WebBoard: An interactive tutoring system

Ashwin Vasani Ravinsingh Jain Sagar Kalburgi

Ira. A Fulton School Ira. A Fulton School Ira. A Fulton School

of Engineering, Arizona of Engineering, Arizona of Engineering, Arizona

State University State University State University

akvasani[@asu.edu](mailto:skalburg@asu.edu) [rvjain@asu.edu](mailto:rvjain@asu.edu) skalburg@asu.edu

***Abstract -***

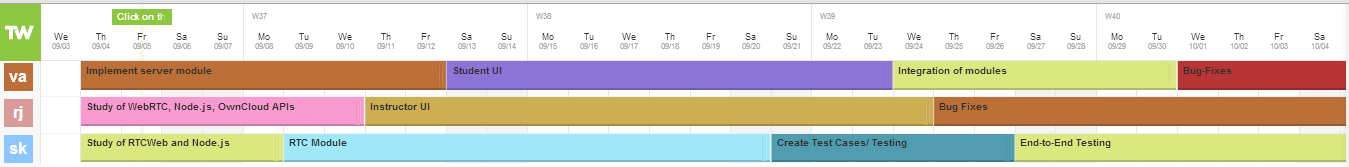
**This project is to meet the necessary requirements of online lectures and to make it easier for the instructors and the students to communicate more effectively and interactively, especially for remote students. WebBoard is an integrated tutoring application. During the presentation, the instructor can write on the application interface called WebBoard, the content of which is replicated to all the students in their WebBoards. It comes with a Question & Answer feature which allows students to ask questions in a manner that can be accessed by students participating in the class. The instructor can easily switch among the slides, video and the WebBoard. While elaborating on the concepts, the instructor can annotate the content of slides, without modifying them which makes it easier for students to understand the concepts. The lectures are recorded in the class rooms and are available to all the students later on the cloud.**

***Keywords — webRTC , Vlab, Node.js, Video conferencing, Screen sharing, WebBoard, tutoring, VP8.***

**Introduction**

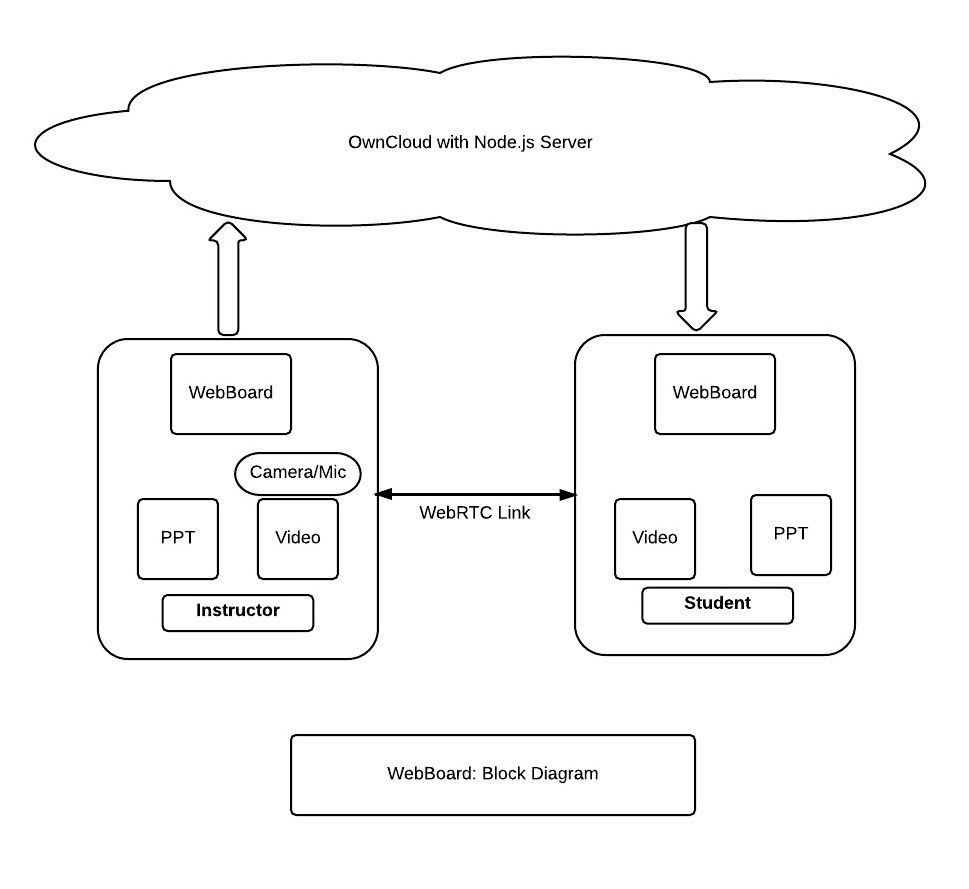
1. **Problem:** Rigid Nature of the hybrid classes and online lectures, current hybrid/online classes and challenges faced by students/instructors.
2. **Why it is important:** It gives a single system which allows the students to communicate with the instructors and other students. The lectures will be more interactive as the professor can ask some questions to the students who are remotely accessing the lectures to answer or vice-versa. The videos of the lectures would be stored on cloud so that it is easily accessible to students to recollect concepts taught in the class room. Moreover they get a classroom feel while watching the video streaming. Instructor can use the Web Board functionality to annotate on some concepts which they might do on a White board in classrooms so that even her explanatory notes can be available to all the users after the class on the cloud.

