1. Introduction

The idea for this project is to create a web app that allows users to video/audio call each other over the internet. The app will run using only JavaScript, which will handle both the client side and the server side. The improvement to an app like this over other competitors such as skype is that no other programmes are needed to be installed to call another user, as long as the user’s internet browser uses JavaScript then the app can be used. There are a number of benefits to using an app that runs purely on JavaScript:

1. No other application needs to be installed, saving memory.

2. Any user using a computer without administrator rights can call another user as no programme needs to be installed.

2. Requirements Analysis

3. System Model

The system is divided between 3 components. The Database (MongoDB), the Server (Node.JS, Socket.IO), the Client (Angular.JS). The client would take in the user's input (The data stream taken from the user's microphone, the user's login information), and then pass that information through a RESTful API, or over WebSockets. This data is then processed and if relevant it will either pass information to another client as requested or the data is logged into or checked against validation with the data in the database.

4. Feasibility

There are many problems with implementing a live web audio system. Most of the problems occur with NAT. Most of these problems can be fixed with enough manual testing. The project management risks stem from is that only two people are developing this project and thus very dependent on one another to make sure the sides of the project work in tandem with each other and if one of us is not able to finish the work in time it would hinder the entire project. This will solved by additional meetings to the supervisor meeting to make sure that the project stays on track. The project will also be using GitHub to ensure that the code being run is latest code developed. This solves the problem of the project developers being both located quite a distance from each other and just manually sending over files every change over a emails and such would be tedious.

5. Tests

Most of the tests will be unit tests. For the server side will be using jasmine. For the client side will be using Protractor.JS. Although some tests will have to be done manually as it is not possible to unit test how audio sounds or whether it is to the standard we expect. Unit testing can only detect whether the client has received the data.