



“Address stability issues within Thimble” and “Thimble and Remote Mentorship”

(Organisation-Mozilla)

Google Summer of Code 2017

My Details:

- Name: Sudhanshu Sambharya
- Email: sambharya88@gmail.com
- GitHub: <https://github.com/sudwebd>
- IRC nick: sud_code
- Country of Residence: India
- Timezone: GMT/UTC + 5:30h
- Primary language: English

About Me:

I am Sudhanshu Sambharya ,BTech undergraduate student at IIT Roorkee. My major region of work is web design and development (both frontend and backend). My skills include:

- Good hand in Javascript and Javascript libraries(eg: JQuery).
- Fluency in HTML and CSS
- PHP
- Good knowledge in C++
- Bootstrap and Google Material Design
- Node.js
- GitHub
- MySQL database



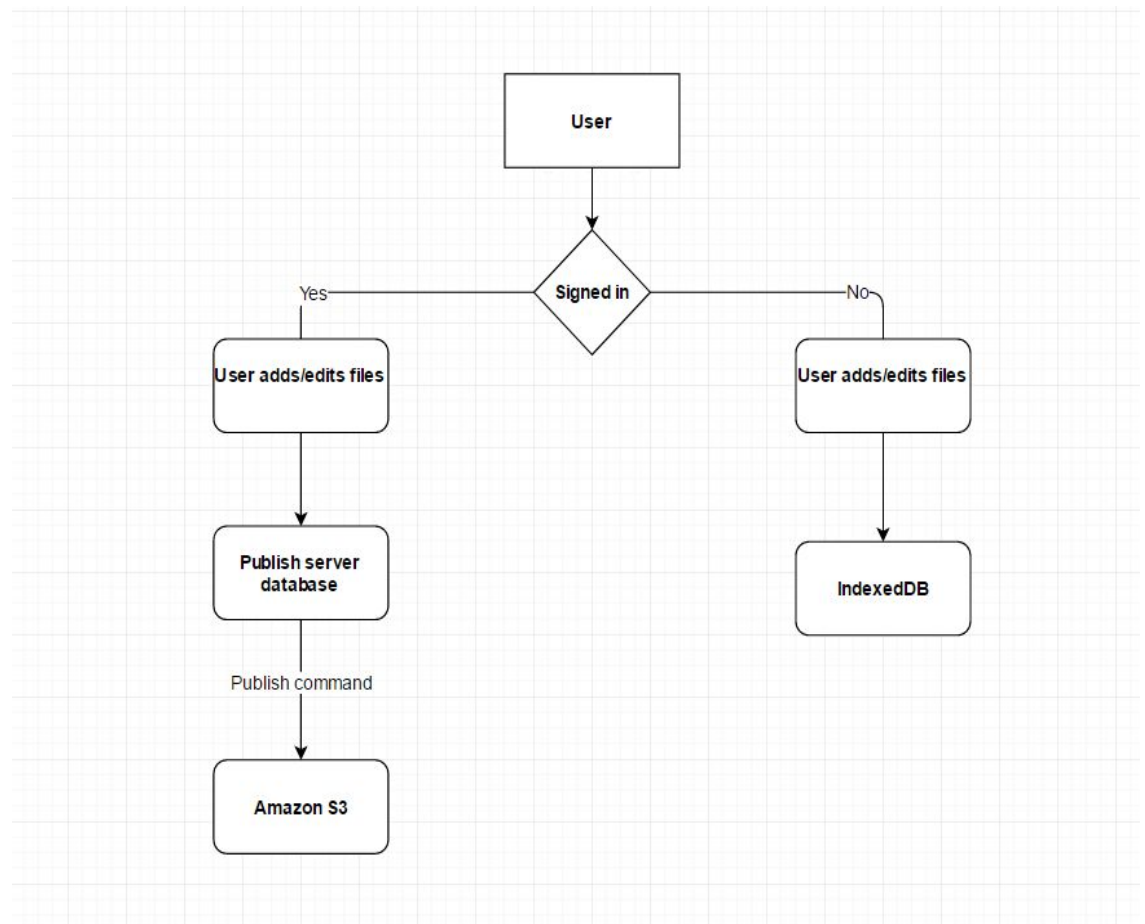
Project Description:

The aim of the Thimble based projects is to improve the functionality and verticality of thimble. Presently Thimble has some bugs and issues, throughout the project work we will be fixing the issues and adding additional functionality to Thimble.

