IVLE Hangout

Project Report

Chua Hong Jing A0074006R
Chua Wei Kuan A0072749U
Tan Swee Khoon A0072707E
Yeo JunZhi A0067400R
Le Minh Khue A0075117J

Team Hangout (Team 9)

13 Nov 2012

Content Page

1 Project Description

1.1) Statement of the Problem

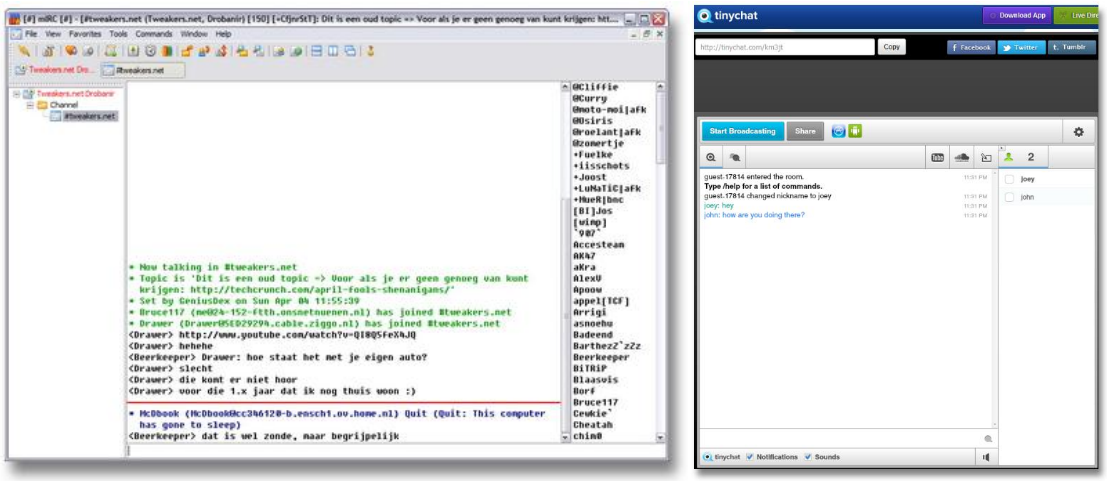
Social platforms have been evolving quickly in the past few years. New forms of interaction was invented and older technologies began to fade away. Our team have observed that social network platforms such as Facebook, Twitter or even Whatsapp might not be the best platform for active community interaction and online discussions. Some reasons could be privacy issues, distractions from Facebook games or complex user interface.

We believe that the chosen problem will benefit our group as we will experience development on cloud services, design of databases, building of Restful services, developing of modern web interface as well as developing mobile application. Furthermore, we will learn to use the IVLE Learning API (LAPI) and build a useful application for better discussion platform for the school.

1.2) Background

We propose to build an IRC-style chatting platform together with a document repository. Some of the examples in the market include mIRC or Tinychat. However, mIRC does not feature document sharing and Tinychat is open to the public. Other platforms such as Facebook or Twitter are hardly used for real-time chat. We intend to implement a real-time chat system where users can join different channels (i.e. Module, CCA, Hall/hostel channels, etc). Furthermore, each channel will contain a document repository to store any files. There will also be a notes folder where these notes can easily be viewed and updated. We believe that we will complete the above features within the given time frame.

Below are two screenshots of mIRC (left) and Tinychat (right):


[#] mIRC [#] - [#tweakers.net (Tweakers.net, Drobanir) [150] [+CfjnrStT]: Dit is een oud topic => Voor als je er geen genoeg van kunt krijgen: htt...
File View Favorites Tools Commands Window Help
Tweakers.net Dro...
#tweakers.net
Tweakers.net Drobanir
Channel
#tweakers.net
tinychat
Download App
Live Dire
http://tinychat.com/km3jt
Copy
Facebook
Twitter
Tumblr
Start Broadcasting
Share
2
tinychat
Notifications
Sounds

```
* Now talking in #tweakers.net
* Topic is 'Dit is een oud topic -> Voor als je er geen genoeg van kunt
  krijgen: http://techcrunch.com/april-fools-shenanigans/'
* Set by GeniusDex on Sun Apr 04 11:55:39
* Bruce117 (me024-152-ftth.onsnetnuenen.nl) has joined #tweakers.net
* Drawer (Drawer@5ED29294.cable.ziggo.nl) has joined #tweakers.net
<Drawer> http://www.youtube.com/watch?v=QI8Q5FeX4JQ
<Drawer> hehehe
<Beerkeeper> Drawer: hoe staat het met je eigen auto?
<Drawer> slecht
<Drawer> die komt er niet hoor
<Drawer> voor die 1.x jaar dat ik nog thuis woon :)
* McDbook (McDbook@cc346120-b.ensch1.ov.home.nl) Quit (Quit: This computer
  has gone to sleep)
<Beerkeeper> dat is wel zonde, maar begrijpelijk
```

```
@Cliffie
@Curry
@moto-moi|afk
@Osiris
@roelant|afk
@zomertje
+Fuelke
+iisschots
+Joost
+LuMaTiC|afk
+HueR|bnc
[BI]Jos
[wimp]
`907`
Accesteam
AK47
aKra
AlexV
Apoow
appel[TCF]
Arrigi
asnoehu
Badeend
BarthezZ`zZz
Beerkeeper
BiTRiP
Blaasvis
Borf
Bruce117
Ceukie`
Cheatah
chim0
```

```
guest-17814 entered the room.                 11:31 PM
Type /help for a list of commands.
guest-17814 changed nickname to joey          11:31 PM
joey: hey                                     11:31 PM
john: how are you doing there?                11:31 PM
```

```
joey
john
```

Figure 1: Similar Chat Platforms

2 Project Planning

2.1) Skills Baseline

- **Wei Kuan**
    - Basic experience with AWS and Azure using Python and Java
    - Basic experience with building REST Api using Python
    - Basic experience with Javascript, Html, CSS
- **JunZhi**
    - Basic experience with Google Appengine using Python
    - PHP Development using CakePHP and Codeigniter
    - HTML, CSS
- **Swee Khoon**
    - Experience with Java
    - Basic experience with C, PHP, HTML, CSS, Flex Actionscript, Database
- **Mnh Khue**
    - Experience with Java
    - Basic experience with PHP, HTML using CodeIgniter
- **Hong Jing**
    - Experience with Java
    - Basic experience with C, PHP, HTML, CSS

2.2) Technical Infrastructure

- **Platform**
    - AWS EC2
- **Operating System**
    - GNU/Linux Ubuntu (Server)
    - Any modern browser (Web)
- **Programming Languages**
    - Javascript
    - Python
- **Products and technologies used**
    - MySQL (Python Library)
        - Library for MySQL connector and cursor to connect to Amazon RDS.
    - Tornado Web Server
        - Web server for serving static pages and web socket connections
    - Nginx HTTP Server
        - HTTP server to route requests
    - difflib (Python Library)
        - Library for searching users and channels which uses levenshtein distance to determine best search matches.
    - Ember.js (Javascript MVC Framework)
        - MVC framework on client browser to manage user interface.
    - urllib (Python Library)

        - To access IVLE LAPI to check user access token and retrieve user information.
    - hashlib (Python Library)
        - Hashing of files to a randomised URL for storage
    - minetypes (Python Library)
        - Guessing of mimetypes of uploaded file
    - Simple HTML5 Drawing App (by *William Malone*)
        - Base drawing script for canvas drawing.
        - Modifications done to convert the script to support multiple users
    - Amazon S3
        - Storage of all documents
        - Python library: Boto

2.3) Product Perspective

IVLE Learning API (LAPI) will be used as the main authentication of students as well as retrieving their modules. IVLE Hangout is more of a add-on service to the use of IVLE for students and teaching staff. The main capabilities include:

- Real-time Chat System
- Reliable Document Repository
- Real-time Sketchboard

2.4) User Characteristics

The expected users of the system include teaching staff, faculty staff and students. Teaching staff can use IVLE Hangout as a discussion channel instead of the IVLE forum. Faculty staff can use it as an outreach to students for administrative issues. Students can join IVLE Hangout for their daily discussions on modules, projects, CCAs/Interest groups or even hall/hostel groups. Other users might be foreign exchange students or faculty guests that will just require an IVLE account to access the IVLE Hangout.

Users are expected to be experienced with the use of Social Media, online forums and real-time chat such as Facebook chat or MSN messenger. Users who wish to access IVLE Hangout through their mobile devices should also be familiar with mobile applications.

3 User requirements

3.1) Functional Requirements

- **Chat Service**
  - Able to speak in a channel
  - Able to join a channel
  - Able to listen to channel
  - Able to chat with another user in private
  - Managing channel access
- **Sketchboard Service**
  - Able to sketch on canvas
  - Able to receive updates on canvas
- **Document Repository**
  - Able to Add/Delete/Download files in a channel