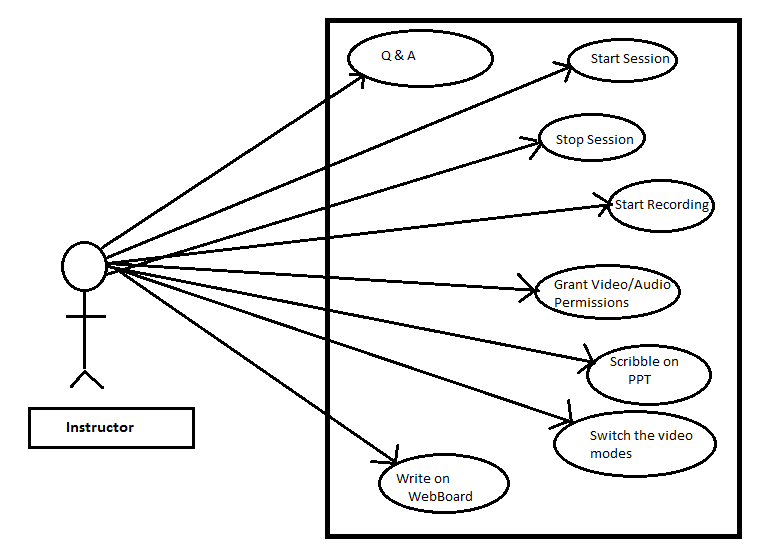
**Technologies used:** Cloud, WebRTC API, Node.js, JavaScript, CSS, HTML.

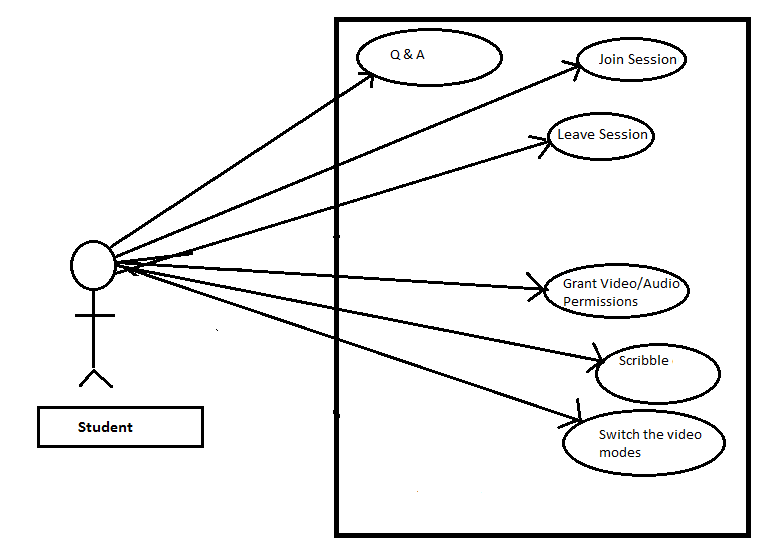
1. **Expected outcomes of this projects:**
2. Software would have the following main features -
   1. Video recording and storing in cloud
   2. WebBoard (Similar to White Board) for instructors
   3. Paper Presentation (PPT) sharing tool
3. On instructor side: She can start the session. Session can be started on Tablet using web or Android browser. She can upload **PPT** or PDF file. This **PPT** or PDF is internally converted into images. She can draw on the image and undo them. She can also switch to White Board Mode where she can draw diagrams or write notes. She can view all the students who are joining the sessions.
4. On Student Side: They can join the session initiated by the instructor. Though the direct support of annotating on WebBoard by students is not allowed in our UI implementation, we have exposed the APIs to enable this feature .Students can view the live streaming of the lectures with one-to-many broadcasts from the instructor.
5. Instructor can share the VM (Vlab) screen with students for live coding sessions.
6. **Project management plan (Timeline, and group members, etc.)**

**System Models**

1. **System Model**







1. **Software**

Tools: Sublime Text, Emacs, Git

APIs: WebRTC API, Node.js, Vlab, Heroku

Language: JavaScript, HTML & CSS.

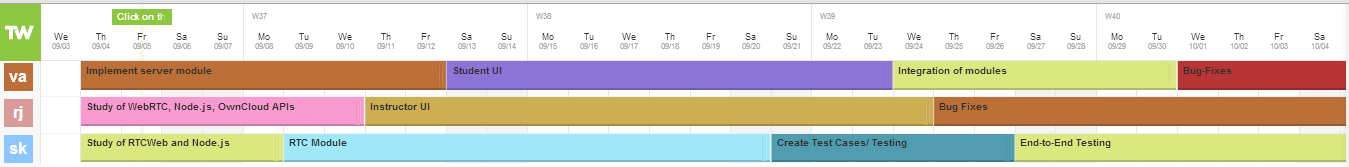
**Project Description**

1. **Project Overview**
   1. Task 1: Study WebRTC, Vlab and Node.js - We shall research on the above topics and start with implementing some of them.
   2. Task 2: Implement server module - We shall install Node.js server on localhost with some server side business logic.
   3. Task 3: RTC Module - Implement WebRTC module.
   4. Task 4: Instructor GUI: Implement Instructor Graphical user interface by using web languages.
   5. Task 5: Student GUI: Implement Student Graphical user interface by using web languages.
   6. New Task: Added a feature that enables the instructor to share her Virtual machine sharing.
   7. New Task: Added user session management.
   8. New Task: Allowing only PDF files for uploading.
   9. Task 6: Create Test Cases.[completed after midterm review]
   10. Task 7: Integration of modules.[completed after midterm review]
   11. Task 8: Test the application.[completed after midterm review]
   12. Task 9: Fixing Bugs.[ completed after midterm review]
2. **Project Task Allocation**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Name/Task | Task 1 | Task 2 | Task 3 | Task 4 | Task 5 | Task 6 | Task 7 | Task 8 | Task 9 |
| Ravinsingh Jain | Co-owner | Co-owner | Co-owner | Owner | Co-owner | Co-owner | Co-owner | Co-owner | Owner |
| Ashwin Vasani | Co-owner | Owner | Co-owner | Co-owner | Owner | Co-owner | Owner | Co-owner | Co-owner |
| Sagar Kalburgi | Owner | Co-owner | Owner | Co-owner | Co-owner | owner | Co-owner | Owner | Co-owner |

