Information

Authors: Trevor Frese, Evan Crook, Britt Christy, Kevin Malta

Team: Struct byLighting

Project Name: Benedictation

Revision History

1/22/2015: Created Living Requirements Doc

1/29/2015: Added glossary of terms, technologies

Intro

Benedict is your British bald bold blunt benevolent bourgeois busy bilingual butler who brings you beneficial background business binaries. Benedict will schedule meetings for you, take notes for you, and provide relevant information to your meeting.

Glossary of Terms

API- Application Program Interfaces are pieces of software that interact with other pieces of software

Speech recognition - the process of taking an audio signal and recognizing spoken words and converting them to text

Natural language processing - the process of analyzing text in human language (e.g. the output of speech recognition) to model and understand its semantic meaning

Benedict - the benevolent British butler who understands your commands and works to help with your meeting in any way he can (aka the personification of our virtual assistant)

System Architecture: High-Level Overview

The project has two basic parts: the video chat system, and the virtual assistant. The video chat is peer-to-peer, using WebRTC and a signalling server: the server acts as an intermediary to set up connections between clients, which then connect directly to each other. The virtual assistant uses speech recognition on each client to interpret the audio stream coming from that client and produce text tokens, and sends these to a server running the Natural Language Processing system and request handler. The server interprets the text it receives, uses natural language processing to understand commands given to it, and performs

the requests via the appropriate APIs. It then creates presentations of these results in views and sends them to the clients, where they are rendered and displayed.

Use cases:

1. As a user I can login to the website via Google Account
2. As a user I can create a chatroom so that I can talk with people
3. As a user I can invite other users to a chatroom
4. As a logged-in user, I can receive an invitation to a chatroom via notification directly on the website
5. As a non-logged-in user, I can receive an invitation to a chatroom via email
6. As a user I can view a list of all currently active chatrooms
7. As a user I can join a chatroom
8. As a user I can leave a chatroom
9. As a user I can send and receive text from one peer to another
10. As a user I can send and receive real-time audio from one peer to another
11. As a user I can send and receive real-time video from one peer to another
12. As a user I can create a group (like our #channels in slack) and have it persistent, whereas chatrooms are created and destroyed much like phone calls
13. As a user I can join a group
14. As a user I can view a list of all groups I am a member of
15. As a member of a group, I can add other users to the group
16. As a member of a group, I can leave a group

Tests

1. Login tests:
    a. can users login via their google account?
    b. Can we receive user information from the google account omniauth?
    c. Are users who don't have google accounts allowed to login?
2. Groups tests:
    a. Can multiple users join multiple groups?
    b. Are all the groups for a user listed in order of creation?
    c. Can users leave groups?
3. Chatroom tests:
    a. Can users create a chatroom?
    b. Can users invite groups to a chatroom?

c. Can invites be sent via email?

Technologies Employed

- **WebRTC** -- a set of standards for providing real-time peer-to-peer video chat in browser, as implemented by Chrome
- **Icecomm.io** -- a startup providing NAT traversal and an API over WebRTC. Icecomm is a client-side wrapper for Web Real-Time Communication (WebRTC). Developers are able to fully utilize the benefits of WebRTC without having to type any server-side code. With only six core methods, developers can easily create dynamic wep apps to transfer data and media between peers, and create connected web apps with many rooms and users.
- **NLTK** or a similar natural language processing system
- Google Calendar API, and any other APIs needed for the services Benedict provides
- **Ruby on Rails**, a full-stack web framework, for building the site
- **Pocketsphinx.js** - a speech recognition library written entirely in JavaScript and running entirely in the web browser.