**Refer to user guide for the list of features implemented*

3.2) Use Case Diagram

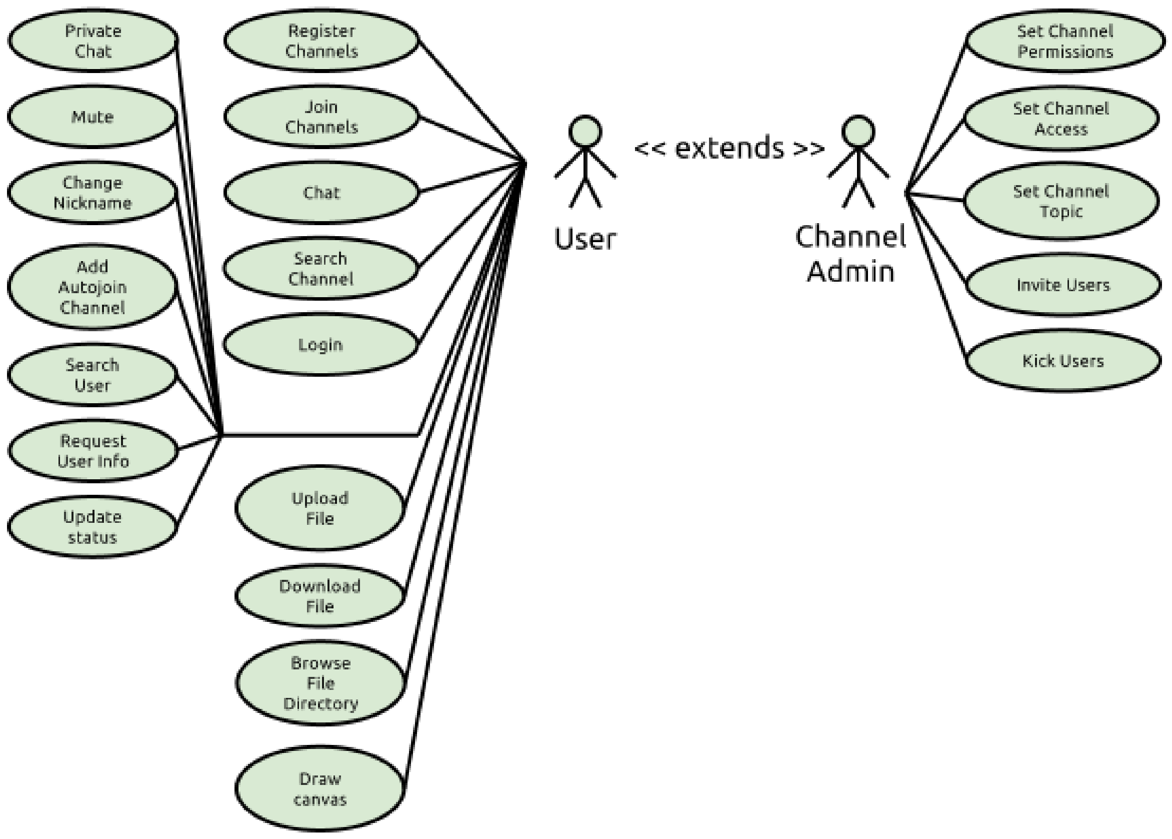


Figure 2: Use Case Diagram

3.3) Class Diagram

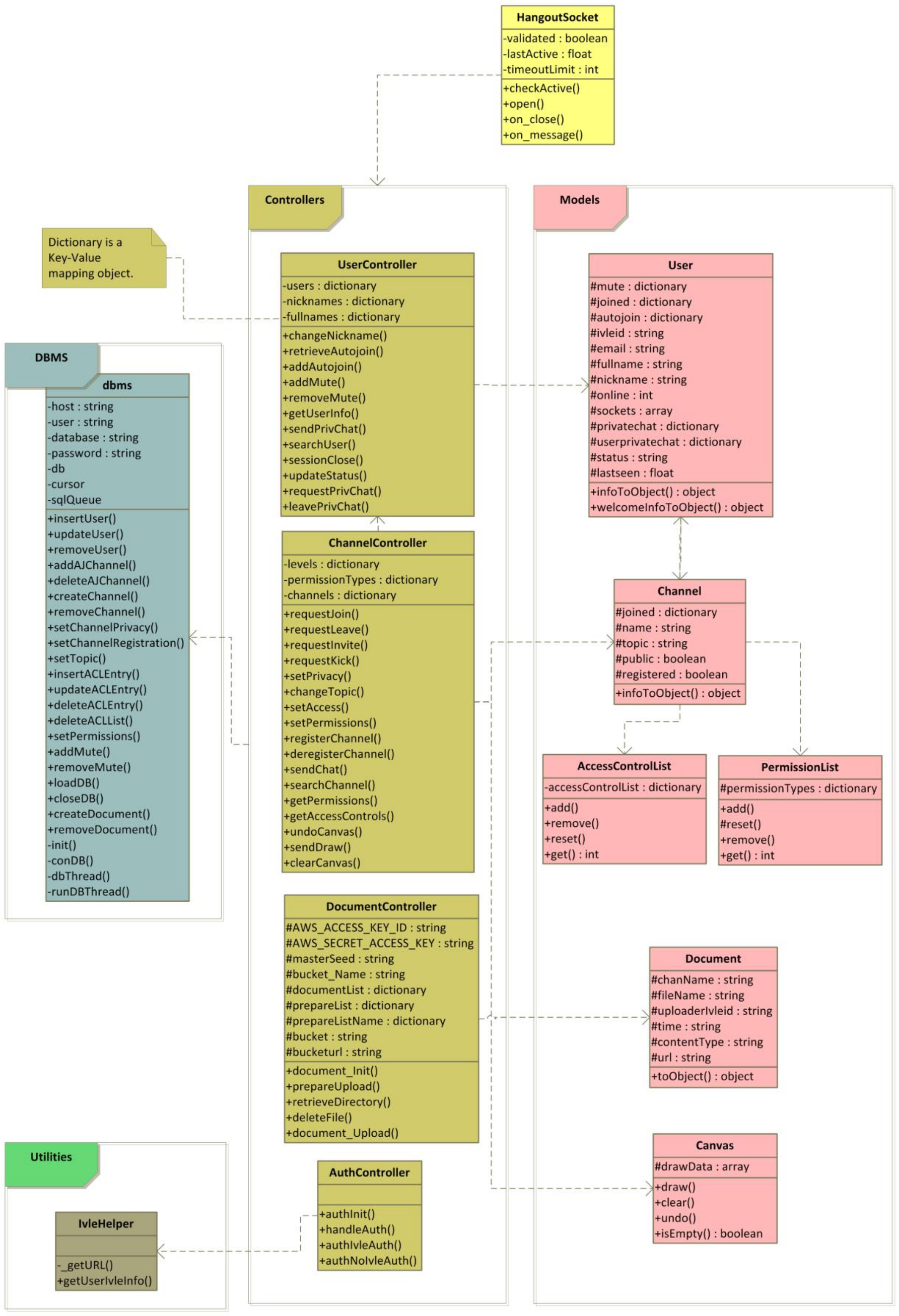


Figure 3: Class Diagram

3.4) Assumptions and Dependencies

Here are the **assumptions** and **dependencies** of IVLE Hangout:

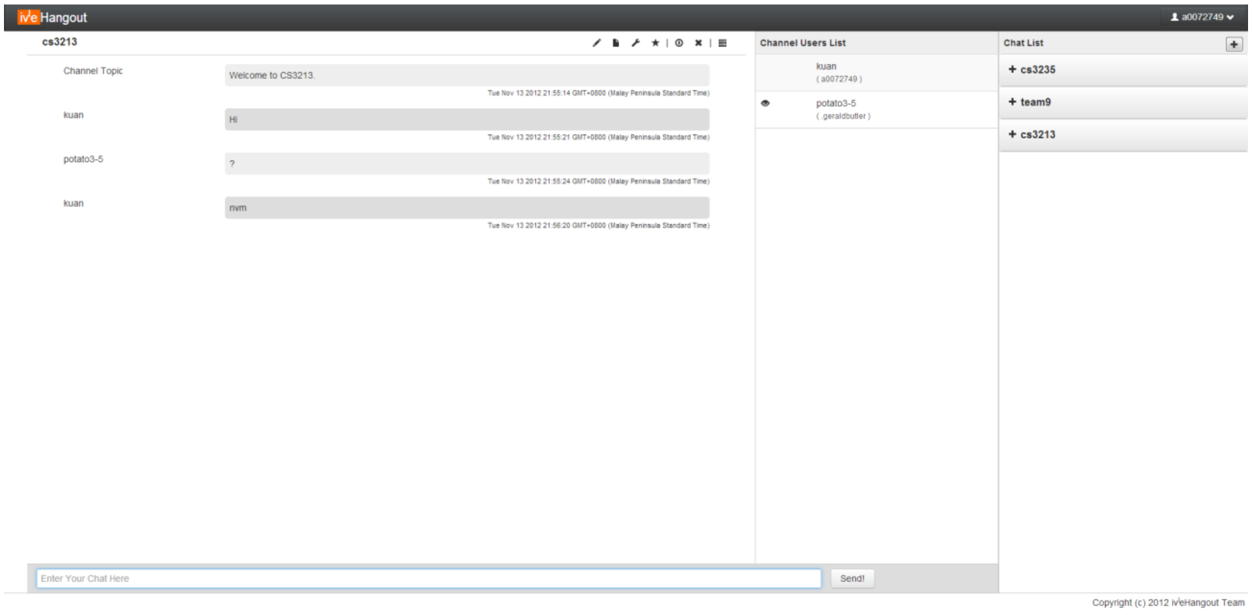
- Assume user have a modern browser that supports HTML5 and Websockets
- Staff/students needs a good discussion/chatting platform
- Dependency on Ivle API
- Dependency on Amazon Web Services

3.5) Non-Functional Requirements

- **Environment**
    - HTML5 compatible browser with Web Sockets
        - Google Chrome is used for development and testing
    - Internet connection
        - Application works on mobile browsers even when user is travelling and connection might not be as stable
- **External requirements**
    - Ivle API
        - Ivleid, full name and email are retrieved from LAPI and stored in database
        - Ivleid acts as the index for users
- **Reliability**
    - Able to handle thousands of connections at one time
        - Application is tested against 10 simultaneous users due to limited resources for testing.
    - Able to broadcast messages within reasonable amount of time
        - Under normal conditions, chat and drawing events are instantaneous.
- **Usability**
    - Minimal time to learn system especially with social network experience
        - Subjected to user preference
    - Able to improve community discussion experience
        - Some project discussions were carried out using the platform
    - Simple and Intuitive user interface
        - Subjected to user preference
- **Safety**
    - Security for channel access
        - A role based access control (RBAC) design is used to restrict user actions.
        - A permission list is used to specify the access required for a user to perform an action.
    - Security for documents access
        - The retrieving/uploading/deleting of documents are based on the "Upload" and "Download" permissions of a channel.
        - However, we have choose to use a static url to store the documents where users download them through Amazon S3.

        - To improve security, the urls are hashed with the masterseed, channel name, file name and time.
    - Security for IVLE access
        - User authenticates to retrieve a IVLE token key from the Ivle LAPI portal
- **Legal**
    - Use of channels for illegal activities
        - No monitoring is done on any channels due to privacy issues.
    - Use of document repository for illegal file hosting
        - No document screening is done on files currently

3.6) Graphical User Interface (GUI): Iteration v0.2



```
Channel Topic   Welcome to CS3213.
                Tue Nov 13 2012 21:55:14 GMT+0800 (Malay Peninsula Standard Time)
kuan            Hi
                Tue Nov 13 2012 21:55:21 GMT+0800 (Malay Peninsula Standard Time)
potato3-5       ?
                Tue Nov 13 2012 21:55:24 GMT+0800 (Malay Peninsula Standard Time)
kuan            nvm
                Tue Nov 13 2012 21:56:20 GMT+0800 (Malay Peninsula Standard Time)
```

```
kuan
( a0072749 )
potato3-5
( .geraldbutler )
```