1. **Deliverables**

It is a web application which enables the instructor to use WebBoard in order to give lucid presentations. The instructor can seamlessly switch among the presentation slides, explanatory notes and video. The students, especially those who use remote access, would benefit greatly since they can ask real time questions using the WebBoard. Additionally, application can be extended to add a feature where student can also take separate personal notes and store them in cloud. She may also share her notes with her classmates, leading to healthy exchange of ideas.

1. **Project Timeline**

**Risk Management of the project**

|  |  |  |
| --- | --- | --- |
| Risk Factor | Description | Workaround |
| Bandwidth | Requires high bandwidth | Can optimize the camera resolution accordingly. |
| Security | Network security for the streaming | Private secure stream can be deployed. |
| WebRTC for native application | It is under development | AppRTC and JNI can be used at some level |

**Investigations/Research:**

**Technologies that were studied for project requirements:**

WebRTC APIs for video and audio, ICE, STUN &TURN servers for remote stream management, Node.js for hosting server, Communication architectures: one-to-one, one-to-many, many-to-many and broadcast, Bootstrap for User Interface design and MongoDB for the database of valid user credentials.

**Following research was done for WebRTC based video sharing between multiple peers:**

During the investigation, we found some potential challenges for our use-case using WebRTC. Presently, WebRTC creates two streams for audio/video for each peer (Please ref. the diagram below). This model is feasible only if there are two users.



In many-to-many approach (Please ref. the diagram below), a complex mesh is created amongst the peers which causes a very heavy network and CPU usage. This model is not feasible in our case since we are targeting that the user might be having a handheld device like smart-phone or tablet. Such devices don't have enough power to handle such a heavy requirement.



To overcome the above challenges, we tried implementing our own design. We have hacked the camera stream which we got from the WebRTC getUserMedia call and reduced the quality of the video frames. Later we sent this video frame to a node.js server using socket.io and client renders from node.js to canvas using WebRTC standard APIs. Here server also adds many headers to the packets and fragmentation of packets on network causes a bad and unacceptable performance. Hence we switched to one-to-many approach.

In one-to-many approach (Please ref. the diagram below), the total number of streams created by the instructor will be equal to twice the number of users, and only two streams will be created at the student end. This means that the instructor module should be running on the device with maximum resources but this does not apply to the students.

The maximum bandwidth of each A/V RTP stream is 1MBPS and for each peer it requires 2MB bandwidth. For future enhancements in WebRTC, we envisaged that adding WebRTC support on the server platform like node.js would prove beneficial for many-to-many and one-to-many use-cases. In this case, the server will act as a multiplexer and all the peers would connect to the server. Instead of having multiple A/V streams for all peers, sharing single video stream to multiple users can also be one of the solutions to this problem.



**Implementation Details:**

* Bootstrap was used for interactive UI design.
* Heroku cloud is an application platform known for its adaptability.
* Login credentials would be stored in MongoDB which is a highly scalable database.
* Presently we are uploading PDFs, converting into images to display them in canvas as a slideshow.
* Used meeting.js for one-to-many WebRTC implementation and firebase.js was used for WebRTC signaling.
* Using HTML5 canvas and sections for encapsulating videos elements.
* Currently we are using only single classroom but it can be scaled to multiple classrooms. Also, it is supporting event based dynamic stream addition and removal.

**Progress after the midterm report:**

* **User Registration completed:** A user can register with the WebBoard either as an instructor or as student. User’s details shall be stored in a Mongo Database. We used mongodb module from node.js to connect to the database.
* **Password security:** are hashed before they are stored in the database for security purposes. We used password-hash module from node.js for this purpose.
* **User authentication completed:** Username and password validation has been fully implemented. A user cannot access instructor home page or student home page directly, without passing authentication.
* **Start/Stop of video session:** Instructor can start or stop the video session anytime.
* **Session management:** Sessions are maintained for users, so that if they open a new tab while they are logged in, they are automatically redirected to home page. We have used express-session and cookie-parser modules for this purpose.
* **Restricted to PDF files:** We have restricted the instructors to upload PDF files only. Any attempt to upload a file of a different format would result in the user being asked to choose a PDF file.
* **Handling special characters in upload PDF:** Earlier the system was crashing, now we are supporting this feature.
* **White Board:** We implemented a feature to add a virtual whiteboard between any pair of slides. The instructor can use it to make notes, describe diagrams etc. This content is visible to all the students on the go.
* **Virtual Machine:** Instructor can share her virtual machine screen with all the students. This helps in live coding demonstrations, especially across different Operating system platforms with minimum efforts. It in turn facilitates in-depth understanding of concepts for students.

However, due to the constraints in the present Vlab system, we are manually entering the URL of the iFrame.

