Bachelors Technology Project Report

Canvoard

Faculty Mentor: Dr. Sumit Kalra

Students: Nikhil Srivastava(B16CS020)
& Saksham Banga (B16CS042)

Semester 6

Contents

Acknowledgements **3**

1. Introduction **5**

Problem Statement 7

Motivation 7

2. Literature Analysis on Previous Works **8**

What are WebSockets? 9

What are WebSockets good for? 10

What is AJAX? 12

What is Long Polling? 13

Mutex 15

Semaphore 16

3. Summary of Contributions **18**

Canvoard's 3 Main Pillars 19

Part 1: Dynamizing Static Canvas of HTML5 20

Part 2: Impactful and efficient Serialization 21

Part 3: Maintaining Synchronicity 23

5. Results **25**

Salient Features with Appropriate Screenshots 26

Draggable interface for Rotating and Scaling Objects including paths, images and shapes 26

Interacting with Images 27

6. Conclusion and Future Prospects **28**

7. References **30**

Acknowledgements

We would like to express our deepest gratitude to our supervisor, Prof. Sumit Kalra, for his innovative ideas, excellent guidance, constant encouragement, invaluable suggestions, and support during our BTP. We would like to thank him for being a wonderful mentor who always inspired and guided us to the right path. We would also express our gratitude to Prof. Santanu Choudhary and Prof. Rajlaxmi Chouhan for giving some extra use cases.

We would also like to express our special thanks to Dhruv Sharma for his advice on getting started with the project and resolving the initial difficulties. and help in realizing the thesis for this project.

In addition, we are grateful to the CSE Department, IIT Jodhpur for providing
us with all the facilities and exposing us to a conducive environment to choose and complete this project. The Work experience has been enjoyable, fruitful and a memorable one.

Last, but not the least, we express gratitude to our families. Without their support we would never have been able to venture into all the activities that we did. We will always be grateful to our family for their constant prayers and immense support when we needed them the most.

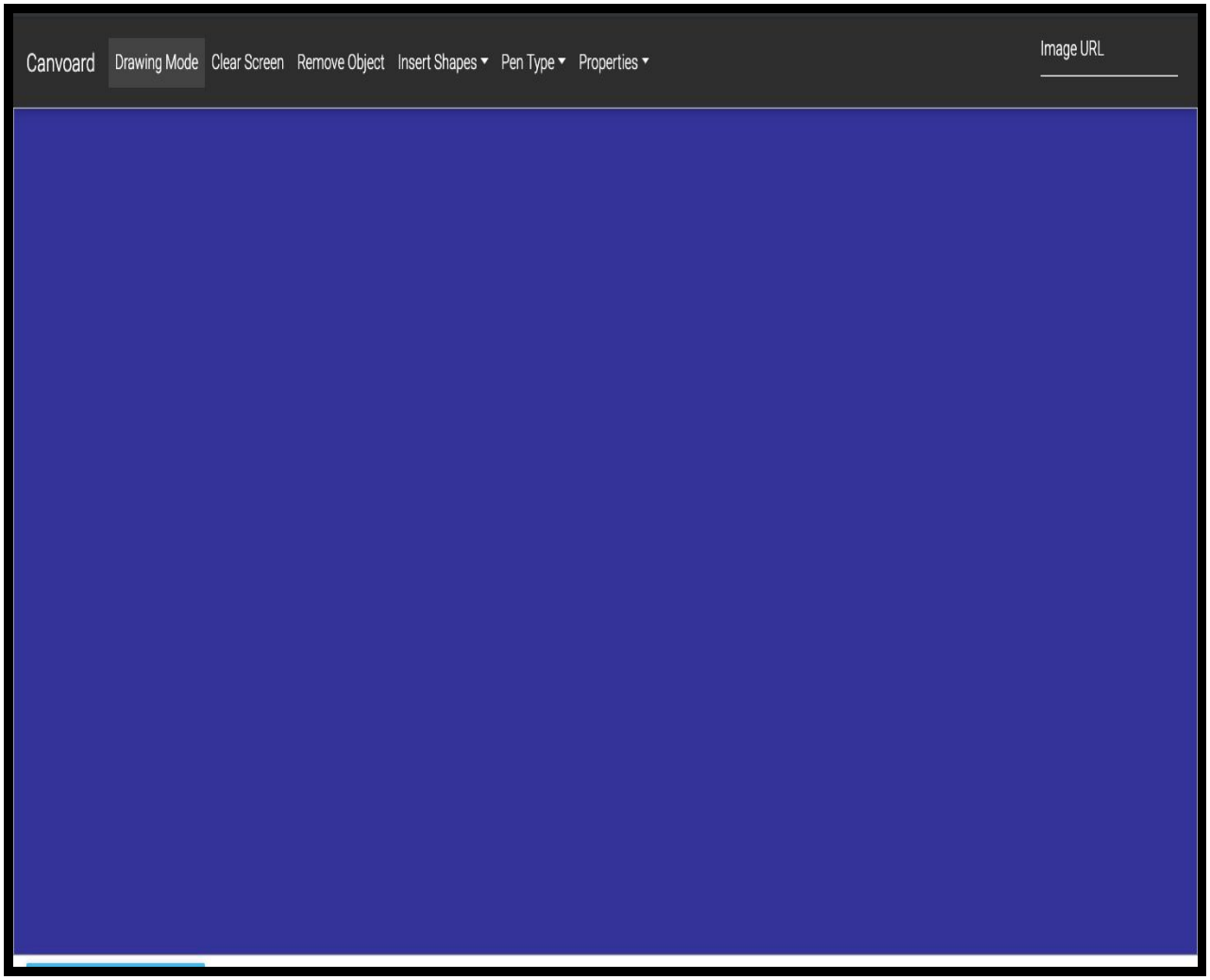
Nikhil Srivastava
B16CS020

Saksham Banga
B16CS042

This Page is Intentionally Left Blank

1. Introduction

During lectures being taken for a massive classroom gathering of more than hundred students, a collaborative canvas resembling whiteboard would serve as an efficient tool for ease of understanding course content.



Paint your heart out

Problem Statement

This project of ours, draws inspiration from the classroom lectures during the course, EE-121 by Prof. Rajlaxmi Chouhan. Often during the lecture, students had doubts which lied in the intricacies of the figures of circuits drawn by the professors. In small classes, interaction was definitely not a problem, but in large classes, in case someone sitting behind has a doubt, needs to either be vocal enough to raise his concern, and be effective in conveying the area of obscurity. This is precisely the problem we are solving.

Motivation

Having done an informative and exciting project on analyzing the structural properties of documents based on their geometric and layout representation during segmentation tasks, in the last semester under Prof. Gaurav Harit, the motto this time was to take a step ahead, and take a deep dive inside the inherent design of a networking based collaborative canvas whiteboard, which can find its application in numerous working scenarios.

