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

2/24/2015: Updated for Draft 2

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

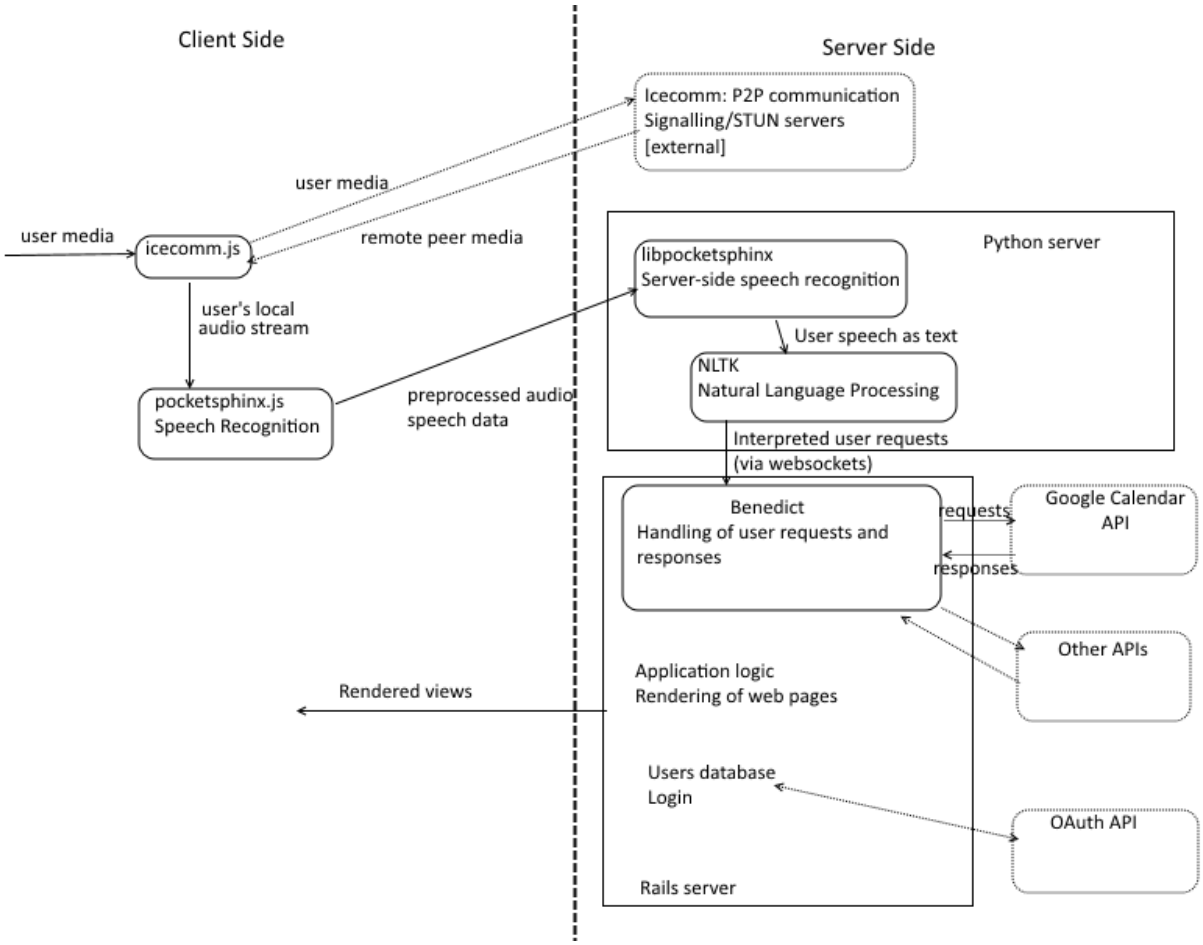
Natural language processing - the process of analyzing text in human language (e.g. the output of speech recognition) to model and understand its semantic meaning

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 via Icecomm.io which handles the 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 via Pocketsphinx.js 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 via the NLTK library 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.