* **WebBoard PPT management: ‘**Next’ and ‘Previous’ buttons function properly while navigating among the slides.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case Type** | **Test Case Detail** | **Success/Failure** | **Comments** |
| User registration | Instructor able to register | Success |  |
|  | Student able to register | Success |  |
|  | Denying duplicate user registration | NA | Currently we are allowing |
|  | User able to login immediately after registration | Failure | Need to rerun index.js after the user registers, in order for her to login. |
|  | Email format check | Success |  |
|  | Password format check | Success |  |
|  | Password hashing | Success |  |
| Authentication & Login | Basic login for Instructor | Success |  |
|  | Basic login for Student | Success |  |
|  | Welcome message for Instructor, with name | Success |  |
|  | Welcome message for Students, with name | Success |  |
|  | Check for Invalid User name/Password | Success |  |
|  | Blocking unauthorized access to the instructor and student pages | Success |  |
|  | Logout | Success |  |
|  | Logout confirmation message | Success |  |
| Video, Audio Streaming | Instructor’s video streaming on student page | Success |  |
| & Storage | Getting only one video stream of instructor himself on instructor page | Success |  |
|  | Streaming video streams of all students who are logged in | Success |  |
|  | Video session start/stop working | Success |  |
|  | Student able to reconnect after network failure | Success |  |
|  | Instructor's session recovery after network failure | Success |  |
|  | Noise in the audio | Success | Local echo as expected. |
|  | Maximum number of students who can connect | Success | 10 |
|  | Instructor's ease of seeing all students who care connected | Success |  |
|  | Audio streaming | Success |  |
|  | Ability to watch the stored class lectures. | Success |  |
|  | Redundant streaming from STUN server | Success |  |
| PDF file Upload | Uploading PDF file | Success |  |
|  | Check other file formats | Success |  |
|  | Check the file name with special characters | Success |  |
|  | Server Packages installation | Success |  |
|  | Wait symbol while uploading the file | Success |  |
|  | Showing image on canvas after uploading | Success |  |
|  | Clarity of uploaded image | Success |  |
|  | Image folder created beforehand | Success |  |
|  | Ability to upload PDF file even if some other previously converted file is already present | Success |  |
|  | Only one video stream opens when I upload second file just after the first file. | Success |  |
| Annotation | Instructor annotates slides | Success |  |
|  | Students can see the instructor's annotation. | Success |  |
|  | Working of previous button | Success |  |
|  | Working of next button | Success |  |
|  | Clicking Next button when there are no more slides | Success |  |
|  | Clicking Previous button makes no changes when first slide is being displayed | Success |  |
|  | Touch pad working | Success |  |
|  | Mouse cursor working | Success |  |
|  | Works on mobile phone/tablet | Success |  |
| White board | White board functionality | Success |  |
|  | All students can see what instructor has written on her White board | Success |  |
|  | Storage of Instructor's White Board notes | Success |  |
|  | Accessibility of stored white board notes by students and instructors | Success |  |
| VM implementation | Instructor is able to share the Virtual machine | Success |  |