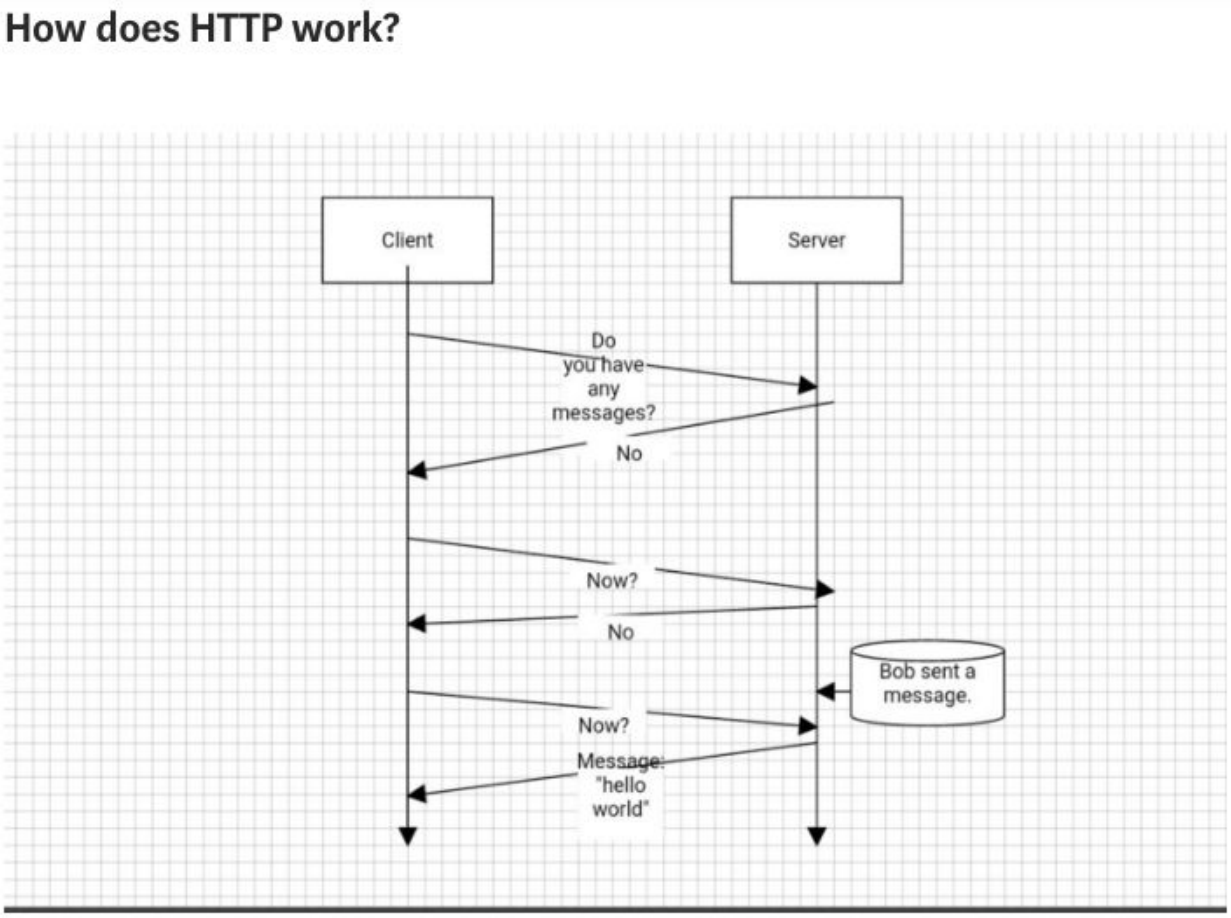
To achieve this task we collaborated with Dr. Sumit Kalra and can't thank him enough for his unconditional help and support throughout the span of this project.

The curiosity was to learn the way document information is vectorised and displayed onto the canvas board available at our disposal. And this project has provided us with a thrust to go about the same in a very productive and pipelined fashion.

2. Literature Analysis on Previous Works

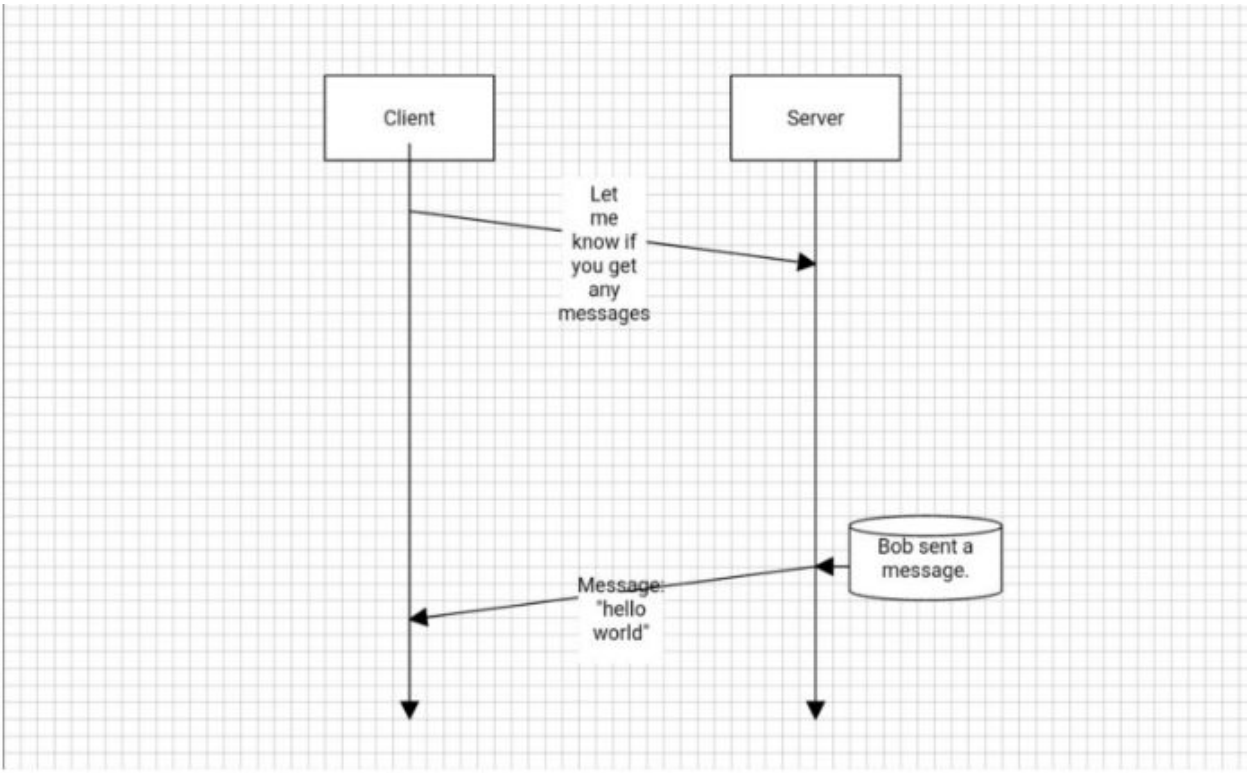
What are WebSockets?

WebSockets" is an advanced technology that allows real-time interactive communication between the client browser and a server. It uses a completely different protocol that allows bidirectional data flow, making it unique against HTTP.



How HTTP works

A request is needed to respond; you to constantly ask the server if there are new messages in order to receive them.