- For the project “Address stability issues within Thimble”, our goal is to overcome the issues with the database (where Thimble stores the files and updates before they being published). The way we are going to achieve this is by storing the files (before publishing) directly on AWS rather than relying on the publish server database, this will help in enhancing the performance by overcoming issues like transaction lags,etc . Thimble locally implements the SQL databases for storage. The database management and testing code is present in the ‘publish.webmaker.org’ directory.

```
Thimble.mozilla.org
|_____services
|_____publish.webmaker.org
```


For writing the files directly to Amazon Simple Storage Service some major alterations in the codes have to be performed.



The above diagram shows how data persists in Thimble presently.

What we want to accomplish is writing directly to Amazon S3 rather than saving the files first in publish server's database.

- The aim of project “Thimble and Remote Mentorship” is to take the learning and development process to the next level by providing group development functionality wherein people can work on a project together in real-time ,this will be implemented with the help of Mozilla’s JavaScript library Together.js which provides with real-time development functionality and tools.



For the synchronization of states across various clients we will be using WebRTC. WebRTC is used in various apps like WhatsApp, Facebook Messenger, appear.in and platforms such as TokBox.

WebRTC implements three APIs:

- `MediaStream` (aka `getUserMedia`): get access to data streams, such as from the user's camera and microphone.
- `RTCPeerConnection`: audio or video calling, with facilities for encryption and bandwidth management.
- `RTCDataChannel`: peer-to-peer communication of generic data.

For a file transfer using WebRTC, the file is split into chunks which are then transferred via the datachannel. The datachannel is reliable and ordered by default which is well-suited for file-transfers. Send and receive progress is monitored using HTML5 progress elements. At the receiver, the file is reassembled using the Blob API and made available for download. For a more real scenario applications require a file transfer protocol to send metadata about the file (such as the filename, type, size, last modification date, hash, ...). This information can be conveyed either via the signaling channel or in-band (Signaling is a process used in WebRTC to detect peers. Signaling is used to exchange session control messages known as SDP; network configurations as ICE candidates; and media capabilities using same session control messages. Signaling can be done either using copy/paste mechanism; or using a gateway like WebSocket, Socket.io, XMPP or session initiation protocol (SIP). You can use the easiest mechanism as well i.e. POST/GET data from a database using XHR). A very simple way to achieve this is by assuming knowledge of the file size at the receiver and closes both the datachannel and the peerconnection when the correct amount of bytes has been received.