Figure 4: Iteration v0.2 Graphical User Interface

**For more explanation on the User Interface, refer to the user guide.*

4 Architectural Design

4.1) High Level Architecture

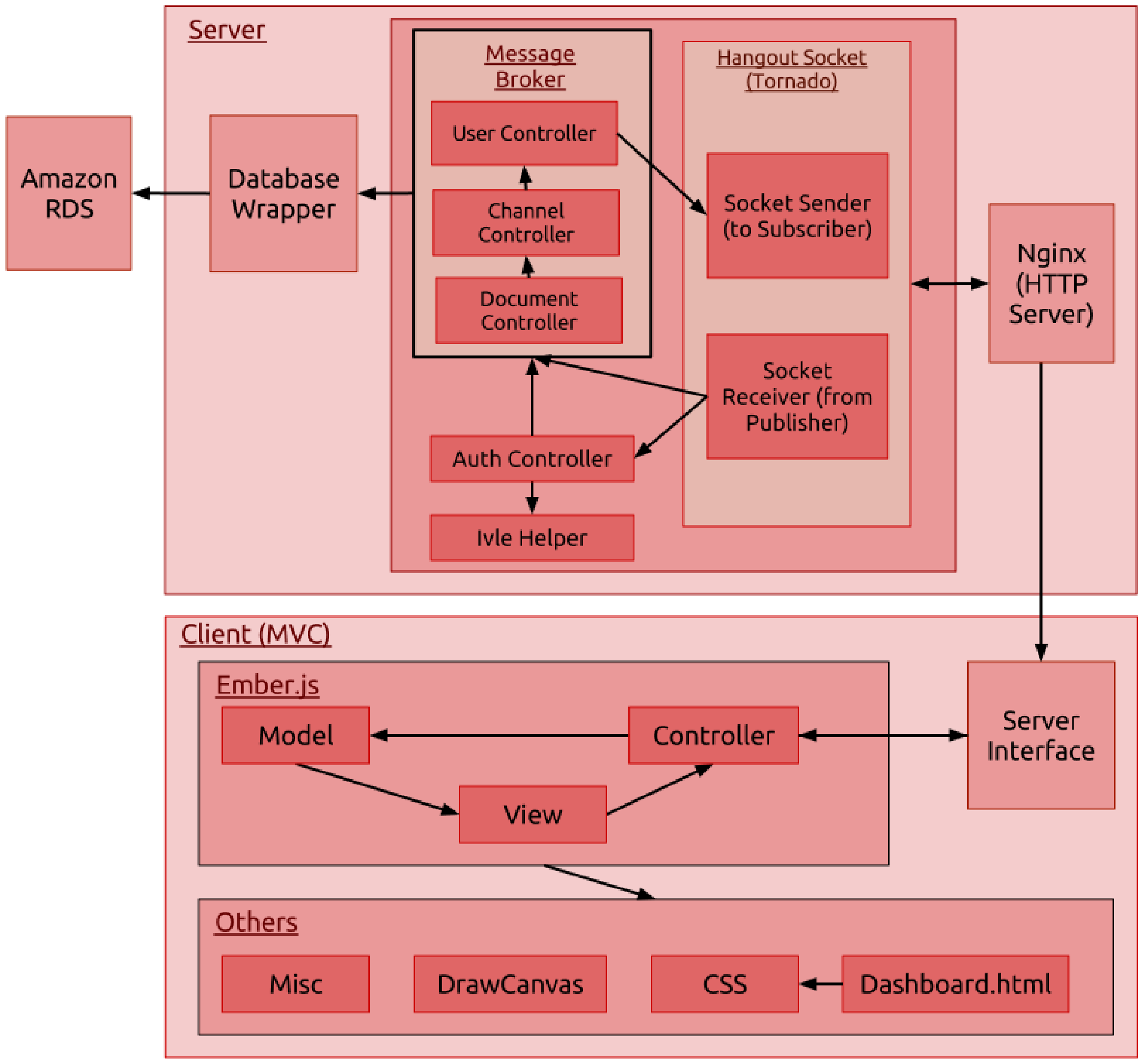


Figure 5: High Level Architecture

4.2) Components Design

4.2.1) Client Application (Ember Framework)

Pattern used: Model View controller. With MVC, updating of information is much easier as developer only needs to set the new object value and the html page will be updated automatically with template binding.
Type: User Interface
Purpose: To provide user with a modern web interface that is simple and intuitive for multiple screen layout.
Function: Provide the user with a medium to interact with the server such as receiving of message and sending of message.
Dependencies: ember.js, jquery.js, bootstrap.js, handlebars.js
Subordinates: HTML5, CSS, app.js, controllers.js, models.js, view.js, misc.js, drawcanvas.js
Interfaces: Uses **Hangout Server Interface** to communication with server.
Resources: Network connection, modern HTML5 browser
Data and Processing:

- dashboard.html
  - Contains the markup and **handlebars** for templating of html.
- view.js
  - Customised views, HTML elements such as *textfield* for further customisation that cannot be achieved with **handlebars** templating.
- models.js
  - Contains the models (extends from Ember Objects, Array etc) for various objects such as Channel, User.
  - The ember objects contain methods to allow the extended objects to be observable.
- controllers.js
  - Contains the main logic for Client application.
  - Handle server events such as new chat messages.
  - Handle user inputs such as clicking of search buttons.
  - Contains the *channelObjCollections*, *userObjCollections* and *documentObjCollections* objects that consist of collections of models (as declared in **models.js**).
- app.js
  - Javascript to create application and initialise the interface.

4.2.2) Message Broker (Channel and User Controllers)

Pattern used: Publisher-Subscriber, Role Based Access Control.
When user joined a channel, he is added to the *joined* list which contains the list of subscriber. Thus, when any channel-related activity occurs, we can immediately notify the *joined* list. As for private chat between 2 users, a *privchat* list is used. When the user updates his information, such as changing nickname, it will be published to those that are in the *privchat* list. Each channel contains an *Access Control List* ranging for -1

(*Banned* user) to 4 (*Admin*), each representing a user role. Each channel also have a corresponding *Permission* list that contains the minimum level required for various actions such as *download, chat, draw*. Requests from users can only be allowed if they are given enough access.
Type: Python script
Purpose: A component to handle all channel and user activities.
Function: Determine events type and decide which subscriber socket to send information to. It also uses database wrapper for persistence.
Dependencies: **Database wrapper** and **Hangout Socket** must be ready. *Difflib* is used for searching users/channel.
Interfaces: Python functions for sockets (Publishers) to access . Uses **DB wrapper** and socket message sending functions.
Resources: Python, difflib, urllib, json (Object representation), re (regex matching), datetime
Processing:

- Typical processing includes input checking, such as checking if the user have enough access to a particular request.
- Each controller also have their private methods, with names starting with "*user_*" or "*channel_*". These methods are not available for users as commands. User controller and Channel controller uses methods like *user_getUser()* to access a user's data.
- To publish messages to users, the *user_send()* or *user_sendSocketMessage()* is used to send to all sockets of a user (allows multiple session) or just to a particular socket respectively.
- Each channel maintains a *joined* list that contains a list of users. When events occur in a channel (such as chat messages), all users in the *joined* list will be notified.
- The private chat (between users) behaves slightly different. To start a private chat, users need to request a private chat before sending any messages. This acts as a subscription between users so that any events that the user made (such as changing nickname) will be notified to the user. The private chat Pub-Sub pattern has a 1-1 relationship between users in contrast to Channel Pub-Sub that is 1-Many.
- Both controllers are initialized with a *user* dictionary or *channel* dictionary respectively (which contains the entire dataset) as well as a reference to the **DB Wrapper** for data persistence.

Data: Channel Controller contains all *Channel* objects. User Controller contains all *User* objects. A channel object is to store all channel data. A user object is to store all user data.

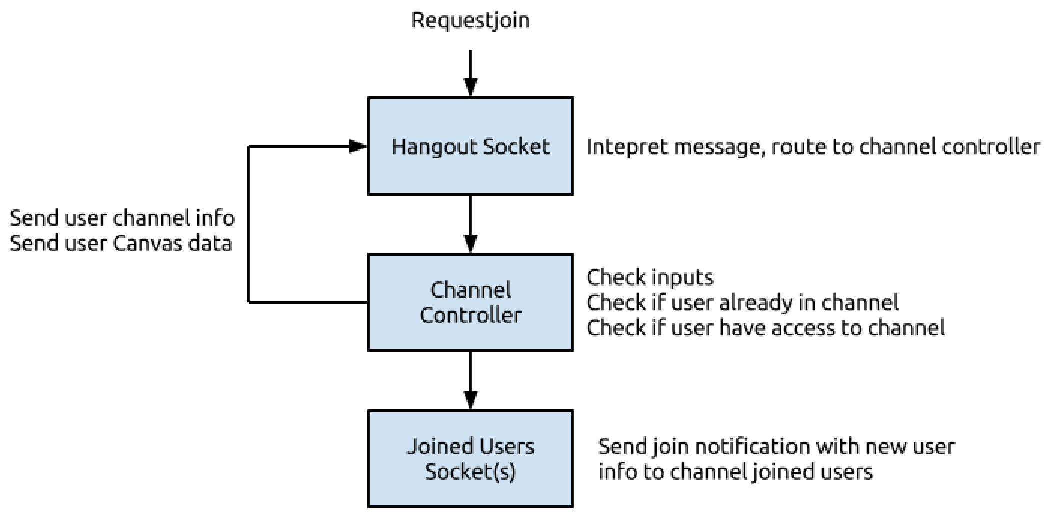

Requestjoin
Hangout Socket
Intepret message, route to channel controller
Send user channel info
Send user Canvas data
Channel Controller
Check inputs
Check if user already in channel
Check if user have access to channel
Joined Users Socket(s)
Send join notification with new user info to channel joined users

Figure 6: Illustration of request join command

4.2.3) Hangout Socket (Tornado)

Pattern used: Facade Pattern.
The socket receive messages according to the required message format for various method calls. Developers do not need to know what components/controllers are available on the server but communicate directly to the **Hangout Socket** which will then forward messages internally if required.
Type: Python script
Purpose: Communication channel for clients to server.
Function: To route messages to Auth Controller, User Controller, Channel Controller or Document Controller. Ensures that the socket is alive. The receiving message component of the Hangout socket is used for publishers to request certain actions to be published, for example sending a chat. The sending message component of the Hangout socket are subscribers and will forward the information to client interface.
Subordinates: Sender and Receiver components.
Dependencies: Auth Controller, message broker, HTTP server and User object.
Interfaces: User, socket and arguments are required to be supplied after routing to the specific functions in the controllers. Messages to client sockets are in the format specified in the "Message Format" document in the appendix.
Resources: Python, Tornado
References: Message Format in appendix
Processing:

- Determine message type and arguments.
- Forward to controllers.
- Forward server messages to users.

Data: Validated (To ensure socket is verified), lastActive (To ensure socket is still alive) and User object to identify user.

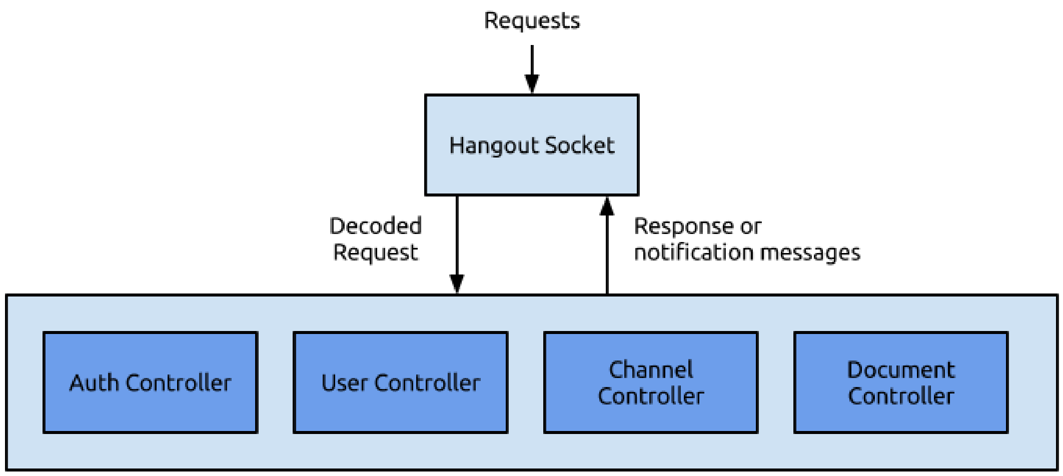


Figure 7: Illustration of relationship between Hangout Socket and Controllers

4.2.4) Server Interface (Interface for Client MVC)

Pattern used: Proxy Pattern.
The Client Application does not need to send messages or receive messages directly through the websocket. Instead, it uses the **Server Interface** for communicating. This separates certain concerns such as format of messages or keeping connection alive.
Type: Javascript
Purpose: Communication interface for clients to server.
Function: Provide methods for Client MVC to use and convert requests into messages sent to server. On receiving messages from socket, it converts messages to event objects that encapsulate event details into a single Javascript Object. It also ensures that "ping" checks are done regularly (2 minutes interval) with server to ensure connectivity.
Dependencies: Websocket object, Client MVC event handler as callback after processing messages to event objects.
Interfaces: Javascript API for Client MVC to use and callback function of Client MVC that is specified by Client MVC when event messages are received. *Refer to appendix 8.4.2 and 8.4.3 for more details about the available methods and events.*
Resources: Javascript, Websocket
References: Message Format (8.4)
Processing:

- Sending to server: Convert objects to representable formats such as JSON and format messages into a standard format understandable by the **Hangout socket**.
- Receiving from server (**Hangout socket**): Convert messages to JSON object which default contains a event type and event name. Passes event object to Client MVC *callback* function for event handling.