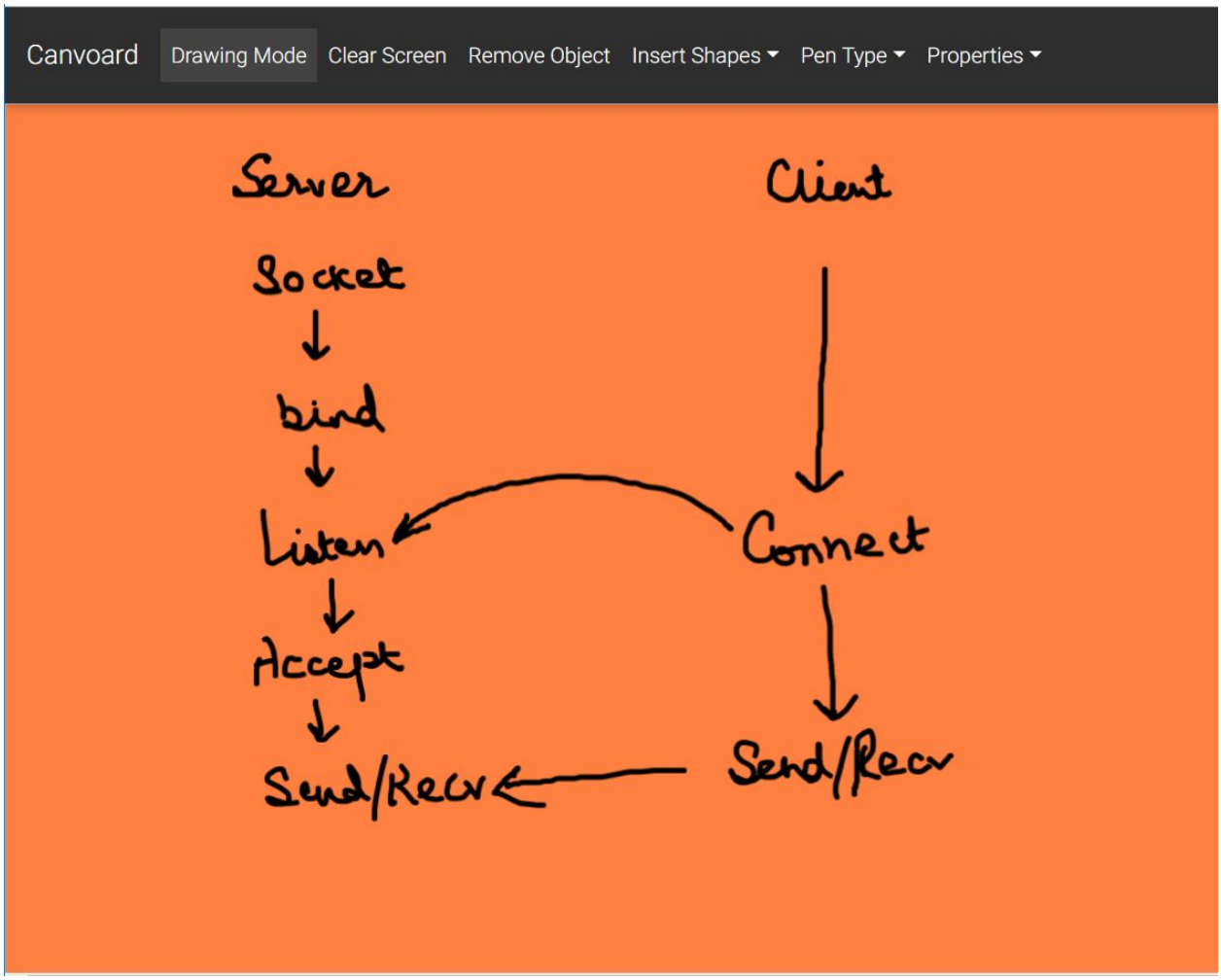


How websocket works

Web Sockets on the other hand don't need us to send a request in order to respond. They allow bidirectional data flow so we just have to listen for any data. We can just listen to the server and it will send us a message when it's available.

What are WebSockets good for?

- Real-time applications
- Chat apps
- IoT (internet of things)
- Online multiplayer games



Outline of establishment of Socket Connection

When we should avoid them?

The answer is almost never. The only reason to think of against using Web Socket is the current compatibility.In some cases, we will not even need web sockets. We don't think we will need a real-time app if we are building a simple CMS, unless we want some kind of a specific real-time feature. For a RESTful-API we shouldn't be using Web Sockets as HTTP GET, POST, DELETE, PUT and many other requests are awesome for that.

What is AJAX?

AJAX stands for Asynchronous JavaScript and XML. AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Java Script.

- Ajax uses XHTML for content, CSS for presentation, along with Document Object Model and JavaScript for dynamic content display.
- Conventional web applications transmit information to and from the server using synchronous requests. It means when a form is filled, hitting submit,we get directed to a new page with new information from the server.
- With AJAX, when submitted, JavaScript will make a request to the server, interpret the results, and update the current screen. In the purest sense, the user would never know that anything was even transmitted to the server.
- XML is commonly used as the format for receiving server data, although any format, including plain text, can be used.
- AJAX is a web browser technology independent of web server software.
- A user can continue to use the application while the client program requests information from the server in the background.
- Intuitive and natural user interaction. Clicking is not required, mouse movement is a sufficient event trigger.
- Data-driven as opposed to page-driven.

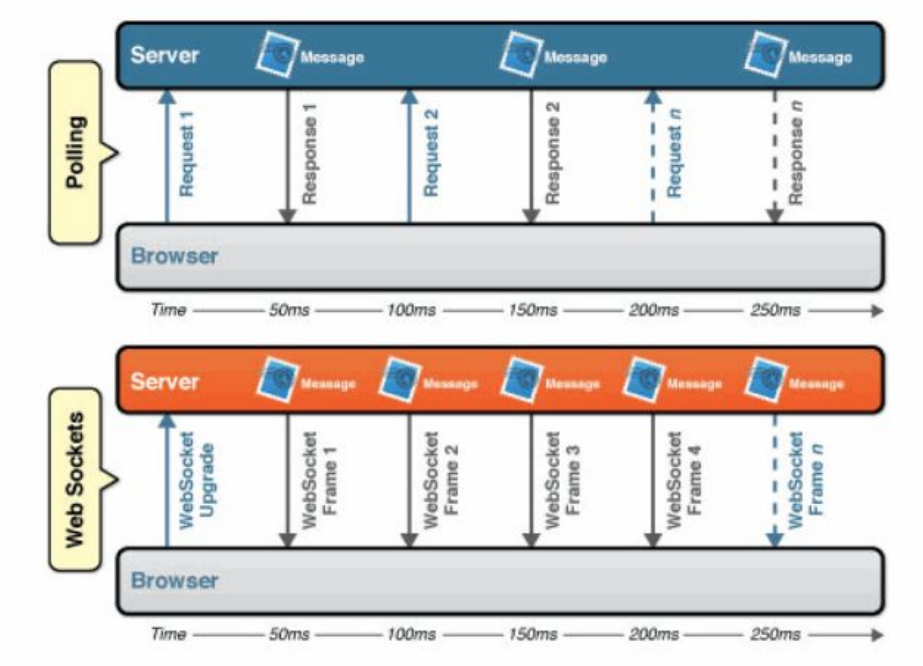
What is Long Polling?

It is a technology where the client requests information from the server without expecting an immediate response or basically involves making an HTTP request to a server and then holding the connection open to allow the server to respond later. Using long polling the server allows approximately 6 parallel connections from the browser.