Implemented User Stories:

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 logged-in user, I can create a chatroom that is associated with my group
4. As a logged-in user, I can create a temporary chatroom not associated with any group.
5. As a user I can join a chatroom
6. As a user I can leave a chatroom
7. As a user I can send and receive text from one peer to another
8. As a user I can send and receive real-time audio from one peer to another

9. As a user I can send and receive real-time video from one peer to another
10. As a user I can create a group and have it persistent, whereas chatrooms are created and destroyed much like phone calls
11. As a non-logged-in user I can visit a chatroom page, get redirected to login, and get redirected back to the chatroom I was originally trying to go to
12. As a user I can use my google profile picture as my chat icon
13. As a user I can see my most recent messages in the chat
14. As a user I can join a chatroom of up to 6 users
15. As a user my video is displayed in a pleasing format
16. As a user I can delete a group that I have created
17. As a user I can login once, and be confident that I will stay logged in until I log out

Not Implemented User Stories:

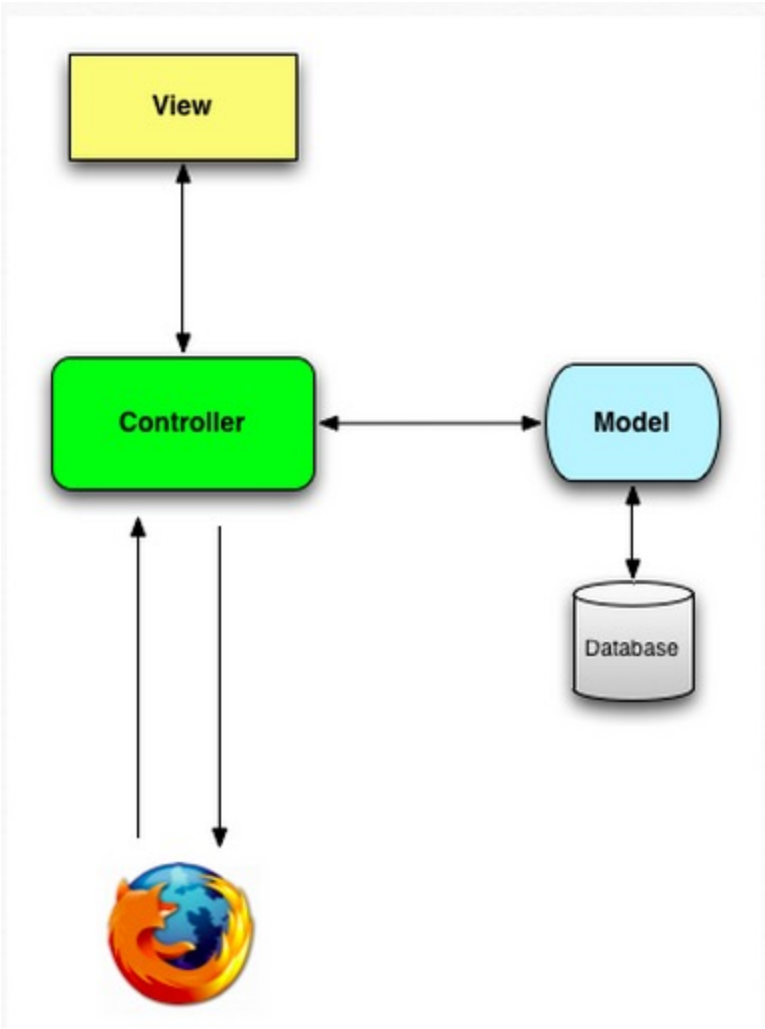
1. As a user I can invite other users to a chatroom
2. As a user I can join a group
3. As a user I can view a list of all groups I am a member of
4. As a member of a group, I can add other users to the group
5. As a member of a group, I can leave a group
6. As a user I can view a list of all currently active group chats
7. As a user I can see my speech recognition results
8. As a user I can request Benedict to schedule a meeting for a given date and time, and have the system create an event in the Google Calendar of all participating users
9. As a user I can ask for the current weather in my location via Benedict
10. As a user I can ask a direct question to Benedict about some information that I would like to receive
11. As a user I can have “Benedict on the go” and can redirect Benedict to my mobile device via our Companion app
12. As a user I can request specific information from Benedict about me from my Calendar
13. As a user I can request information about other users in the call from Benedict