Data: Websocket object to connect to server.

4.2.5) Database Wrapper

Pattern used: Database Mapper.
Objects does not contain persistent methods. Instead, the database wrapper obtain information to update by accessing the objects.
Type: Python script
Purpose: Wrapper to the storage engine (*Amazon RDS, Mysql*)
Function: Provides a gateway and wrapper to a persistent storage for use in the application. Loads all objects when server application initialises. Controllers calls database wrapper functions when persistent data has changed.
Dependencies: MySQL database
Interfaces: HTTPS request, function calls from Main application to initialise data for User and Channel controllers.
Resources: MySQL
References: Developer Guide, Database Schema
Processing:

- The entire database is loaded in when the application starts. This way, we can assure that the performance of the real-time application. For writing back, each data change is written to DB shortly after the data is changed.
- A *queue* is used to store sql queries. This way, DB accesses are not done by the main application that is running the **Tornado** Application. Otherwise, with the non-blocking nature of Tornado, there is significant blocking calls to the entire server when the user does any action that require DB access.
- A DB worker thread is used to continuously read from the *queue* and writing them to the DB.
- *Logging* is also done when any errors occur (saved in log folder of application).
- Below is a sample DB write query:

```
insertData(data):
        sql = """INSERT INTO _User(IVLE_ID, Name, Email, Nickname) VALUES('%s',
        '%s', '%s', '%s')""" % (user.ivleid, user.fullname, user.email,
        user.nickname)

        self.sqlQueue.put(sql)
```

Data: Database connection and cursor.

4.2.6) Ivle Helper

Type: Python script
Purpose: Authenticate user with IVLE LAPI.
Function: Retrieve user Fullname, IvleID and email address.
Dependencies: Ivle LAPI
Interfaces: Method for Auth controller to retrieve user details.
References: IVLE Lapi documentation: http://wiki.nus.edu.sg/display/ivlelapi
Processing:

Upon calling ivle_auth, the token key supplied is then used to access IVLE LAPI to retrieve user information. Threading is used to speed up getting all three urls (Fullname, IvleID, Email). Once all data are retrieve, it is returned to the caller.

Data: IVLE API Key and IVLE LAPI authentication URL

4.2.7) Auth Controller

Type: Python script
Purpose: Authenticate users.
Function: Authenticate user and send welcome messages to initialise user interface.
Dependencies: Ivle Helper, User Controller, Channel Controller
Interfaces: Method for Hangout Socket to authenticate user. After authentication, it calls function from User Controller and Channel Controller for initialisation of user interface.
Processing:

- Once all data are retrieve from IvleHelper, it updates user with database wrapper. Finally, User controller is used to send messages to Client to initialise the User Interface. Channel controller is called to automatically allow user to join their autojoin or currently joined channels.
- *Refer to authentication sequence diagram.*

4.2.8) Client-Side Misc(Helper functions for Client Application)

Type: Javascript
Purpose: Various helper methods used in client application.
Function:

- Save a blob file to local
- Ajax uploading of file
- Get cursor positions
- Component Height
- Auto scroll channel chats
- Tooltip binding
- Help Dialog binding

Dependencies: Client Application
Processing:

- Sample code for saving a file (ANSI format) to local disk:

```
this.saveFile = function(filename, content) {
  var blob = new Blob([content]);
  var evt = document.createEvent("HTMLEvents");
  evt.initEvent("click");
  $("<a>", {
        download: filename,
        href: webkitURL.createObjectURL(blob)
  }).get(0).dispatchEvent(evt);
};
```

Data: Current uploading file (To upload after *prepareupload* is successful).

4.2.9) DrawCanvas

Pattern used: State Pattern, Command Pattern.

The current tool used (*Marker*, *Eraser* or *PaintBucket*) contains methods to handle state changes (*mousedown, mousemove, mouseup, mouseleave*) of the user's cursor. Command Pattern is used when user does a draw action or receive a draw action from other users. For example,

- When a new *stroke* is drawn, the canvas appends a new *drawdata* (contains command information) to its array and execute the particular stroke command.
- When the user *clears* the canvas, the *drawdata* commands array is cleared.
- When the user *undos* the canvas, the previous command is popped and the entire canvas is re-rendered, starting from the first command.

Type: Javascript

Purpose: Drawing canvas object for handling all drawing related actions.

Function:

- Receiving user drawing inputs
- Handling drawing input from other users
- 3 Tools (State): *Marker, Eraser, PaintBucket*
- Multiple Colors
- Multiple Sizes for Marker and Eraser

Interfaces: Uses server interface to send various user commands. Has method for client controller to edit canvas based on *drawdata* sent by server.

Processing:

- The canvas contains 2 layers, upper and bottom. When users draw, their drawdata are appended to the upper layer and sent to the server. When the user receives the same stroke from the server, it represents the completion of the command and the stroke is then removed and added to the bottom layer. This way, users can see immediately what they have drawn. For draw commands by other users, it will be applied to the bottom layer directly.
- Sample code when user moves mouse:

```
$(name).mousemove(function(e){
        //      Check access level
        if (!canvas[this.channelname].checkAcl())
                return;


        //      Get canvas position
        var mouseX = e.pageX - canvas[this.channelname].offsetLeft;
        var mouseY = e.pageY - canvas[this.channelname].offsetTop;


        //      Pointer position
        canvas[this.channelname].coordinates(mouseX, mouseY);


        //      State draw
        canvas[this.channelname].curTool.mousemove(this.channelname, mouseX, mouseY);
});
```

Data: Current Tool, current color, current size, drawdata of both upper and lower layer.

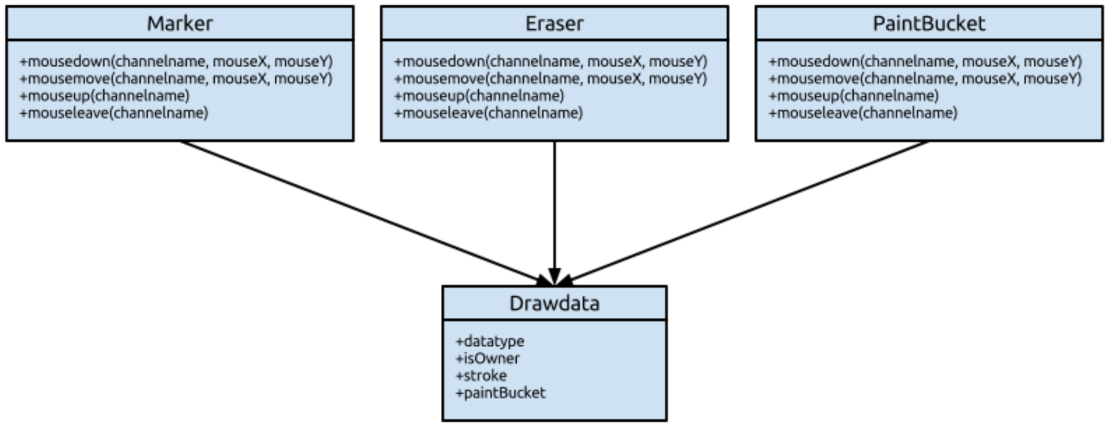

Marker
+mousedown(channelname, mouseX, mouseY)
+mousemove(channelname, mouseX, mouseY)
+mouseup(channelname)
+mouseleave(channelname)
Eraser
+mousedown(channelname, mouseX, mouseY)
+mousemove(channelname, mouseX, mouseY)
+mouseup(channelname)
+mouseleave(channelname)
PaintBucket
+mousedown(channelname, mouseX, mouseY)
+mousemove(channelname, mouseX, mouseY)
+mouseup(channelname)
+mouseleave(channelname)
Drawdata
+datatype
+isOwner
+stroke
+paintBucket

Figure 8: Canvas States

4.2.10) Document Controller

Pattern Used: Role Based Access Controller
The use of *RBAC* is the same as Channel Controller, except that only "*upload*" and "*download*" permissions are concerned.
Type: Python script
Purpose: Management of Document Repository for channels.
Function:

- Uploading of files to Amazon S3.
- Preparing newly uploaded files and checking if they are duplicates.
- Store and handle request for document directory information.
- Deletion of files that are uploaded.

Dependencies: User Controller, Channel Controller, mimetypes (python lib), base64 (python lib), hashlib (python lib), boto (Amazon lib)
Interfaces: Python functions for socket to use. Uses **DB wrapper** for persistent of file index.
Processing:

- The user first calls *prepareupload* to reserve the filename in the channel.
- The *prepareupload* function also acts as input checking.
- After the checks are done, a *sessionkey* is computed based on:
  - masterseed
  - channel name
  - file name
  - uploaderivleid
  - time
- The *Document* object (contains information of the uploading file) is then mapped into the *preparelist* dictionary, indexed by the *sessionskey* . The *sessionkey* is then sent to the user.

- When the user receives the sessionkey, it begins uploading to a separate url route, "*hangout/upload*". The Hangout_uploader component then receives the file and checks it against the *preparelist*.
- If the *sessionkey* exist, the blob file will then be set to the Document object mapped by the *sessionkey*. The blob will then be uploaded to Amazon S3 using the *sessionkey* as the url. The *Document* objection will then be moved from *preparelist* to *documentlist*.
- Uploading of files to S3 requires some additional steps. Instead of uploading the received data directly, we will need to decode it since it is in *base64*. The metadata of the file, "*Content-Disposition*", also needs to be edited in order for the file to be read correctly upon download.
- The two-part upload is used to save bandwidth as well as prevent congestion in the user's websocket that might reduce performance. The secret *sessionkey* is then used to verify the user's identity as the upload url is stateless for handling request.
- For downloading of files, users only need to retrieve the Amazon S3 url in order to access the file.

Data: Contains the dictionary containing all Document object. The Document objects are index based on "ChannelName/FileName".