Load balancing in this is easy compared to other ways. Long polling is the oldest ways and hence is supported on all web browsers. Though due to the fewer updates in this it does not provide re-connection handling. Long polling is a lot more intensive or heavy on the server, but more widely accepted for browsers.

Long polling addresses the weakness of traditional polling methods by asking the server for new information on a certain interval, but keeping the connection open if the server has nothing new for us. This technique dramatically decreases latency and network traffic, which means it efficiently disguises itself as a server-push technique.

HTML5 Web Sockets thus provides an enormous step forward in the scalability of the real-time web. HTML5 Web Sockets can provide a 500:1 or—depending on the size of the HTTP headers—even a 1000:1 reduction in unnecessary HTTP header traffic and 3:1 reduction in latency. That is not just an incremental improvement; that is a revolutionary jump.



Contrast between network latency

Brief Contrast amongst the three:

METHOD	Client to Server	Server to Client	Bidirectional
Ajax(Short polling)	**Works well**	----------	No Server Client Communication channel
Long polling	----------	**Works well when infrequent updates from server exist**	No Client Server communication channel
Web Sockets	**----------**	**----------**	**Works well**

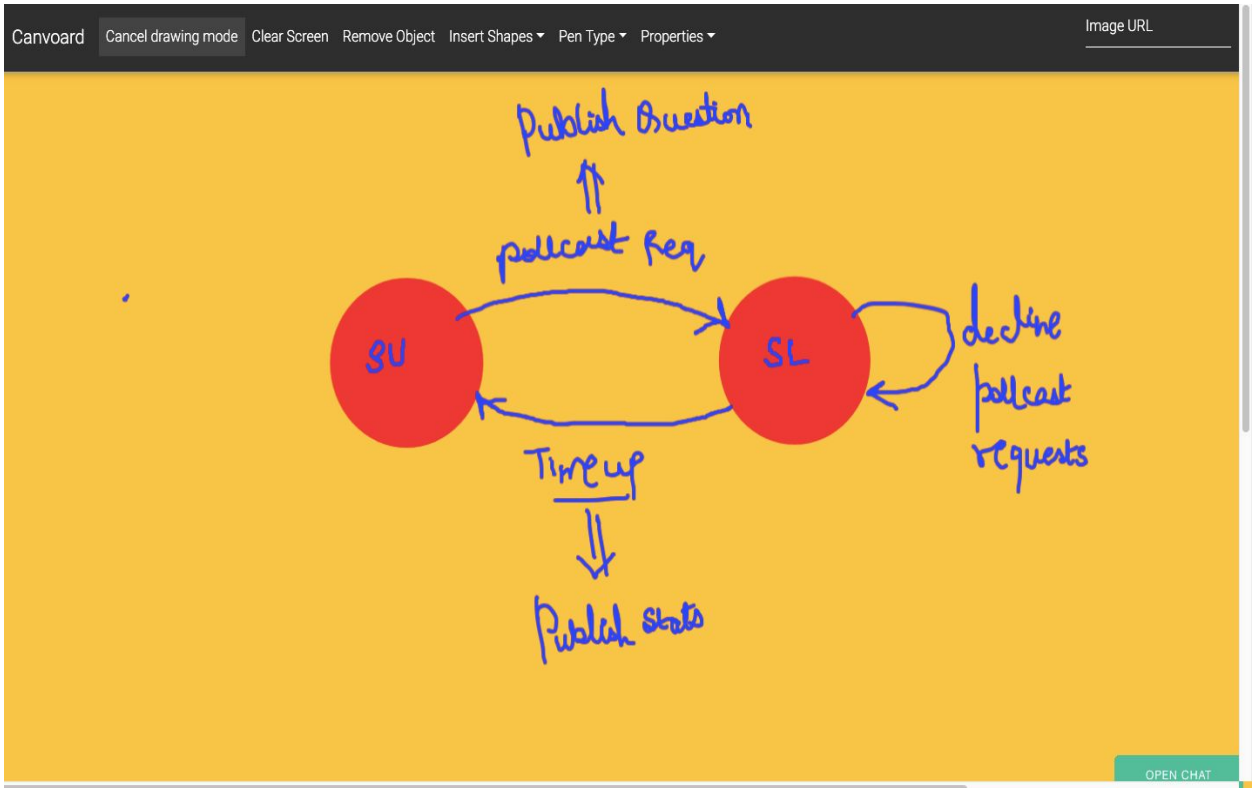
Mutex

Mutex is a mutual exclusion object that synchronizes access to a resource. It is created with a unique name at the start of a program. The Mutex is a locking mechanism that makes sure only one thread can acquire the Mutex at a time and enter the critical section. This thread only releases the Mutex when it exits the critical section.

This is shown with the help of the following example:

```
wait (mutex);

    .....

        Critical Section

    .....

signal (mutex);
```

A Mutex is different than a semaphore as it is a locking mechanism while a semaphore is a signalling mechanism. A binary semaphore can be used as a Mutex but a Mutex can never be used as a semaphore.



Finite State Automaton for Polling Feature

Semaphore

A semaphore is a signalling mechanism and a thread that is waiting on a semaphore can be signaled by another thread. This is different than a mutex as the mutex can be signaled only by the thread that called the wait function.

A semaphore uses two atomic operations, wait and signal for process synchronization.

The wait operation decrements the value of its argument S, if it is positive. If S is negative or zero, then no operation is performed.

wait(S)

```
{
   while (S<=0);
    S--;
}
```

The signal operation increments the value of its argument S.

```
signal(S)
{
   S++;
}
```

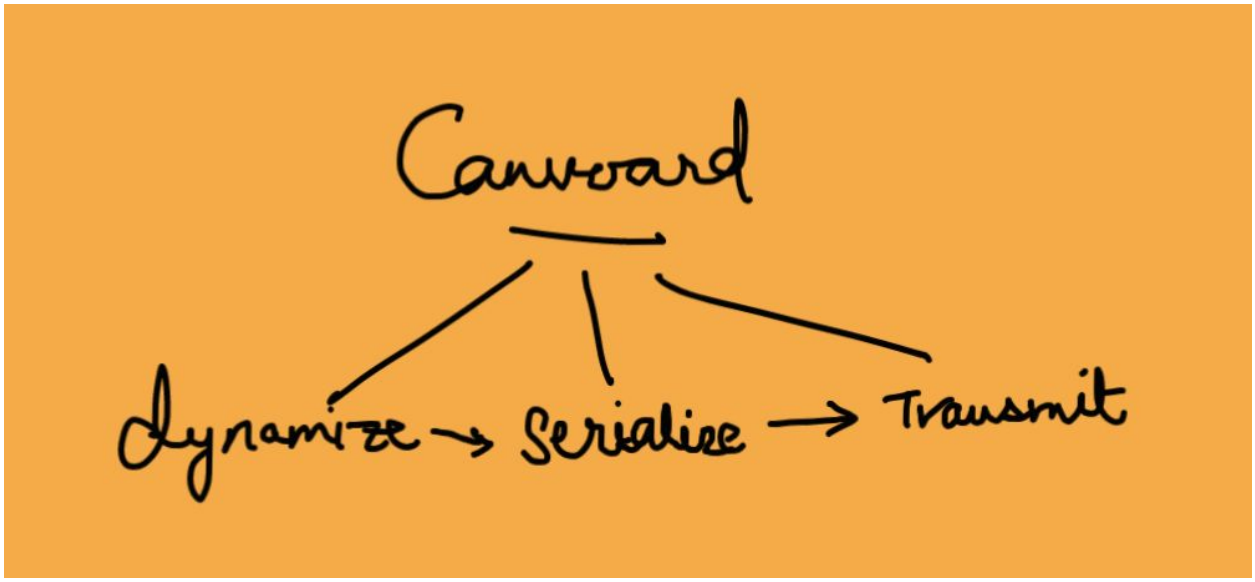
There are mainly two types of semaphores i.e. counting semaphores and binary semaphores.