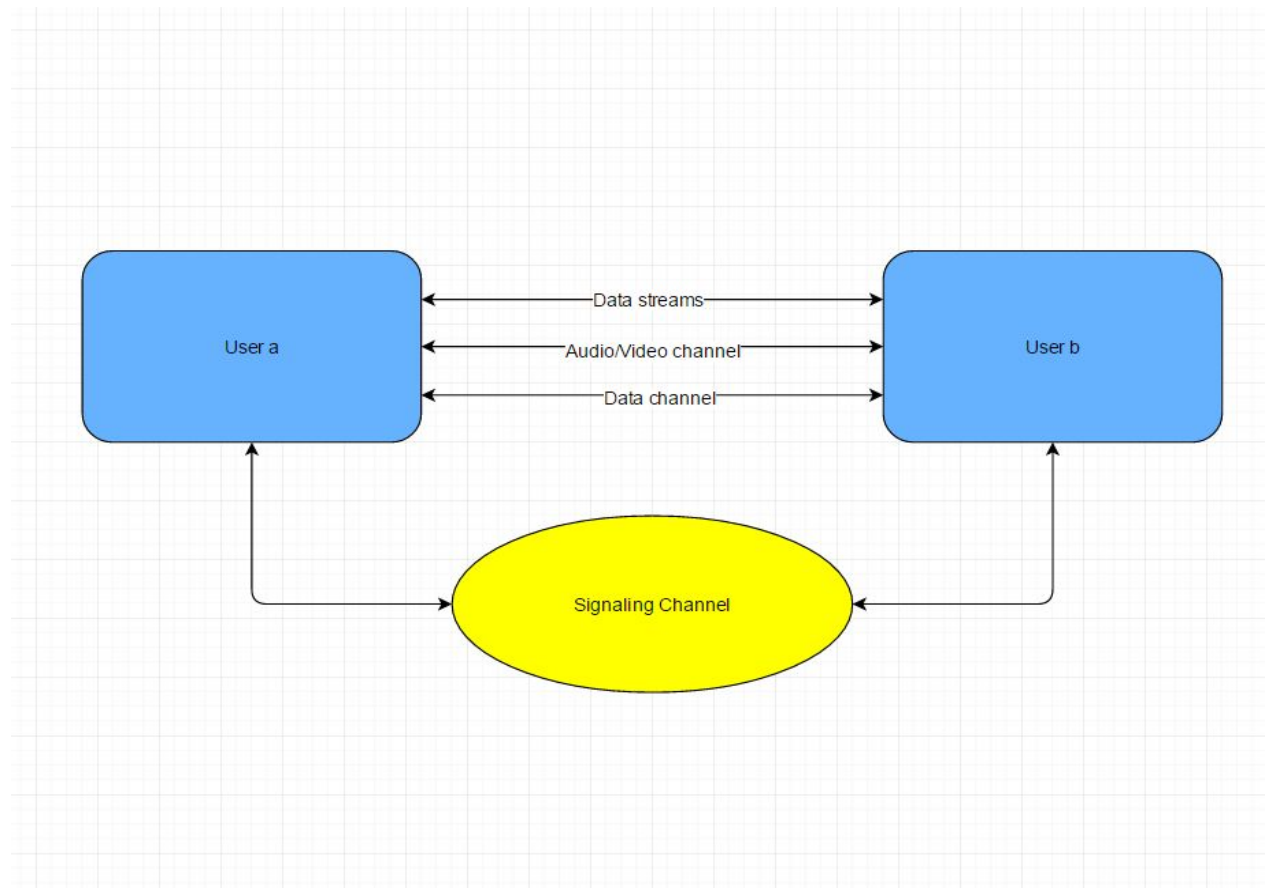
Signaling is not implemented by WebRTC, therefore WebRTC has to be coded for signaling, assuming we have some signaling mechanism `SignalingChannel`, the following code summarises the signaling process:

```

1  var signalingChannel = new SignalingChannel();
2  var configuration = {
3    'iceServers': [{
4      'url': 'stun:stun.example.org'
5    }]
6  };
7  var pc;
8  // call start() to initiate
9  function start() {
10     pc = new RTCPeerConnection(configuration);
11     // send any ice candidates to the other peer
12     pc.onicecandidate = function (evt) {
13       if (evt.candidate)
14         signalingChannel.send(JSON.stringify({
15           'candidate': evt.candidate
16         }));
17     };
18     // Let the 'negotiationneeded' event trigger offer generation
19     pc.onnegotiationneeded = function () {
20       pc.createOffer(localDescCreated, logError);
21     }
22     // once remote stream arrives, show it in the remote video element
23     pc.onaddstream = function (evt) {
24       remoteView.src = URL.createObjectURL(evt.stream);
25     };
26     // get a local stream, show it in a self-view and add it to be sent
27     navigator.getUserMedia({
28       'audio': true,
29       'video': true
30     }, function (stream) {
31       selfView.src = URL.createObjectURL(stream);
32       pc.addStream(stream);
33     }, logError);
34   }
35
36   function localDescCreated(desc) {
37     pc.setLocalDescription(desc, function () {
38       signalingChannel.send(JSON.stringify({
39         'sdp': pc.localDescription
40       }));
41     }, logError);
42   }
43
44   signalingChannel.onmessage = function (evt) {
45     if (!pc)
46       start();
47
48     var message = JSON.parse(evt.data);
49     if (message.sdp)
50       pc.setRemoteDescription(new RTCSessionDescription(message.sdp), function () {
51         // if we received an offer, we need to answer
52         if (pc.remoteDescription.type == 'offer')
53           pc.createAnswer(localDescCreated, logError);
54       }, logError);
55     else
56       pc.addIceCandidate(new RTCIceCandidate(message.candidate));
57   };
58
59   function logError(error) {
60     log(error.name + ': ' + error.message);
61   }
62

```

A signaling service is required to initiate a WebRTC session. However, once a connection has been established between two peers, RTCDataChannel could, in theory, take over as the signaling channel.