4.3) Sequence Diagrams

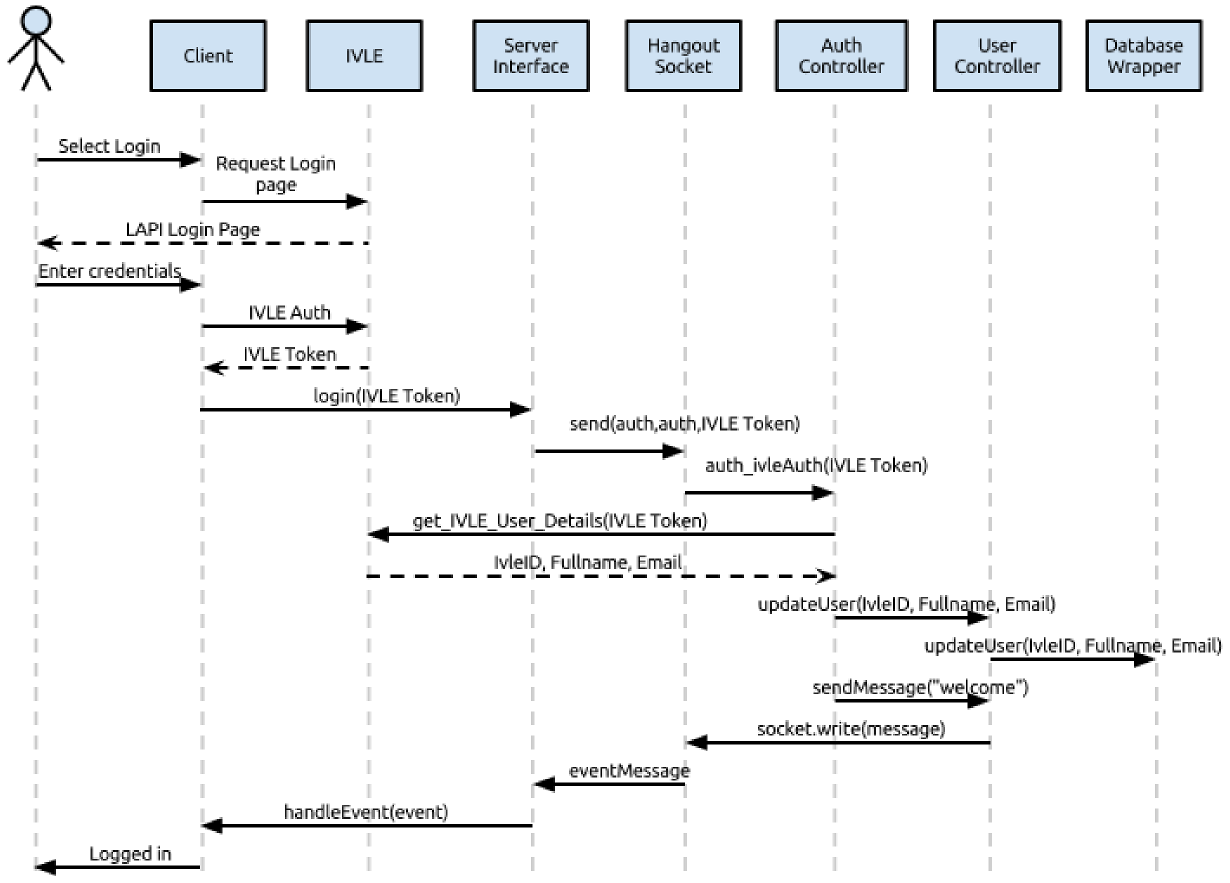


Figure 9: Authentication Sequence Diagram

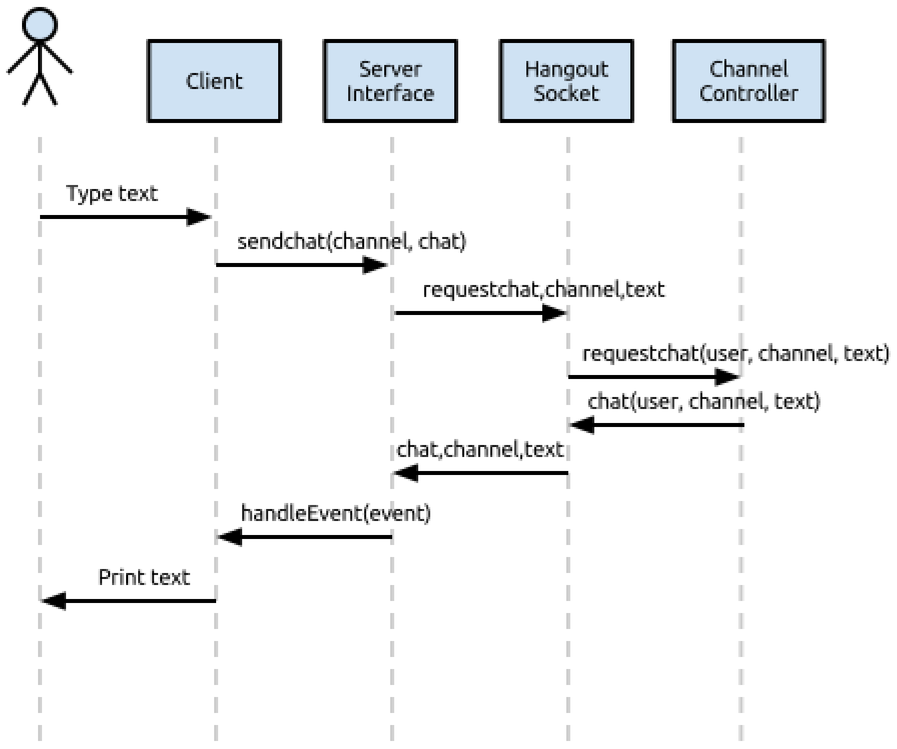


Figure 10: Send chat Sequence Diagram

4.4) Deployment Guide for Developers

The deployment is set up on Amazon EC2, Ubuntu OS 12.04. Amazon S3 is needed for documents hosting.

4.4.1) Setting up Firewall

- Firewall rules on security group:
  - 22 (SSH)   0.0.0.0/0
  - 80 (HTTP) 0.0.0.0/0
  - 443 (HTTPS) 0.0.0.0/0
  - 8888 (Websocket) 0.0.0.0/0

4.4.2) Update and installing additional software

- Commands entered:
  - sudo apt-get update
  - sudo apt-get install nginx-full

4.4.3) Configuration for Nginx HTTP server

- Nginx setup, nginx located in /etc/nginx/nginx.conf:
- Set upstream frontends ports (8888, etc..) to the same as tornado app
- Might need to open those port at AWS console.

nginx.conf:

```
user nginx;
worker_processes 1;

error_log /var/log/nginx/error.log;
pid /var/run/nginx.pid;

events {
    worker_connections 1024;
    use epoll;
}

http {
# Enumerate all the Tornado servers here
    upstream frontends {
        server 127.0.0.1:8888;
    }

    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    access_log /var/log/nginx/access.log;

    keepalive_timeout 300;
    proxy_read_timeout 300;
    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    gzip on;
    gzip_min_length 1000;
    gzip_proxied any;
    gzip_types text/plain text/html text/css text/xml
    application/x-javascript application/xml
    application/atom+xml text/javascript;

    # Only retry if there was a communication error, not a timeout
    # on the Tornado server (to avoid propagating "queries of death"
    # to all frontends)
    proxy_next_upstream error;

    server {
        listen 80;

        # Allow file uploads
        client_max_body_size 50M;

        location ^~ /static/ {
            root /var/www;
            if ($query_string) {
                expires max;
            }
        }

        location = /favicon.ico {
            rewrite (.*) /static/favicon.ico;
        }
        location = /robots.txt {
            rewrite (.*) /static/robots.txt;
        }

        location / {
            proxy_pass_header Server;
            proxy_set_header Host $http_host;
            proxy_redirect false;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Scheme $scheme;
            proxy_pass http://frontends;
        }
    }
}
```

4.4.4) Start Application

- (Copy deployment folder to "~/")
- cd HangoutApp
- screen python app.py

4.4.5) Configuration File

- The config file is available in the config folder.
- It follows a format of:
    - ITEM=VALUE
- Several configurations are required:
    - Database host, user, password, name
    - Available user *roles* in *Channel*
        - Admin (4)
        - Co-Admin (3)
        - Member (2)
        - Guest (1)
        - Visitor (0)
        - Banned (-1)
    - *Permission* Types of a *Channel*
        - Draw
        - Chat
        - Invite
        - Change Topic
        - Download
        - Upload
        - Kick
    - User *status*
        - Available
        - Away

        - Busy
    - Ivle API Key
    - Ivle API Url
    - Amazon S3 Key
    - Amazon S3 Secret
    - Amazon S3 Bucket
    - Amazon S3 Url
    - Document Master Seed
        - For hashing of url when storing documents

5 Development State

5.1) Contributions of Team members

- Server Socket and Controllers (**Wei Kuan**)
    - Main PubSub architecture (Channel and User Controllers) for server backend
    - Authentication module with IVLE
    - Includes: Channel component, User component, Auth component
- Database Wrapper (**Swee Khoon**)
    - Schema and setting up of Database (Amazon RDS)
    - Database wrapper for persistent functions and initialisation of data for server controllers
- Client Application v0.1 (**Hong Jing, Wei Kuan**)
    - Use of KnockoutJS (MVVM) for client web application
    - CSS and HTML styling and data binding
- EC2 Server setup (**Everyone**)
    - Ubuntu server on EC2
    - Installation of various middleware required for development
- Client Application v0.2 (**Junzhi**)
    - EmberJS framework (MVC) to replace iteration v0.1 framework – KnockoutJS (MVVM), which is not as suitable.
    - Model, View, Controller components
    - User Interface Design (CSS)
- Document Repository (**Khue**)
    - Amazon S3 uploading of data
- Client Application Canvas (**Swee Khoon**)
    - Sketchboard for users to draw on.
    - Downloadable copy of the sketchboard
- Chat history archive (**Hong Jing**)
    - Users can download an archive of chat messages
- Documentation
    - Class Diagram, user guide (**Hong Jing**)
    - Use Case, Sequence Diagram (**Swee Khoon**)
    - Document Component (**Khue**)
    - The rest (**Wei Kuan**)

5.2) Features not completed

- Notes Uploading/Viewing/Downloading
    - Lower priority as compared to Documents Repository
    - Various features on Client Application
        - User status, register channel, channel privacy, channel permissions, invite, kick, set access level, mute

        - However, all user actions above can be entered using a command line by prefixing “/” in a channel chat. Refer to user guide for more details and command messages for format of commands.

5.3) Future Works and Suggestions

- Document Component
    - Allow use of sub folders.
    - Currently, URL are used to retrieve the files. Although the URL are only available to those with access, the users can pass these URLs to other users who might not have permissions. Thus, it might be better to download the file through our server instead of from S3 directly.
- Client Application
  - More checks can be done on the client.
  - Currently, only a portion of user inputs are using the advantage of *Client Side Architecture*. As for now, when the user draw, his *permissions* is checked locally before sending the data to server. This reduces the server workload.
- Canvas
  - The current architecture uses command pattern to determine the canvas’s data.
  - However, when more commands are added, there could be performance issue.
  - A combination of *memento* pattern might be useful.
- Server messages
  - Currently, messages are in custom format, separated by commas for most messages (i.e, *channel,requestjoin,cs3213*).
  - There is some performance advantage as *json/xml* parser might be more resource intensive.
  - However, it might be better to conform to widely adopted standards such as *json* or *xml* when sending messages.

6 Definitions, acronyms and abbreviations

- Channel
    - Refers to the chatroom where users get to chat about a specific topic/reason.
- Channel Admin/Sub-Admin/Member/Guest
    - Refers to the administrator/sub-administrator/member/guest of the channel. Initial admin is the user who registered a room. All new users who join the channel are guests in default.
- Private Channel
    - Refers to a channel that can be accessed only to restricted users
- Hangout API
    - Refers to the API for interacting with web server
- Docs
    - Refers to document repository
- Canvas
    - Refers to the sketch area in a channel
- HangoutDB
    - Contains all persistent data attributes of users, channels, etc
- Chat
    - Refers to all chat activities
- Permissions
    - Channel minimum access required for actions such as changing topic.
- ACL
    - User access level in a channel.
- Private Chat
    - Private chatting between 2 users.
- Archive
    - Archiving of channel chat messages to a local file.