Counting Semaphores are integer value semaphores and have an unrestricted value domain. These semaphores are used to coordinate the resource access, where the semaphore count is the number of available resources.

The binary semaphores are like counting semaphores but their value is restricted to 0 and 1. The wait operation only works when the semaphore is 1 and the signal operation succeeds when the semaphore is 0.

3. Detailed Methodology of Contributions

Canvoard's 3 Main Pillars

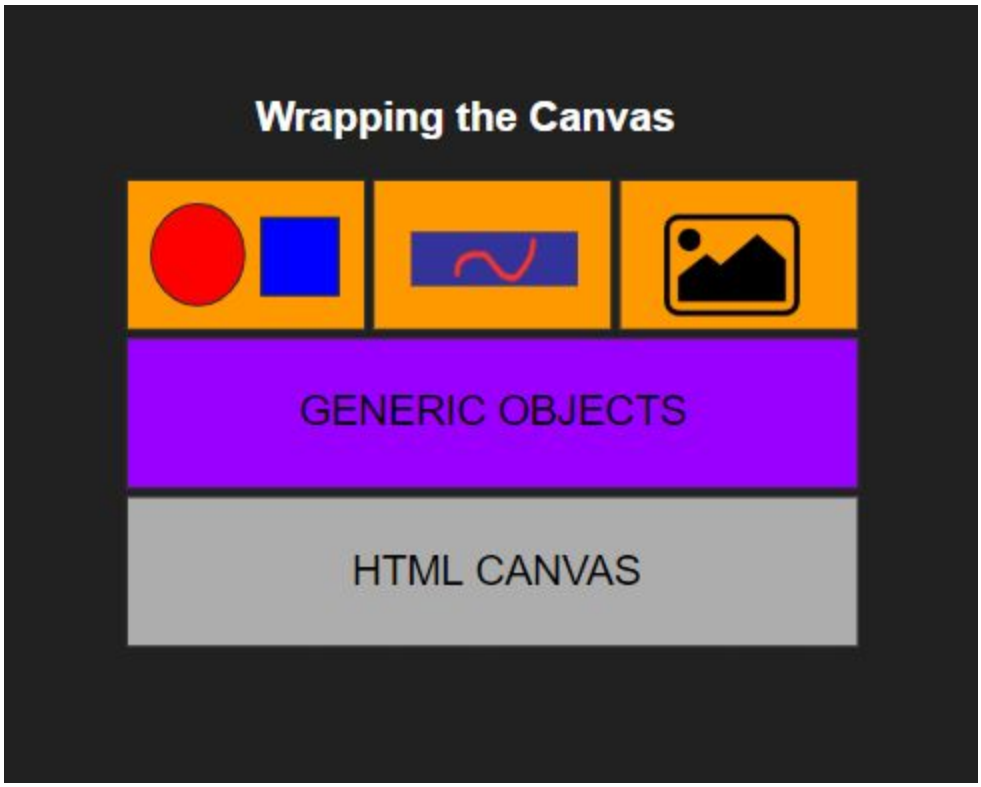


The software revolves around 3 main pillars:

1. **Dynamize** the extremely rigid and static Canvas element by the process of continuous and fast re-rendering
2. Improving **Serialization** Efficacy
3. Maintaining **Synchronicity**

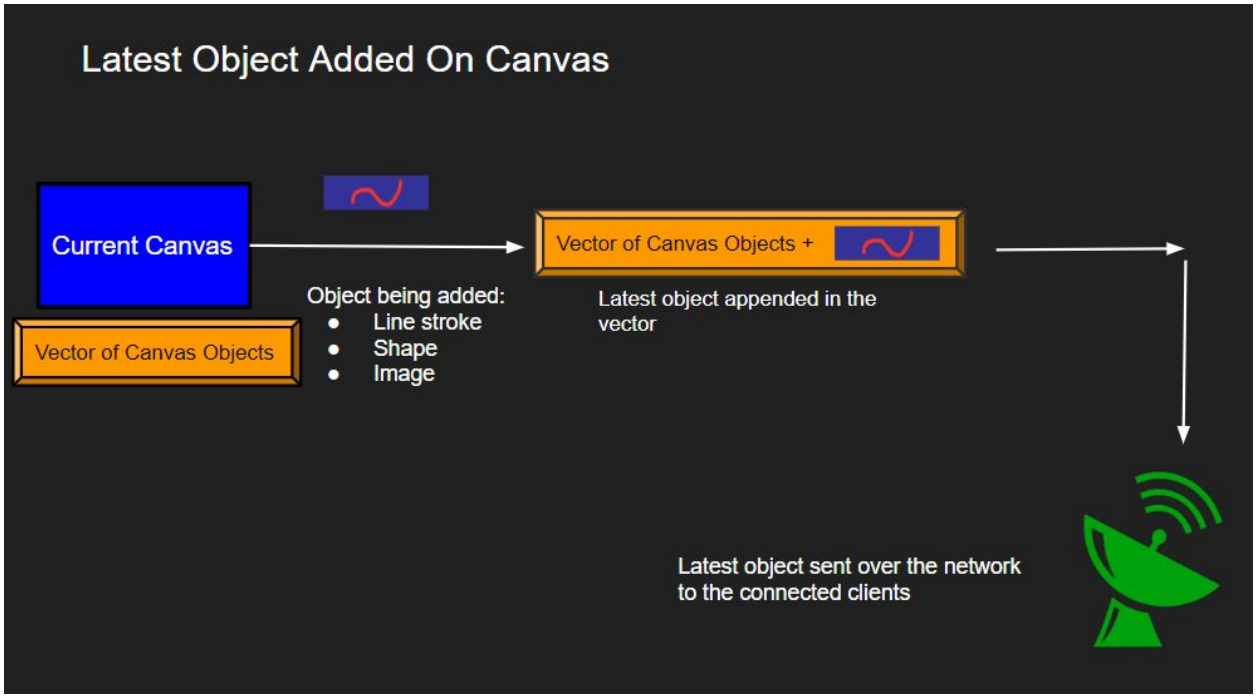
Part 1: Dynamizing Static Canvas of HTML5

→ Currently canvas of HTML5 offers many features in order to conduct animations. However, it is pretty static and inorder to make any changes to existing canvas board and its entities, one would need to reconfigure the whole board once again and simulate code for the same.



→ We enable a robust way to do this. Wrapping the existing canvas objects helps us treat all the shapes and paths as different individual entities. This objectification and individualisation gives way in order to edit the scene and make the canvas dynamic.