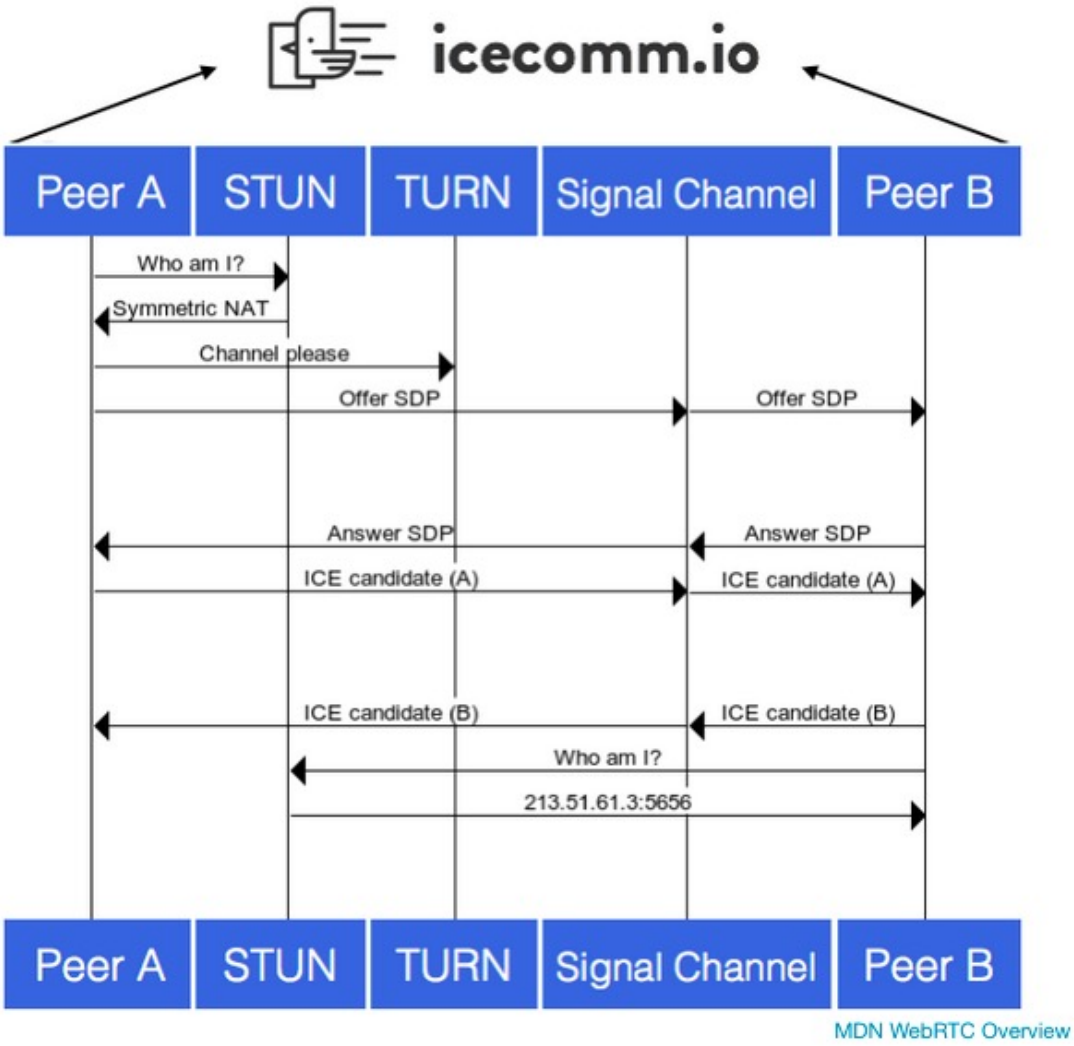
Tests

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 delete groups?
    d. Can users invite others to their 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?
    d. Can more than 6 users join a chatroom?
    e. Can a user's message show up in the chat?
    f. Does a user's correct profile picture show up in the chat?
    g. Can a not logged in user get redirected to the login page and then get redirected back to the chatroom page after?

Github URL: https://github.com/Team-Struct-by-Lightning/Benedictation

System Models (design)


View
Controller
Model
Database



MDN WebRTC Overview

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.