7 References

Tinychat: http://tinychat.com/
LAPI: http://wiki.nus.edu.sg/display/ivlelapi/Module
mIRC: http://www.mirc.com/
Twitter: http://www.twitter.com
Facebook: http://www.facebook.com
Whatsapp: http://www.whatsapp.com/
Cosketch: http://www.cosketch.com
HTML5 Drawing Canvas: http://www.williammalone.com/articles/create-html5-canvas-javascript-drawing-app/

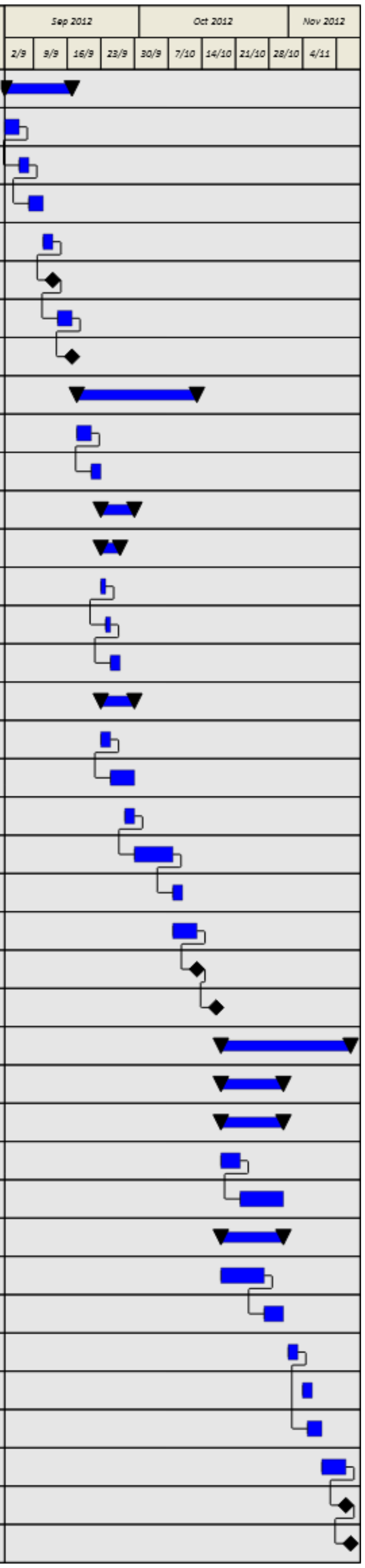
8 Appendix

8.1) Team Information

- **Chua Hong Jing A0074006R**
  - Email: a0074006@nus.edu.sg
  - Personal Number: 90922840
  - Role: Project Manager
- **Chua Wei Kuan A0072749U**
  - Email: khankuan@gmail.com
  - Personal Number: 90070372
  - Role: Software Architect
- **Tan Swee Khoon A0072707E**
  - Email: laflagaa@gmail.com
  - Personal Number: 92275178
  - Role: Software Tester
- **Yeo JunZhi A0067400R**
  - Email: yeo.junzhi@gmail.com
  - Personal Number: 96336797
  - Role: Designer
- **Le Minh Khue A0075117J**
  - Email: lmk19922000@gmail.com
  - Personal Number: 90379960
  - Role: Business Analyst

8.2) Gantt Chart

ID	Task Name	Start	Finish	Duration
1	**Iteration 1**	**3/9/2012**	**17/9/2012**	**14d**
2	Feasibility Analysis	3/9/2012	5/9/2012	3d
3	Project Planning	6/9/2012	7/9/2012	2d
4	Requirement Specification	8/9/2012	10/9/2012	3d
5	Requirement Review	11/9/2012	12/9/2012	2d
6	Feedback and review	13/9/2012	13/9/2012	0d
7	Drafting Proposal	14/9/2012	16/9/2012	3d
8	Proposal Submission	17/9/2012	17/9/2012	0d
9	**Iteration 2**	**18/9/2012**	**12/10/2012**	**25d**
10	Design	18/9/2012	20/9/2012	3d
11	Design Review	21/9/2012	22/9/2012	2d
12	Implementation	23/9/2012	29/9/2012	7d
13	**User Management**	**23/9/2012**	**26/9/2012**	**4d**
14	Basic User Adminstration	23/9/2012	23/9/2012	1d
15	Basic Authentication	24/9/2012	24/9/2012	1d
16	IVLE Authentication	25/9/2012	26/9/2012	2d
17	**Channel Management**	**23/9/2012**	**29/9/2012**	**7d**
18	Basic Channel Administration	23/9/2012	24/9/2012	2d
19	Channel Chatting	25/9/2012	29/9/2012	5d
20	Unit Testing	28/9/2012	29/9/2012	2d
21	System Integration	30/9/2012	7/10/2012	8d
22	System Testing	8/10/2012	9/10/2012	2d
23	Documentation	8/10/2012	12/10/2012	5d
24	Document Submission	13/10/2012	13/10/2012	0d
25	Prototype Demo	17/10/2012	17/10/2012	0d
26	**Iteration 3**	**18/10/2012**	**14/11/2012**	**27d**
27	**Implementation**	**18/10/2012**	**30/10/2012**	**13d**
28	**Document Management**	**18/10/2012**	**30/10/2012**	**13d**
29	Notes	18/10/2012	21/10/2012	4d
30	Other media	22/10/2012	30/10/2012	9d
31	**Sketchboard**	**18/10/2012**	**30/10/2012**	**13d**
32	Sketchboard implementation	18/10/2012	26/10/2012	9d
33	Sketchboard Integration	27/10/2012	30/10/2012	4d
34	Unit Testing	1/11/2012	2/11/2012	2d
35	System Integration	4/11/2012	5/11/2012	2d
36	System Testing	5/11/2012	7/11/2012	3d
37	Documentation	8/11/2012	12/11/2012	5d
38	Documentation Submission	13/11/2012	13/11/2012	0d
39	Software Demo	14/11/2012	14/11/2012	0d



8.3) Database Schema

```
CREATE TABLE `_Permission` (
  `ChannelName` varchar(128) NOT NULL PRIMARY KEY,
  `Download` int(11) NOT NULL DEFAULT '1',
  `Upload` int(11) NOT NULL DEFAULT '1',
  `Kick` int(11) NOT NULL DEFAULT '1',
  `Invite` int(11) NOT NULL DEFAULT '1',
  `Promote` int(11) NOT NULL DEFAULT '1',
  `Demote` int(11) NOT NULL DEFAULT '1',
  `Change_Topic` int(11) NOT NULL DEFAULT '1'
)
CREATE TABLE `_ACL` (
  `IVLE_ID` varchar(45) NOT NULL,
  `ChannelName` varchar(128) NOT NULL,
  `Level` int(11) NOT NULL DEFAULT '1',
  PRIMARY KEY (`IVLE_ID`,`ChannelName`)
)
CREATE TABLE `_Channel` (
  `ChannelID` int(11) NOT NULL AUTO_INCREMENT,
  `ChannelName` varchar(128) NOT NULL,
  `Topic` varchar(256) DEFAULT NULL,
  `Type` varchar(7) NOT NULL DEFAULT 'public',
  PRIMARY KEY (`ChannelID`,`ChannelName`)
)
CREATE TABLE `_AutoJoin` (
  `IVLE_ID` varchar(128) NOT NULL,
  `ChannelName` varchar(128) NOT NULL,
  PRIMARY KEY (`IVLE_ID`,`ChannelName`),
  KEY `_ivle_id` (`IVLE_ID`),
  CONSTRAINT `_ivle_id` FOREIGN KEY (`IVLE_ID`) REFERENCES `_User` (`IVLE_ID`) ON DELETE NO
ACTION ON UPDATE NO ACTION
)
ALTER TABLE `HangoutDB`.`_AutoJoin`
  ADD CONSTRAINT `_Channel_Name`
  FOREIGN KEY (`ChannelName` )
  REFERENCES `HangoutDB`.`_Channel` (`ChannelName` )
  ON DELETE CASCADE
  ON UPDATE CASCADE
, ADD INDEX `_Channel_Name` (`ChannelName` ASC)

CREATE TABLE `_Document` (
 `channel_name` varchar(128) NOT NULL,
 `filename` varchar(256) NOT NULL,
 `ivleid` varchar(128) NOT NULL,
 `time` varchar(64) NOT NULL,
 `content_type` varchar(64) NOT NULL,
 `url` varchar(256) NOT NULL,
 KEY `chan_name` (`channel_name`),
 FOREIGN KEY (`channel_name`) REFERENCES `_Channel` (`ChannelName`) ON DELETE CASCADE ON UPDATE
CASCADE
)
```

8.4) Message Formats

8.4.1) Messages from client to server

```
Login: auth,auth,IVLE_TOKEN
Logout: user,sessionclose
Channel Chat: channel,sendchat,CHANNEL_NAME,MESSAGE
Register Channel: channel,register,CHANNEL_NAME
```

```
Deregister Channel: channel,deregister,CHANNEL_NAME
Join Channel: channel,join,CHANNEL_NAME
Leave Channel: channel,leave,CHANNEL_NAME
Invite to Channel: channel,invite,CHANNEL_NAME,IVLEID
Kick from Channel: channel,requestkick,CHANNEL_NAME,IVLEID
Change Channel Topic: channel,changetopic,CHANNEL_NAME,TOPIC
Change Channel Privacy: channel,setprivacy,CHANNEL_NAME,public/private
Set Channel ACL: channel,setaccess,CHANNEL_NAME,IVLEID,-1,0,1,2,3,4
Set Channel Permissions: channel,setpermissions,CHANNEL_NAME,{“chat”: 1, “upload”: 2}
Search Channel: channel,searchchannel,QUERY_NAME

Change Nickname: user,changenickname,NEW_NICKNAME
Mute: user,addmute,IVLEID
Unmute: user,removemute,IVLEID
Private Chat: user,requestprivchat,IVLEID
Leave Private Chat: user,leaveprivchat,IVLEID
Send Private Chat: user,sendprivchat,IVLEID,MESSAGE
Add Autojoin Channel: user,addautojoin,CHANNEL_NAME
Delete Autojoin Channel: user,deleteautojoin,CHANNEL_NAME
Search User: user,searchuser,QUERY (query can be IVLEID, NICKNAME or FULLNAME)
Get User Info: user,getuserinfo,IVLEID
Update Status: user,updatestatus,Away/Available/Busy

Send Draw: channel,senddraw,CHANNEL_NAME,DRAWDATA
Clear Canvas: channel,clearcanvas,CHANNEL_NAME
Undo Canvas: channel,undocanvas,CHANNEL_NAME

Prepare Upload: document,prepareupload,CHANNEL_NAME,FILENAME
Retrieve Directory: document,retrievedirectory,CHANNEL_NAME
Delete File: document,deletefile,CHANNEL_NAME,FILENAME
```

8.4.2) Methods available in Hangout Server Interface

```
this.login = function(type, key)
this.logout = function()
this.sendCommand = function(command)
this.sendchat = function(channelname, text)
this.registerchannel = function(channelname)
this.deregisterchannel = function(channelname)
this.requestjoin = function(channelname)
this.requestleave = function(channelname)
this.requestinvite = function(channelname, ivleid)
this.requestkick = function(channelname, ivleid)
this.changetopic = function(channelname, topic)
this.setprivacy = function(channelname, publicBit)
this.setaccess = function(channelname, ivleid, level)
this.setpermissions = function(channelname, permissions)
this.searchchannel = function(query)
this.changenickname = function(nickname)
this.addmute = function(ivleid)
this.unmute = function(ivleid)
this.requestprivchat = function(ivleid)
this.leaveprivchat = function(ivleid)
this.sendprivchat = function(ivleid, text)
this.addautojoin = function(channelname)
this.deleteautojoin = function(channelname)
this.searchuser = function(query)
this.getuserinfo = function(ivleid)
this.updatestatus = function(status)
this.senddraw = function(channelname, drawData)
this.clearcanvas = function(channelname)
this.undocanvas = function(channelname)

this.prepareupload = function(channelname, filename)
this.retrievedirectory = function(channelname)
this.deletefile = function(channelname, filename)
```