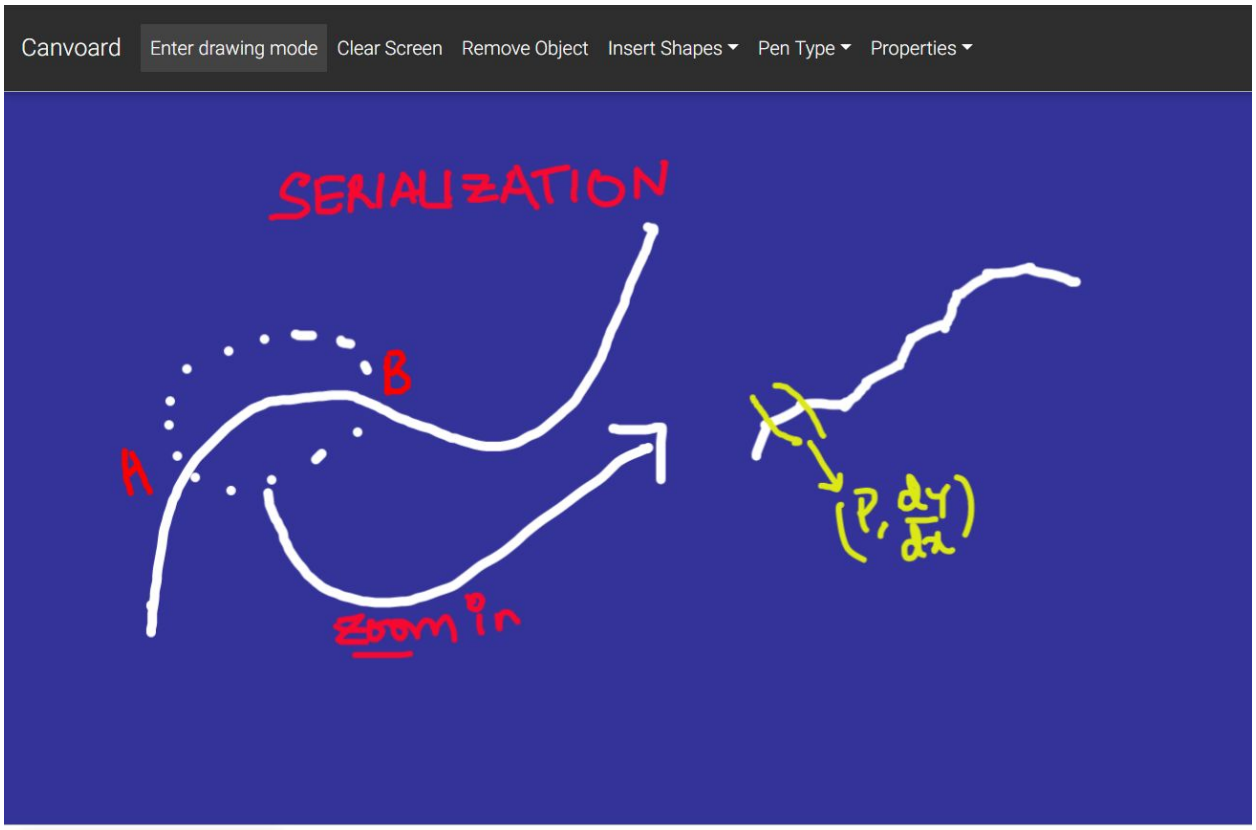
→ Once the objects are dynamized, the canvas can be re-rendered. This abstraction allows us to perform more with less code.



Part 2: Impactful and efficient Serialization

→ As we begin using the marker/mouse to put down the pen strokes on the canvas board, we initiate a serialize module as a part of our canvas backend.
→ In order to assist the user to systematically broadcast the state of the canvas board at all times, it becomes a need of the hour to store and send the drawn strokes to effectively decode the same and re-render the canvas for the other clients with the latest changes performed.
→ The prime functionality that is achieved by this, is to seamlessly allow the user to draw free hand drawings and textual writings on the face of the canvas.
→ As soon as the pen starts laying down the strokes on the canvas, they are dynamically sampled at a rate of 5 pixels, to fix the end points of a tiny little straight line.
→ For each of these straight lines, the starting point and the slope is stored to be then broadcasted over the network.

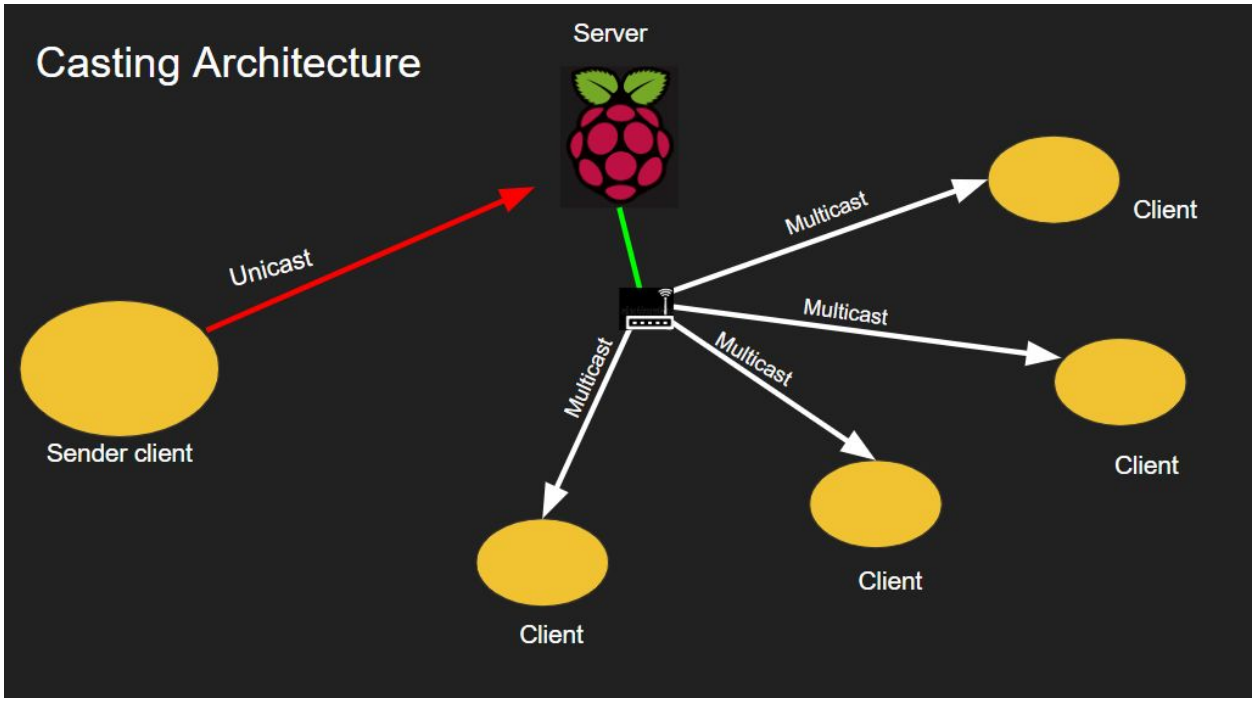
→ This sampling rate empirically works the best in order to capitulate the most efficient seamless rendering of the line strokes.