The above project description is a glimpse of the task to be done to achieve the project goals.

These two projects will greatly enhance the experience of the users and the additional tools that we add to Thimble during the project will provide the users with a great learning and development environment.

What is required to succeed at the project ?

In order to succeed at the project work one has to have a good hand in html, css, JavaScript, JavaScript libraries and other web development languages and libraries. I have



been working with web development for a long time and possibly I can go on with the development tasks that I have to face during the project development work.

For the first project “Address stability issues within Thimble” pre knowledge of AWS and database is necessary. Also knowledge of IndexedDB is required. In addition to these, one should know how to tackle loading speed issues and various issues with files in the databases like missing files, security issues, etc.

The focus for the project “Thimble and Remote Mentorship” will be on Together.js and Synchronisation of states (using WebRTC) in all the Thimble editors that wish to work together. Together.js is a JavaScript library which provides real-time development tools for doing online group development /project tasks. I also have to fix issues with Together.js to enhance its functionality for providing a better experience to the users. Also knowledge of using WebRTC for synchronization of the remote clients is a must.

What I hope to learn from this project?

I hope to learn a lot from this project like; this will help improve my knowledge on working with databases, and how databases are managed and maintained in big projects like Thimble, also I will learn a lot about the Amazon Web Services cloud, and the plus points that it has over the conventional database systems. The project will also help me learn how to reduce the lag in synchronisation of states on different remote systems. On a whole the project will help me enhance my development and coding skills. I will also learn how big organisations like Mozilla manage their work.

Work I have done till now:

I have discovered bugs: Responsiveness issues in the editor (issue #1894 on GitHub), UI issues and worked on the solutions for these issues.

I also suggested UI improvements for better user experience in a multi-user environment.

I studied the code for both Thimble and Brackets and also Together.js.



Deliverables:

- Thimble with functionality to publish directly on AWS (increased speed and reduced issues).
- Thimble providing multi-user support with tools capable of real-time group Development across remote Thimble editors.
- Together.js with better functionality and reduced bugs.

Schedule/work timeline:

- May 4 – May 29: Studying the code and fixing bugs
- May 30 – June 10: Working with the database and making Thimble save files directly to AWS
- June 11 – June 25: Fixing issues/bugs in Together.js making it functional for integrating with Thimble
- June 26 – June 30: Phase 1 evaluation (working on Mentor's feedback)
- July 1 – July 24: Synchronising states and filesystems on Thimble editors of all remote users using WebRTC for providing real-time development experience
- July 24 – July 28: Phase 2 evaluation (working on Mentor's feedback)
- July 29 – August 7: Fixing bugs and issues that cause functional problems
- August 7 - August 14: Integrating thimble with Together.js real-time development tools
- August 14 - August 28: Fixing all remaining issues and bugs ,working on final project deployment
- August 29 - September 5: Final project evaluation.



Experience in open source development:

I started working on open source development in the first year of my BTech course. I have found issues in codes and UI of open source organisations, and I have also developed some small projects like:

- **NotifyMe:** NotifyMe is a web app that I developed during the Microsoft's event Code.Fun.Do, it provides with mails(using gmail and outlook apis), weather information (using yahoo weather api) and a to-do-list to keep track of our work.
- **SignUp_MaterialDesign:** A login system whose UI is made with google's material design.
- **SoundProfile:** An android application that helps manage the sound profile of the phone, the profile is set according to a person's weekly time table and different time slots within the day (which is taken as input from the user). The app is developed while taking into consideration all the software development life cycles.

And many more small frontend and backend projects.

Work/Internship:

I have worked as a web designer and developer with a startup named Filterlady which provides construction tools and construction assistance. I worked in the main website design team for the startup.