8.4.3) Event callbacks from Hangout Server Interface

```
--------- Format ---------
*** event.type, event.name, event.timestamp
[Other variables]
--------------------------

/*
*************************
* User
*************************
*/

*** user, userinfo
event.user (object)

*** user, changednickname
event.ivleid
event.nickname

*** user, addedautojoin
event.channelname

*** user, deletedautojoin
event.channelname

*** user, muted
event.ivleid


*** user, removedmute
event.ivleid

*** user, requestedprivchat
event.ivleid

*** user, leaveprivchat
event.ivleid

*** user, privchat
event.fromIvleid
event.toIvleid
event.message

*** user, searcheduser
event.results (Object: Array of User objects, refer to objects info section below)
```

```
*** user, sessionclose

*** user, updatestatus
event.ivleid
event.status



/*
 *************************
 * Channel
 *************************
 */

*** channel, chat
event.channelname
event.ivleid
event.message

*** channel, join
event.channelname
event.user (Object: User object, refer to objects info section below)

*** channel, leave
event.isDisconnect (boolean)
event.channelname
event.ivleid

*** channel, joinedchannel
event.channel (Object: Channel objects, refer to objects info section below)

*** channel, topic
event.channelname
event.ivleid
event.topic

*** channel, permissions
event.channelname
event.permissions (Object: Permission object, refer to objects info section below)

*** channel, setaccess
event.channelname
event.ivleid
event.level

*** channel, registered
event.channelname

*** channel, deregistered
event.channelname

*** channel, privacy
event.channelname
event.isPublic (boolean)

*** channel, invite
event.fromIvleid
event.toIvleid
event.toNickname
event.channelname

*** channel, invited
event.fromIvleid
event.fromNickname
event.channelname

*** channel, kicked
event.fromIvleid
event.toIvleid
event.channelname

*** channel, searchedchannel
event.results (Object: Array of channel objects, refer to objects info section below)



/*
 ************************
 * Channel, Canvas
 ************************
 */

*** channel, draw
event.channelname
event.ivleid
event.drawData
event.isOwner

*** channel, clearedcanvas
event.channelname
event.ivleid

*** channel, canvasdata
event.channelname
event.canvasdata

*** channel, undidcanvas
event.channelname
event.ivleid




/*
 ************************
 * Document
 ************************
 */

*** document, preparedupload
event.sessionkey

*** document, retrieveddirectory
event.files
event.channelname
```

```
*** document, uploadedfile
event.file
event.channelname

*** document, deletedfile
event.file
event.channelname

*** document, useruploaded
event.file
event.channelname



/*
 *************************
 * Notice
 *************************
 */



*** notice, [event.name can ignore, but currently contains either channel or user]
event.message
event.commandtype (channel or user)
*Note: event.name refers to command name




/*
 *************************
 * Welcome
 *************************
 */

*** welcome, userinfo
event.selfuser (Object: User object of ownself)

*** welcome, userpreference
event.preferences (Object: Preference Object)

*** welcome, availablelevels
event.availableroles (Object: Available roles)

*** welcome, availablepermissions
event.availablepermissions (Object: Available permissions)



/*
 *************************
 * Connection
 *************************
 */

*** connection, closed
*** connection, error
event.error (exception objection)



/*
 ************************
 * Object Info
 ************************
 */

*** User Object
user['ivleid'] = user.ivleid
user['email'] = user.email
user['fullname'] = user.fullname
user['nickname'] = user.nickname
user['online'] = user.online
user['joined'] = [channelname1, channelname2]
user['status'] = user.status
user['lastseen'] = user.lastseen

*** Channel Object
channel['name'] = self.name
channel['topic'] = self.topic
channel['isPublic'] = self.public
channel['isRegistered'] = self.registered
channel['joined'] = [user_object1, user_object2]
channel['acl'] = AccessControlList Object
channel['permissions'] = PermissionTypes Object

*** AccessControlList Object
dictionary:
{ivleid: level}

*** PermissionTypes Object
dictionary:
{type: level}

*** Preference Object
preferences['mute'] = [muteIvleid1, muteIvleid2]
preferences['autojoin'] = [channelname1, channelname2]

*** Available Roles
{"Guest": 1, "Admin": 4, "Visitor": 0, "Banned": -1, "Member": 2, "Co-Admin": 3}

*** Available Permissions
{"draw": 0, "invite": 0, "upload": 0, "topic": 0, "chat": 0, "changeTopic": 0, "download": 0,
"kick": 0}

*** File Object
output['channame'] = self.channame
output['filename'] = self.filename
output['uploaderivleid'] = self.uploaderivleid
output['time'] = self.time
output['contenttype'] = self.contenttype
output['url'] = self.url
```

IVLE Hangout

User Guide

Features of IVLE Hangout

1. Drawing and Saving of Canvas
    a. Multiple tools
    b. Multiple colors
    c. Multiple sizes
    d. Sync across users
2. Document Repository for channels
    a. Uploading of documents
    b. Downloading of documents
    c. Retrieve of directory information
3. Logging of chat history
4. Public/Private chat room (channel)
    a. Chat among users
    b. Emoticons for chat with regex and emoji.js
5. Private chatting between 2 users
6. User management
    a. Customisable nickname
    b. Mute list
    c. Autojoin (channel) preferences
    d. Status (Available, Away, Busy)
7. Channel management
    a. Registration of Channel
    b. Invite users to channel
    c. Kick users
    d. Change channel Topic
    e. Set Channel Privacy
    f. Set access levels of users in channel
    g. Set permissions levels of channel
8. Searching of users and channels

The Login Screen

In order to use IVLE Hangout system, user has to login using their IVLE ID.

1. Highlighted in **red** is the "Login!" button for user to click in order to authenticate himself with IVLE to use the IVLE Hangout system.

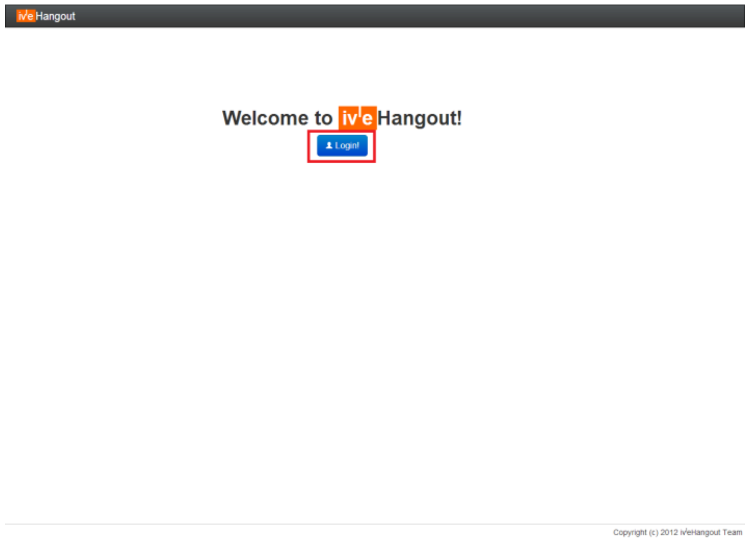
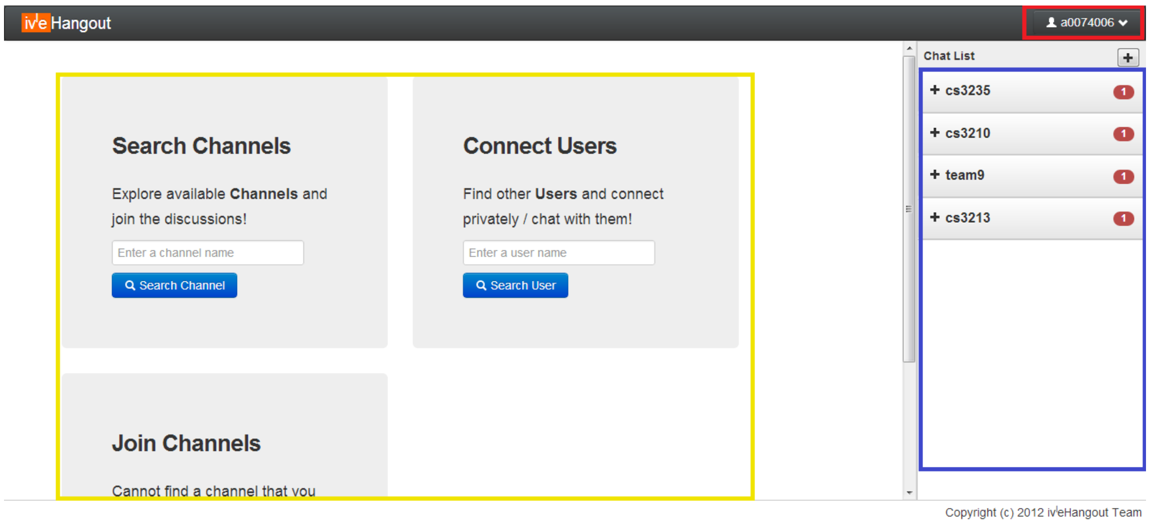


Figure 1: Login

The Dashboard - Overview

Upon successful login, user will be bought to our dashboard as shown.

1. Highlighted in **red** is a button for user to change their nickname, views their personal details or look up for help with other available command in IVLE Hangout
2. Highlighted in **blue** is the list of default channels that the user will join upon login. It is also define to be their favourite channel. Notice that there is a "**+**" beside the channel name, it define that the channel is **public**. On the other hand, a **private** channel will have a "**-**"beside its channel name.
3. Highlighted in **yellow** is some action that the user can perform in order to create a channel with a single user (private chat) or with a group of user. Details about each action will be further explained in the next few sections.



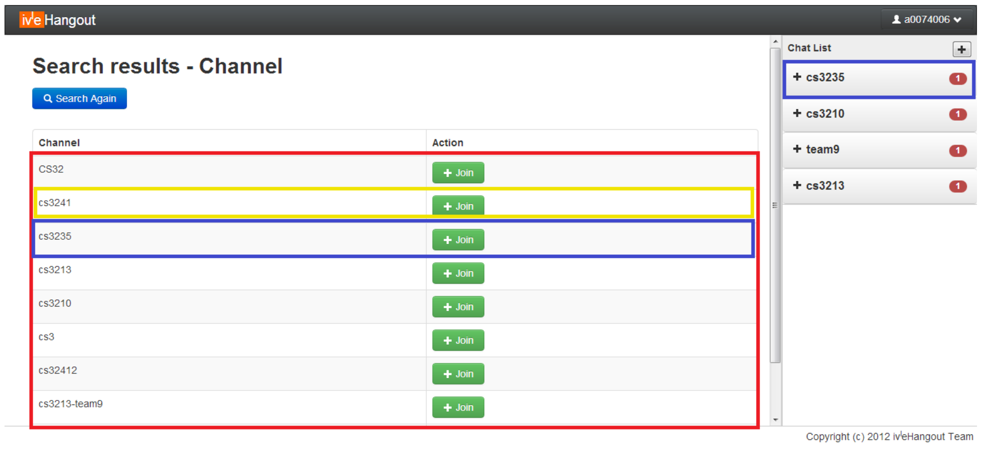
iv'e Hangout
a0074006
Search Channels
Explore available Channels and join the discussions!
Enter a channel name
Search Channel
Connect Users
Find other Users and connect privately / chat with them!
Enter a user name
Search User
Join Channels
Cannot find a channel that you
Chat List
cs3235
cs3210
team9
cs3213
Copyright (c) 2012 iv'eHangout Team

Figure 2: Dashboard

The Dashboard – Search Channel

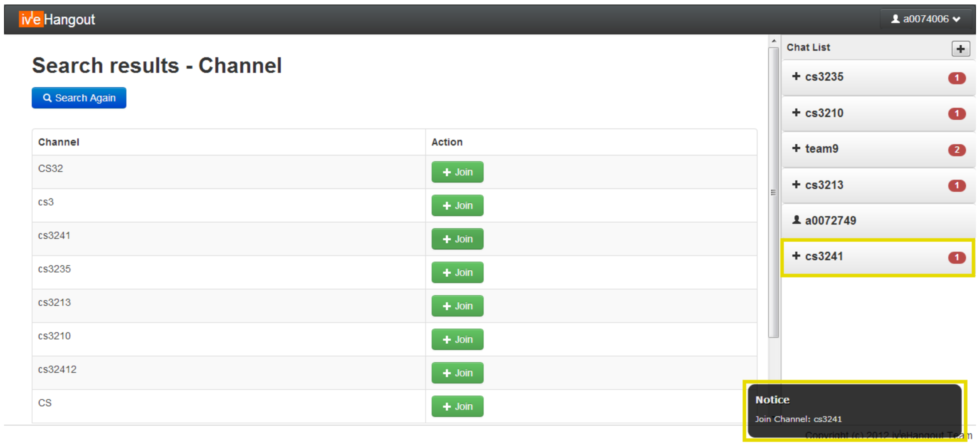
A user can search for available channel to join by typing a substring of the channel name in the "Search Channels" field.