Part 3: Maintaining Synchronicity

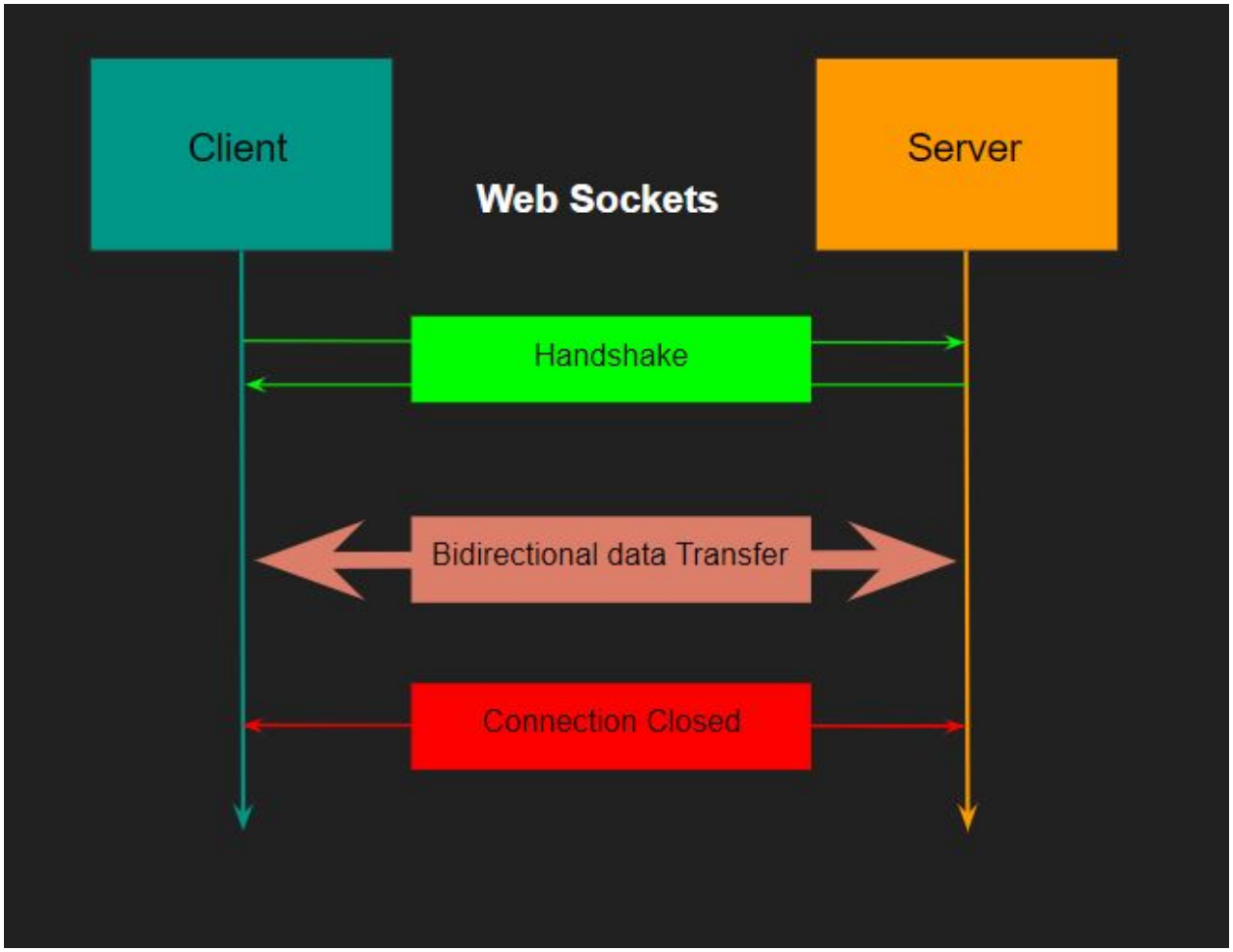
→ Being a real time system, it is necessary that we have almost negligible latency, or else the system would rendered useless. In order to do so, we improved the serialization in order to reduce the data being sent. However the process behind sending data is equally and rather more important. Thus efficient pipelining of data is the need of the hour.

→ The requirements are for all the devices to be inter-connected with the source of modification and the server must be able to connect with all the clients at all times. In order to satisfy this bidirectional need and with a moderately large strength in terms of the classroom(200+ in some cases), the P2P structure is somewhat eliminated and the best option is to use WebSockets in order to communicate.



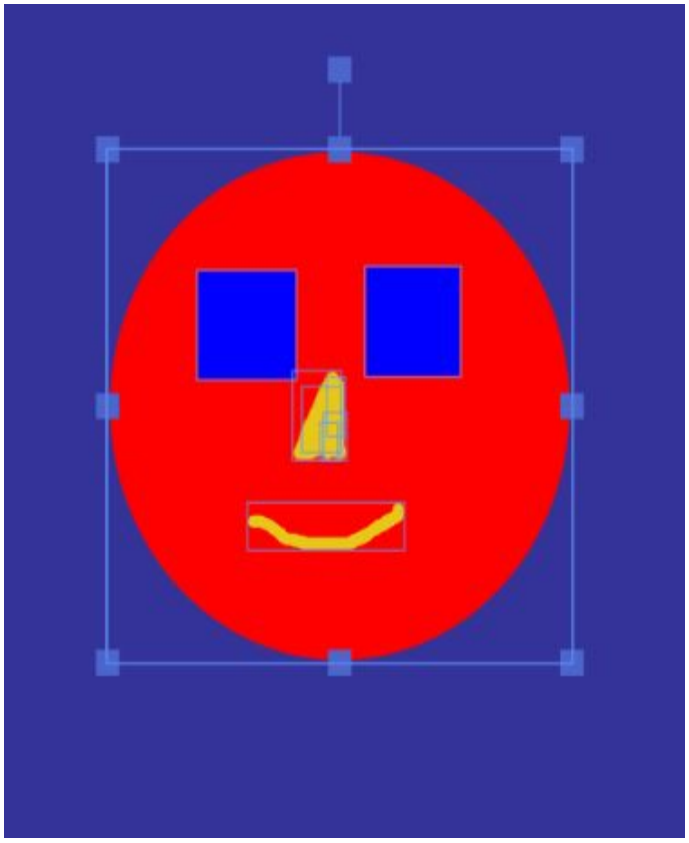
→ This is because AJAX(although asynchronous) requires connection to be set-up

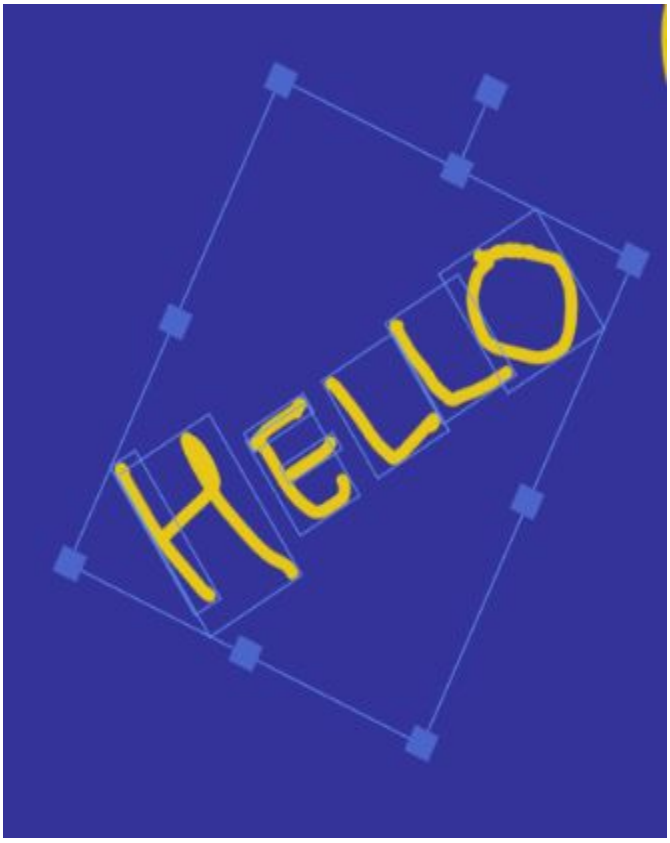
and breaks down after every message delivery. In an environment involving real time changes to the state of the board, having low latency with minimal bandwidth consumption is the universal aim. Thus methods involving a sampling in order to fetch requests(short polling as well as long polling are not recommended.