**Demo Snapshots(Latest):**

Figure 1: Instructor Screen

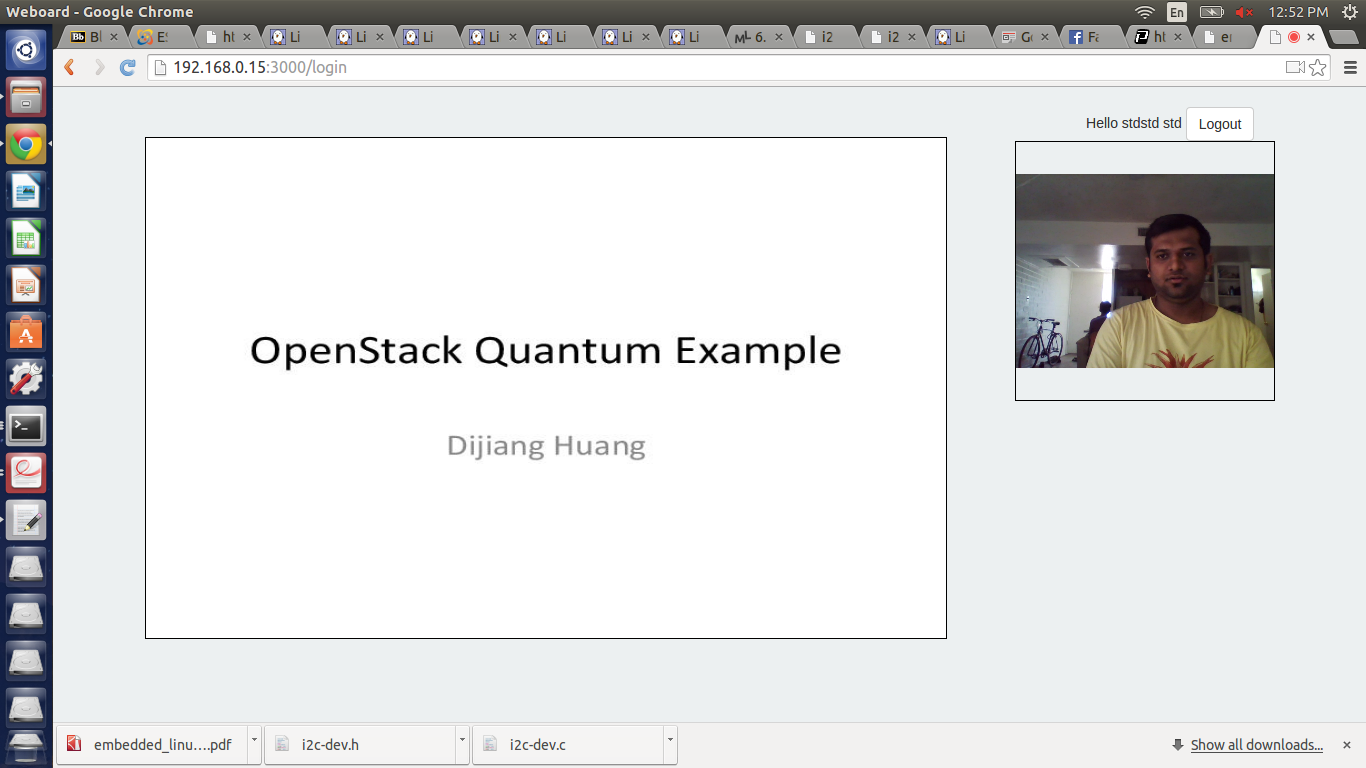


Figure 2: Interactive Class

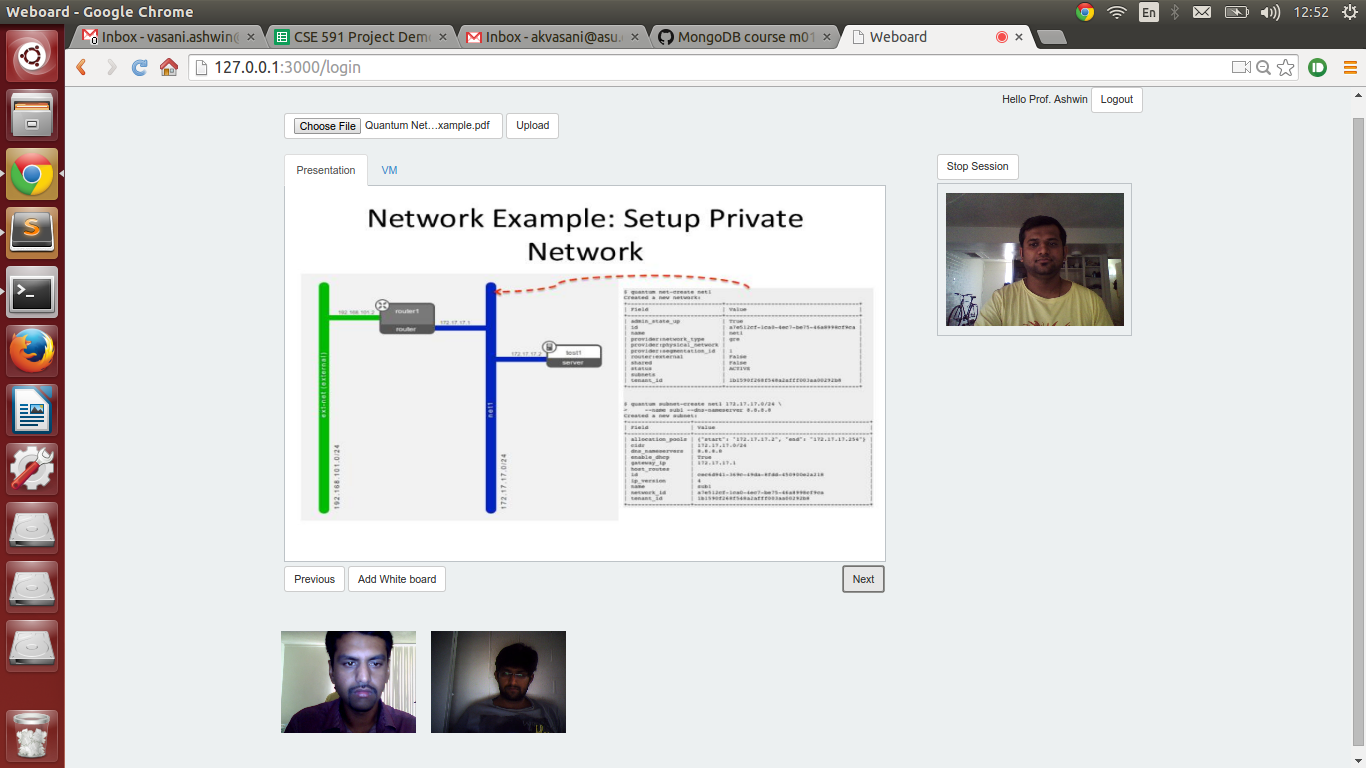


Figure 3: White board on instructor side

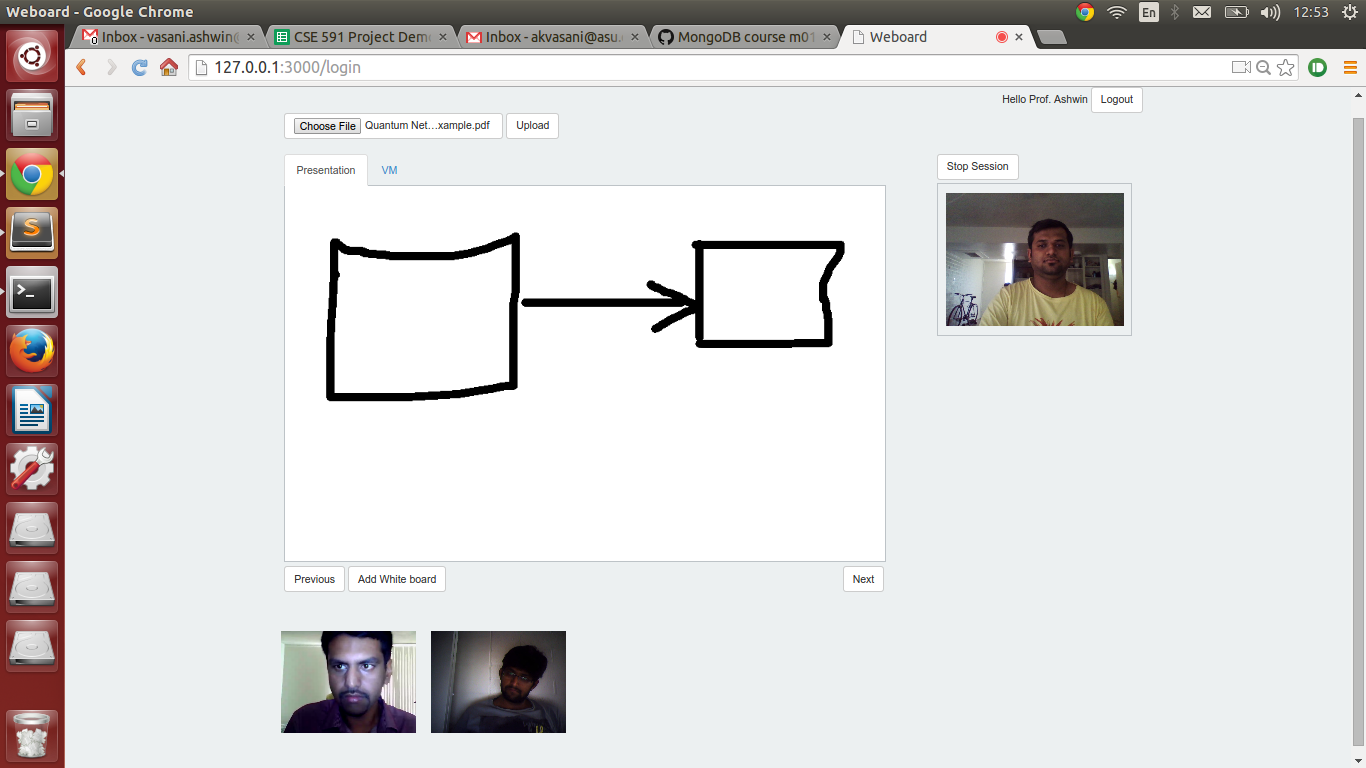


Figure 4: White board on student’s side

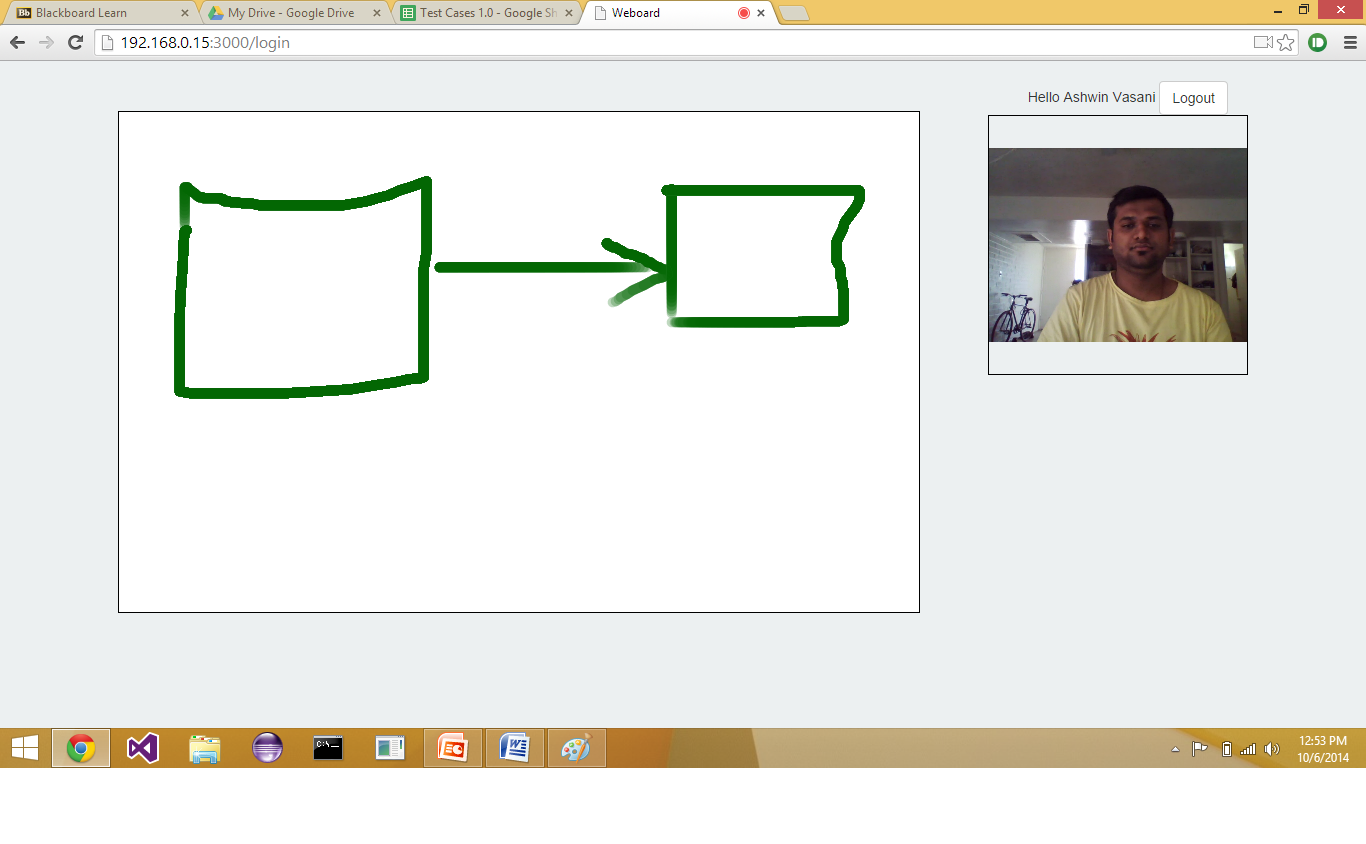


Figure 5: Virtual machine on instructor’s side

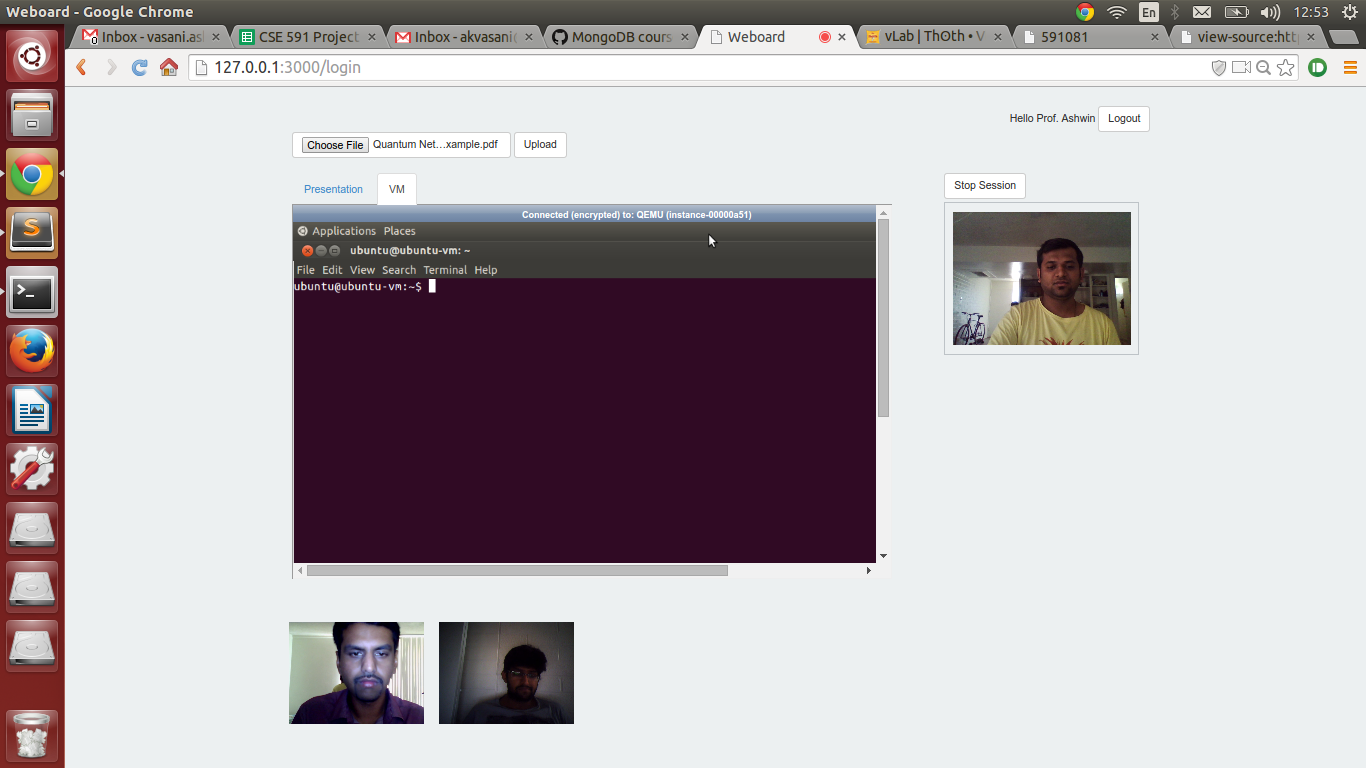


Figure 6: Instructor’s Virtual Machine on student’s side

