**White Board Interviewer**: Requirements Document

1. Product Description

* Whiteboard interviewer is web-application that allows a better means of interaction for programming-related interviews between interviewer and interviewee.
* Generally what it does?

Essentially, it is text editor that provides real-time code-sharing such that an interviewer can see what the interviewee codes. We want to make communication during the interview is more streamlined and natural, in that video call and chat is built into our client.

* Our target audience would be company recruiters and job applicants.

Since our application streamlines the process of communication, it is much less of a clunky experience for the customer than having to switch between the code sharing interface, and handling a phone at the same time. Interviewees can focus more on the substance of the technical questions, rather than being distracted by logistics. In turn, interviewers will have a more accurate picture of the candidate’s programming abilities.

* The alternatives to this product would be :
  + Skype
    - Pros : Proven to be reliable security-wise and usability-wise. Many people are already familiar with it.
    - Cons : Need to download the desktop client, and requires frequent updates. The chat interface does not really provide coding environment (no syntax highlighting, hard to edit code). While users can alternate back and forth between skype and some code-editor, it is distracting to users who are trying to focus on the interview.
  + Google hangout
    - Pros : In-browser web app, so no need to download any software. Has also been used by many people and proven to be reliable.
    - Cons : No coding environment. Again, the purpose of our app is to reduce confusion of having to alternate between the chat-client and the programming client. Requires a google account.
  + ProgBuddy
    - Pros: Complete sandbox environment for compiling and executing code
    - Cons: Must register to join sessions, and actually creating them costs money
* Support

Given our context and constraint, we want to start small. We want to first support widely adopted browsers (Chrome and Firefox?). Also, we want to support desktop clients. Possible with a minimum resolution of (800 x 600).

2. Software Toolset

HTML5 Standard

Has features like video / audio built in, and will allow us to reduce using other API.

HTML5 adoption is not universal yet, so one disadvantage with using this standard is that there are potential people with incompatible browsers. At the same time, HTML5 will allow us to better evaluate what is supported and what is not, rather than having to research quirkiness of different browsers for HTML4 Strict, for example. Plus, we want to promote the latest and greatest, and HTML5 will allow us to market our product to people who want to support the adoption of HTML5 as well.

Frontend Framework: Twitter Bootstrap, or something equivalent.

Front-end frameworks allow web developers to rapidly create professional, clean looking pages, with dozens of UI components bundled. One burden of these frameworks is that they are written as a framework… meant to have many use cases, and so the CSS files end up being huge. By huge, I mean 50kb. This might be a concern for multi-billion dollar companies looking to reduce every millisecond of latency. But we value developer time. Given our context and deadline, loading a page 1/10th a second slower is worth saving hours of recreating the wheel. And of course, we can chop off part of the CSS we don’t need. These frameworks also have a lot of eye-candy built in, such as navigation menus, typography, progress bars, and more.

jQuery

- Dom traversal is much cleaner than native JS.

- Wider range of Dom Selectors. Also, adding / removing ID’s / Classes from

var domObject = $(“#myDomId”)  **jQuery**

vs.

var domObject = document.getElementById(myDomId) **Native JS**

from Dom nodes much easier.

- Node creation / appending to DOM tree is more intuitive

- CSS manipulation upholds OOP

jObject.width(“500”)

vs

jsObject.css.width = “500px”

- jQuery also provides better cross-browser support.

- Ajax calls are simpler

- .animate() is really fun

JSHInt

Javascript Syntax Parser. We’ll need to declare Javascript Strict in our code.

Video Communication API

We need something that extracts webcam data from client browsers, and then a means of communicating this data between client browsers. We plan to build prototypes using one or more of these APIs. Most likely, it will be a Proof of Concept to show that the API allows us to open a video stream between two browsers. We will choose the API that eases the coding, and also reliable / provide good quality video to the client. Considerations:

WebRTC

Pros: Built into browsers

- Component for extracting video/audio stream

- Component that actually transfers this data.

- The component for transferring data allows developers freedom in choosing a signalizing mechanism, ie: web sockets.

Cons: Chrome and Firefox Only

Some modern browsers support:

navigator.getUserMedia(). Possibly just find API that transfers the object

returned by navigator.getUserMedia()

Javascript Library for Code Syntax Parsing

LAMP (Web Server Stack)

If we are to host our own servers, we need several layers of software.

First there is the OS, then the web server, then some server side language that servers pages. LAMP bundles Linux, Apache (web server), and PHP into one.

We have not looked into the server side yet, but members of our team have experience with PHP from CSE154 / CSE190M.

We can also consider Ruby, and Ruby on Rails.

Amazon S3 has really cheap static web host, but thats for static web pages.

3rd party software to handle emailing clients about session infor

Possibly including API that generate unique keys for these sessions aswell.