4. Results

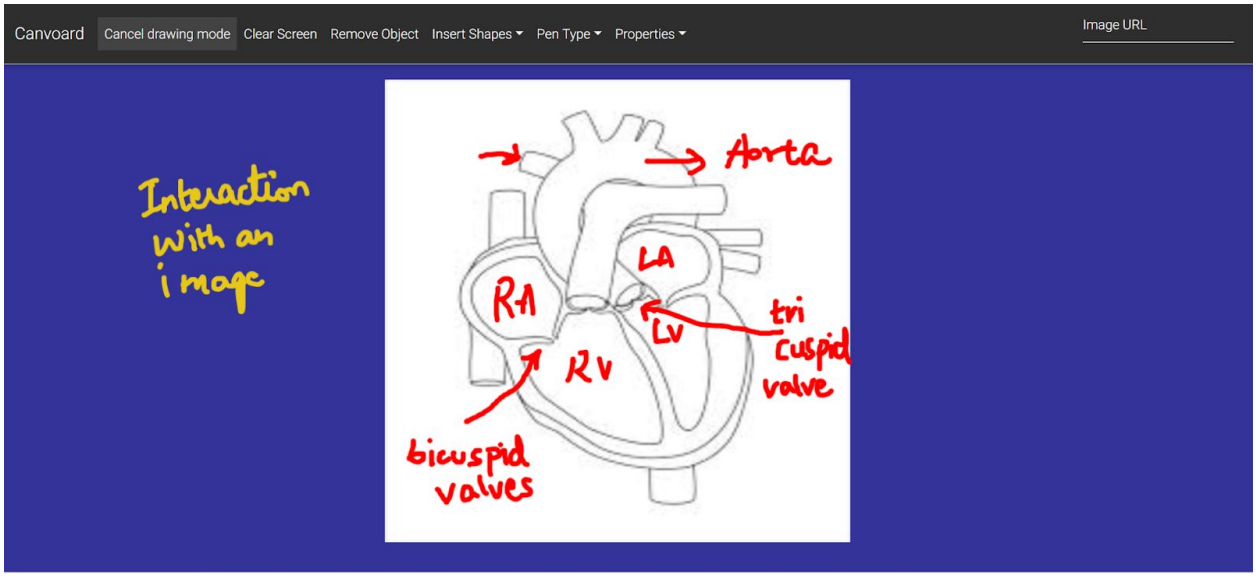
Salient Features with Appropriate Screenshots

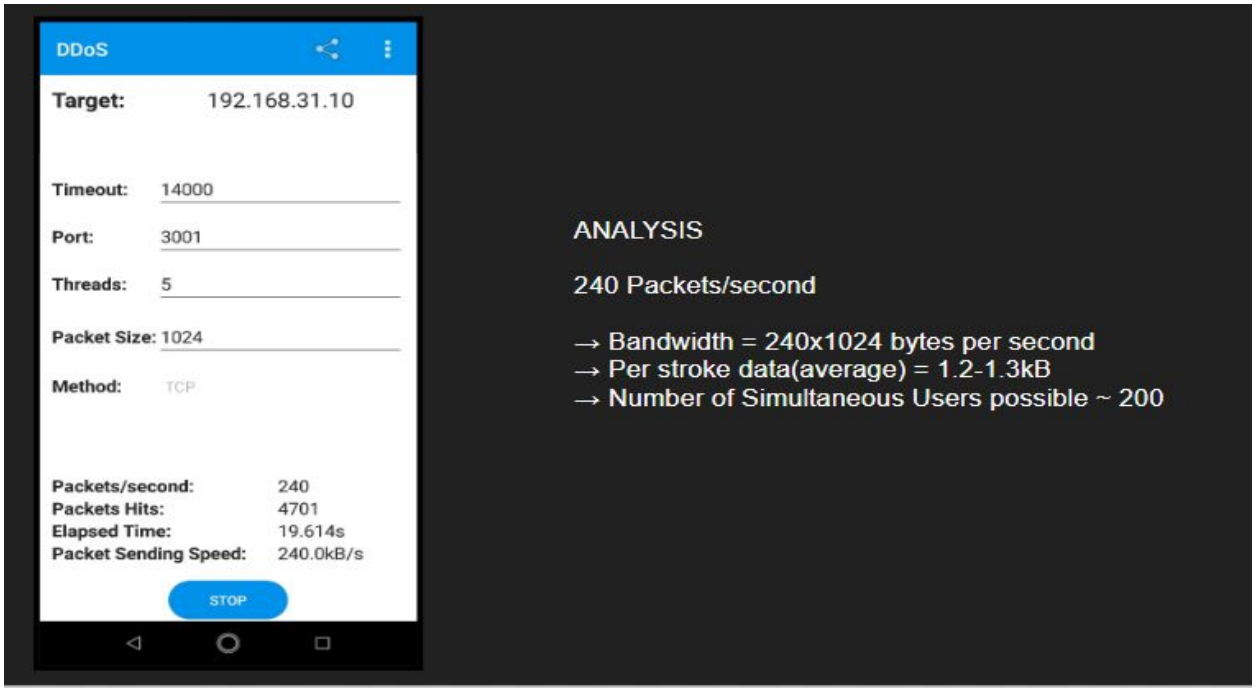




Draggable interface for Rotating and Scaling Objects including paths, images and shapes

Interacting with Images





Analysis of Bandwidth using DDoS

5. Conclusion and Future Prospects

What initially started out as a development project eventually caught our interests due to the peculiar use cases. This motivated us to scratch the surface harder and come up with the approach of developing more features. This approach negligibly affects the latency and is indeed state of the art. This is due to the simplicity of the structural representation of the problem statement and breaking it down into 3 parts which essentially follows from intuition.

Canvoard was tested for multiple devices but sky is the limit for its usecases. We plan to extend Canvoard and put it for public use very soon by following these 3 steps

1. Developing and testing across IIT-Jodhpur.
2. Commercialising the platform across other universities.
3. Extending a feature of the current Google Suite of products → Google Draw / Microsoft Paint Collab version.
4. Could serve as plugin for Education and conferneces via VoIP
5. Would serve as a collaborative notes taking software invoking P2P based learning in schools and institutes.

The factor of optimization and scalability is something in which the project can be very flexible. This can be attributed to the future of this project.

6. References

1. https://tools.ietf.org/html/rfc6455
2. https://www.techopedia.com/definition/16208/socket
3. https://nodejs.org/en/docs/
4. https://expressjs.com/
5. https://www.postgresql.org/docs/