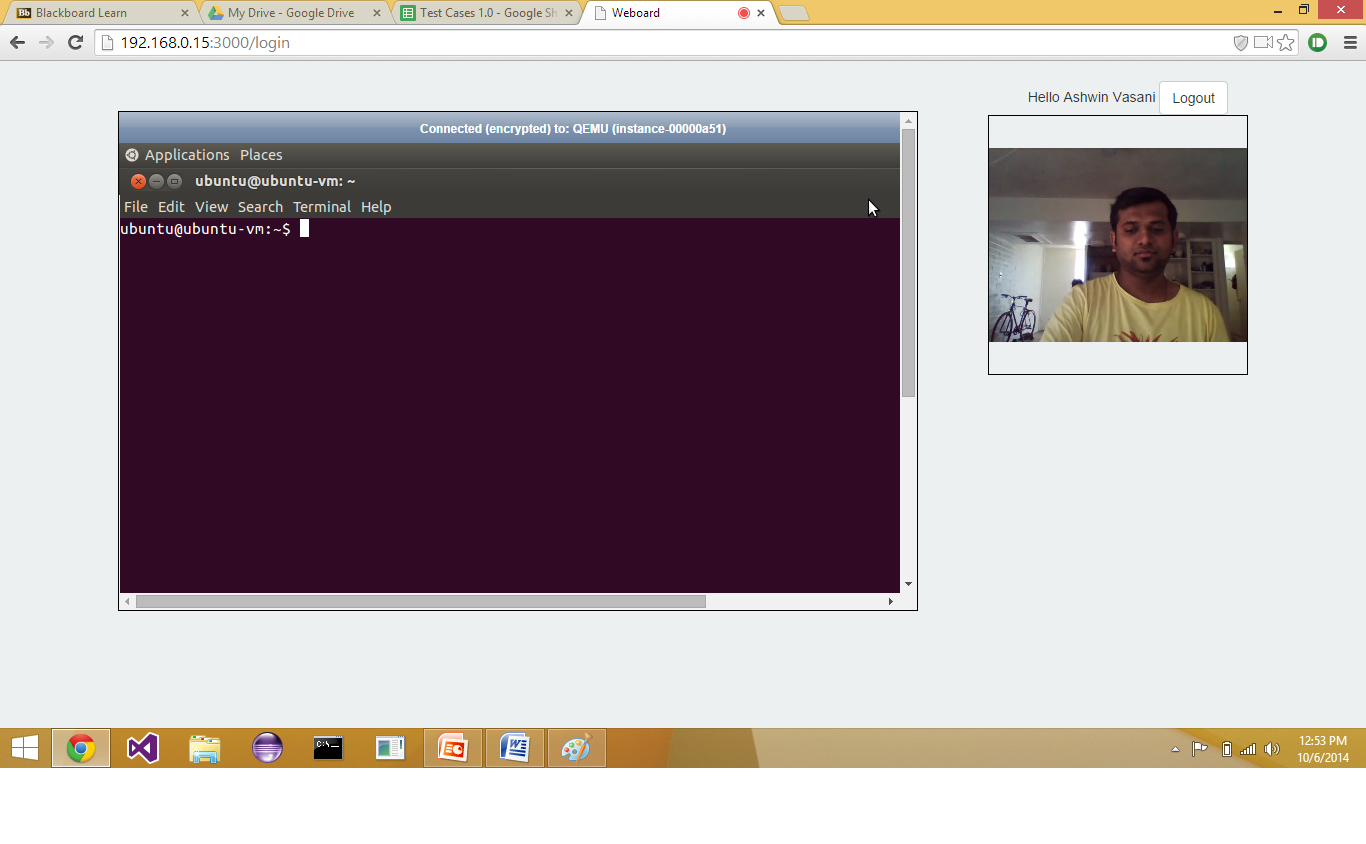
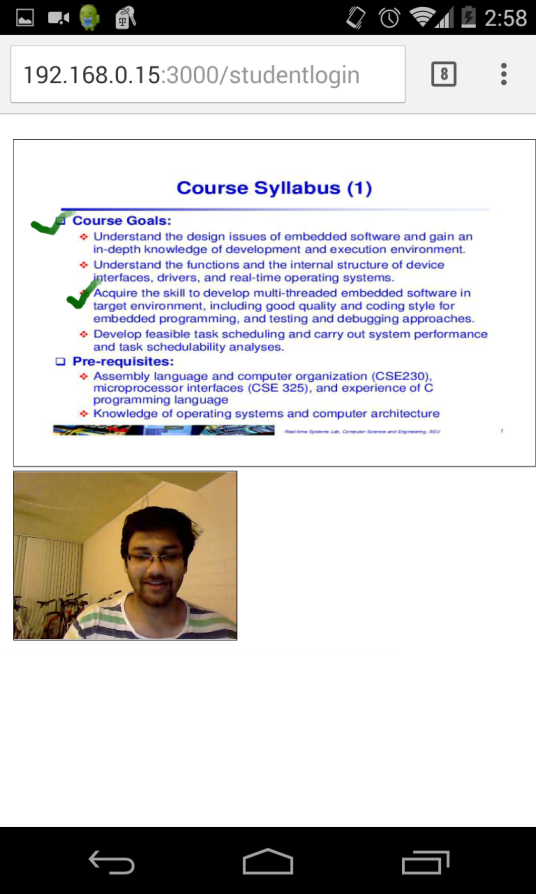


Figure 7:Application demo on Android Chrome Brower



**Conclusion**

Considering the limitations of the current online learning technologies, our WebBoard system aims to better the classroom experience, thus ensuring that the students and the instructors are on the same page, especially for remote students.

**Future prospects:**

* A Student may use a real time language translator in case she doesn’t know the medium of instruction.
* Hearing impaired students can be provided with subtitles.
* The application can be integrated with native applications
* Many to many interaction can be provided.
* Desktop sharing on cloud using WebRTC .

**Acknowledgment**

We are grateful to Prof. Dijiang Huang, Huijun Wu and Bing Li who guided and motivated us to take up this project.

**References**

1. <http://www.webrtc.org/>
2. http://tools.ietf.org/html/draft-ietf-rtcweb-data-channel-07
3. <https://www.webrtcexample.com/>
4. [https://myasucourses.asu.edu](https://myasucourses.asu.edu/)
5. Lezli Kubo, “The Rise of Online Education: Exploring the Phenomenon”
6. <http://nodejs.org/>
7. [www.htmlrocks.com](http://www.htmlrocks.com/)
8. <http://www.tmcnet.com/tmc/whitepapers/documents/whitepapers/2013/9187-webrtc-reach-web-with-new-conversation-experience.pdf>
9. https://www.digitalocean.com/community/tutorials/how-to-connect-node-js-to-a-mongodb-database-on-a-vps