3. Feature Set

* Major
  + Live video/audio chat - real time communication between interviewer and interviewee.
  + Live text editor - A text editor built not for chatting, but for code editing.
  + Chat Interface, other fallbacks in case client doesn’t have webcam / microphone
  + Accountless sessions - based on passwords and unique URLs, such that there is no need for users to create an account prior to using our service.
* Minor
  + Syntax highlighting - adding syntax-capability for the live text editor.
  + Canvas area - a feature where users can draw any image in real time.
  + Sessions only exist for a user-specified, limited time period - disabling the unique URL and password once the interview session is over.

4. Schedule / Workload

1. Alpha
   1. Finished
      1. Have all libraries loaded.
      2. Set up our servers, and server stack. (OS, Databases (if relevant), Apache)
      3. Basic site layout for the actual interview session

“Zero Feature” / Mission Statement Site. This site will hold both the basis

for our application, in addition to other information about our application.

* + 1. Real-time communication ability (text, audio, and video)

We will try to make isolated prototypes for each of the communication mechanisms. (Separate text / video prototypes). This would also be the time to choose between different webcam communication APIS, based on which one eases the development process, while still being reliable and high video quality for our customers. We can refactor this in later phases, and combine them into a more structured application that accommodates both of them.

1. Beta
   1. Finished
      1. Working sessions - require passwords and unique URLs
      2. Basic working setup UI for interviewer
      3. Email notifications
   2. Started
      1. Polished UI
      2. Syntax highlighting
      3. Canvas
2. Release 1.0
   1. Finished
      1. Complete, polished UI
      2. Syntax highlighting
      3. Canvas
      4. Clean up code base
      5. Optimizations
      6. Documentation

We put the basic requirements - server, basic libraries - in the Alpha version, because they’re absolutely required to get anything else done. We also put real time communication ability in the Alpha version because we foresee it being the most tricky to get done in a stable and reliable manner. No in-progress features for this stage because those core features should be everyone’s main focus at this point. The Alpha version will thus also be able to serve as proof-of-concept for our service.

For the Beta version, we want all features to be at least in-progress, so we have time to polish them by the release. We chose to complete the features that make up the core functionality of our service, so they can undergo the most extensive testing.

We chose to dedicate the release version to finishing up the more minor features, polishing the UI, and cleaning up our code because those tasks are important enough to deserve a large portion of time.

5. Risks, Cuts, Adjustments

* Teamwork.
  + Since we have the biggest team in this project, conflict between team members will be inevitable. It might not be easy to find a schedule, where everybody can gather and work on the project at the same time.
* Lack of knowledge about the technology.
  + We’ll be dealing with technology that we are not familiar yet. Hence, everybody in our team needs to overcome the steep learning-curve.
* Server Costs. We have not fixated on a particular Video Communication API yet. At the very least, these Video Communication API’s require our server to setup a communication line between the two browsers. Other APIs, might possibly use servers as a proxy to pass the Video Data around. This could be costly…

6. Use Cases Documenting Requirements

**Use Case 1. Entering an interview.**

Primary Actor: Interviewee or Interviewer (User) Level: User Goal

Precondition: User has gotten email with unique url and password, and is within the time limit of the interview page.

Minimal Guarantee: User knows what to do to eventually be able to access page.

Success Guarantee: User is forwarded to working interview page.

Main success scenario:

1. User goes to link in the email they received from our service.
2. User gets page asking for password to access the page they requested.
3. User enters password.
4. Interview page loads.

Extensions:

1a) User is using a web browser that will not support the interview page with full functionality.

1a1) Page shows error message communicating to User that the web browser they are using is not supported, and which web browsers (including version) are supported.

**Use Case 2. Setting up an interview.**

Primary Actor: Interviewer (User) Level: User Goal

Precondition: None.

Minimal Guarantee: Email is sent to interviewer’s email, if the provided is valid.

Success Guarantee: Unique url and password are generated for an interview, and appropriate emails are sent out to all valid emails entered.

Main success scenario:

1. User goes to our main site.
2. User is asked for emails of all parties who will be involved in the interview.
3. User enters emails.
4. Unique url and password are generated for the interview.
5. Emails are sent out to all email addresses given containing:
   1. Unique url / link
   2. Password for the session
   3. Which web browsers will support all of the functionality of the interview.
   4. When the page for the interview will cease to be available.

Extensions:

2a) Extra functionality (minor goal): The interviewer is asked when they would like the interview page to expire (within reasonable limits).

4a) No unique urls are available.

4a1) Interviewer is informed that an interview page cannot be created for them at this time, and how long they should wait before trying again.

A user goes to an expired page:

The interviewer or interviewee goes to the unique url which has been emailed to them after the date at which that page is no longer available. The user encounters the page asking for a unique password. The user inputs their password. The user is directed to an error page that lists possible reasons for the error (ie. mistyped url, mistyped password, expired page) and encourages the user to check the email that they received. The error page should also contain an appropriate email address to notify the person in charge of maintenance of the software in case of a true error.