework.

1. What are we going to make?

Our intention is to make a web based dashboard which shows Data in a Graphical Manner also which the user can customize to view a graph of his own choice.

The dashboard will contain a Login Form to take the user to a page restricted to a few features whereas admin can overview all the features.

The Dashboard will contain Progress Bars too showing the Remaining part to achieve the target. Once target completes This Bar Appends to the last Of the Task Queue.

It will also contain a download page if the user wants to download a full review of his/her Progress in different formats such as PDF, XLS, CSV etc.

***Technologies to be used***

Front End:-

1. Boot-strap CSS:-

Boot-strap is the most popular HTML, CSS, and JavaScript framework for developing responsive, mobile-first Websites i.e. the website need not be customized again for mobile or tablet specific views.

It is used in the dashboard to setup alignments colours and responsiveness.

1. JQuery:-

It is a fast, small, and feature-rich JavaScript Library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an API which works across many browsers.

Also JavaScript will be used to Render Animations and other responsive attributes.

Back End:-

1. D3.js:-

It is a JavaScript library for maintaining documents based on data. D3 helps us to design data into pictorial forms using HTML, SVG and CSS. D3 provides a data driven approach to DOM.

We will be using D3.js framework for designing and customizing our dashboard heavily. Also in the front end Boot-strap CSS will help us to make the dashboard more attractive.

1. Node.js:-

Node.js is an Event-driven asynchronous framework. Asynchronous means it can handle multiple requests at the same time unlike PHP and Event-driven means it will fire call back as the event occurs. It is always in a circular loop to execute its functions.

We will be using Node.js as the main framework on top of which everything on our dashboard will actually be built i.e. the fetching of data from the database, query processing from dGraph etc.

1. Apache Storm:-

It is a free open source distributed real-time computation system. It has become a standard for distributed realtime processing system that allows us to process large amount of data similar to Hadoop.

1. Apache Kafka:-

It is publish-subscribe messaging rethought as a distributed commit log. A single Kafka broker can handle hundreds of megabytes of reads and writes per second from thousands of clients. Kafka is designed to allow a single cluster to serve as the central data backbone for a large organization.

1. dGraph:-

It is a graph database. It provides high level scale and throughput, with low latency to be serving real-time queries over Terabytes of structured data. Dgraph supports GraphQL-like query syntax, and responds in JSON and Protocol Buffers over GRPC.