1. Highlighted in **red** is the result of the search if it exists.
2. If a user attempt to join a channel that he/she is already in highlighted in **blue**, a notice to indicate that the user is already in the channel.
3. For example the channel that the user wish to join is highlighted in **yellow**, the user can simply click on the "Join" button to enter the channel. A notice to indicate that the action is successful will be shown in **Figure 4**.



Channel	Action
CS32	Join
cs3241	Join
cs3235	Join
cs3213	Join
cs3210	Join
cs3	Join
cs32412	Join
cs3213-team9	Join

Figure 3: Search Channel



Search results - Channel

Channel	Action
CS32	Join
cs3	Join
cs3241	Join
cs3235	Join
cs3213	Join
cs3210	Join
cs32412	Join
CS	Join

Chat List	
cs3235	1
cs3210	1
team9	2
cs3213	1
a0072749	
cs3241	1

Notice
Join Channel: cs3241

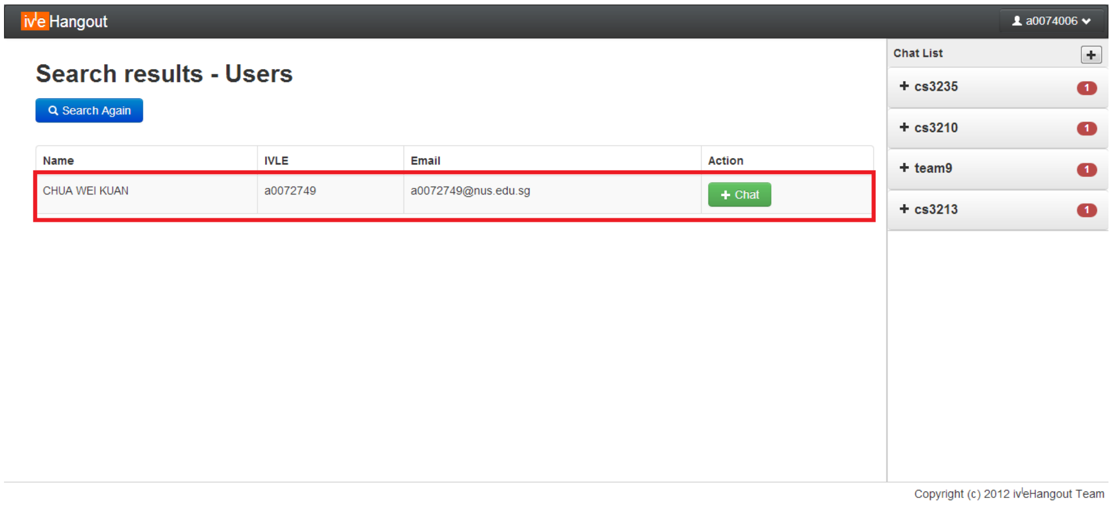
Copyright (c) 2012 ivleHangout Team

Figure 4: Join Channel

The Dashboard – Connect User

A user can search for other users by its name, IVLE ID and nickname in the "Connect Users" field.

1. Highlighted in **red** is the search result if it exists in **Figure 5**.
2. If a user attempt to private chat with another user that is currently **offline** as highlighted in **blue**, a notice to indicate that user is offline will be shown in **Figure 6**.
3. If a user attempt to private chat with another user that is currently **online** as highlighted in **yellow**, simply click on the "Chat" button to initial the private chat. A new channel will be created with the user as shown in **Figure 7**. Note that private chat the icon is a human like shape.



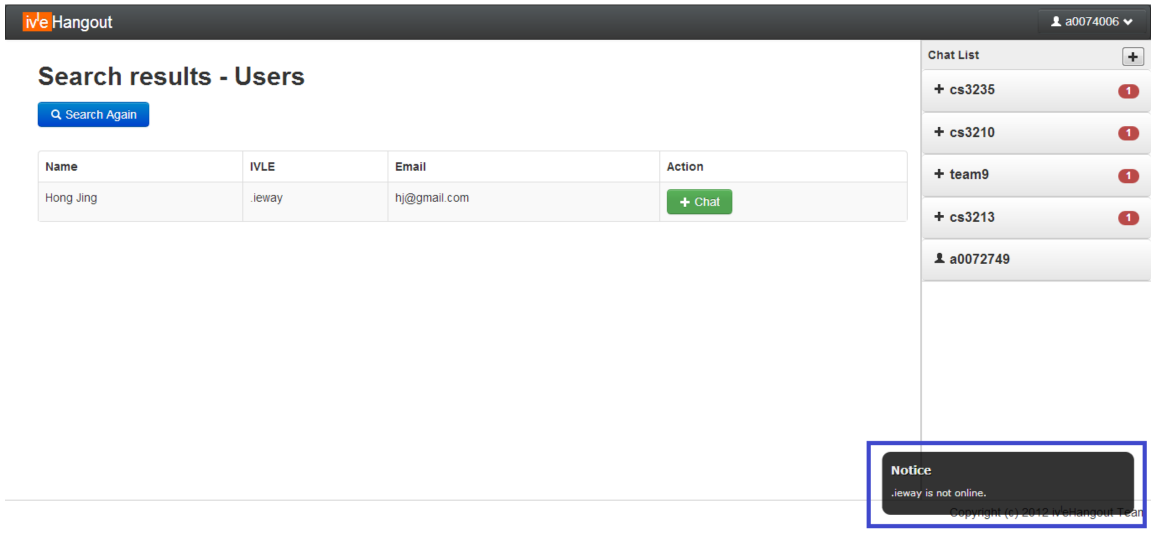
Search results - Users

Name	IVLE	Email	Action
CHUA WEI KUAN	a0072749	a0072749@nus.edu.sg	Chat

Chat List	
cs3235	1
cs3210	1
team9	1
cs3213	1

Copyright (c) 2012 ivleHangout Team

Figure 5: Search User



Search results - Users

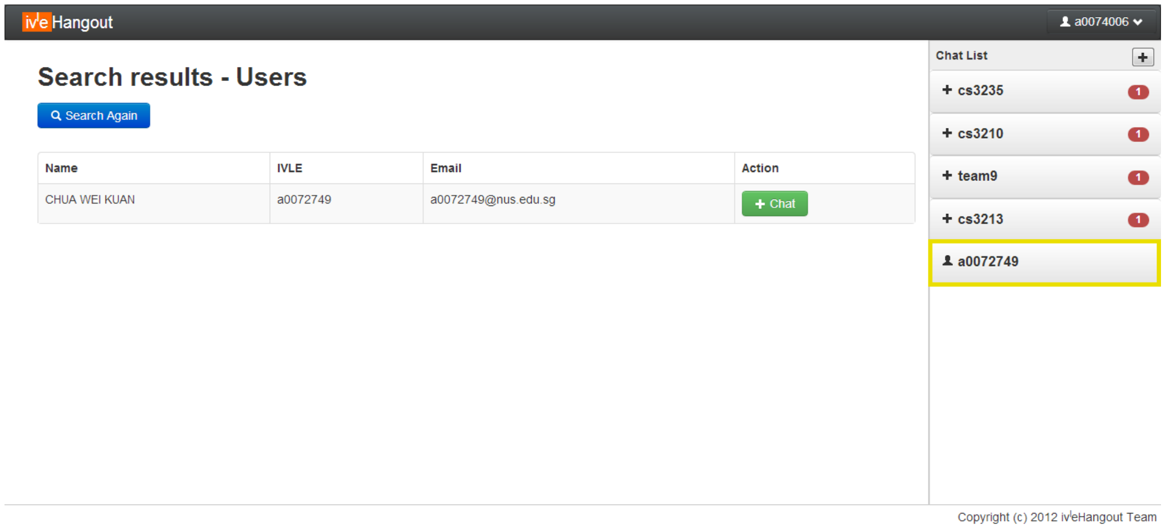
Name	IVLE	Email	Action
Hong Jing	.ieway	hj@gmail.com	Chat

Notice

.ieway is not online.

Copyright (c) 2012 ivleHangout Team

Figure 6: User offline



Search results - Users

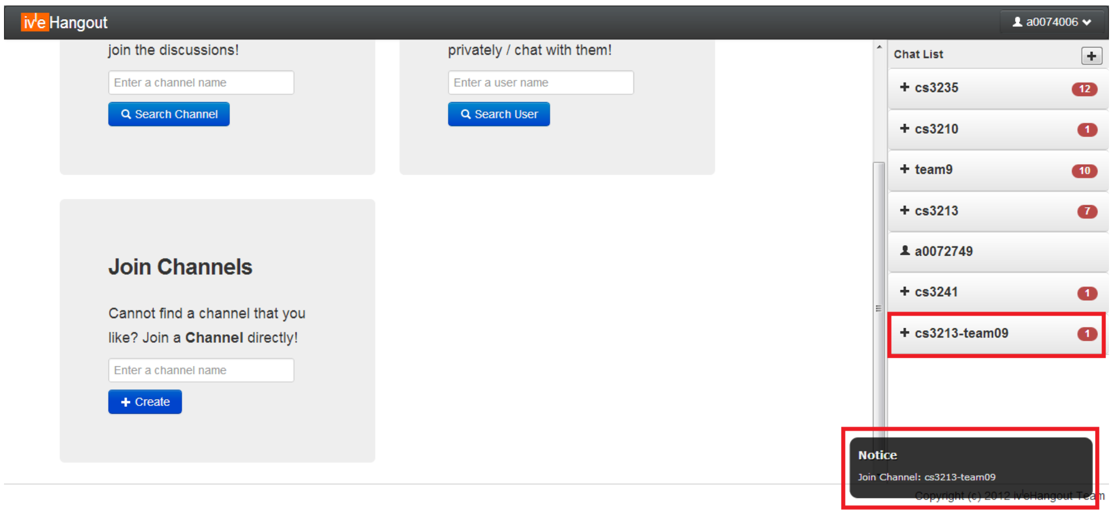
Name	IVLE	Email	Action
CHUA WEI KUAN	a0072749	a0072749@nus.edu.sg	Chat

Copyright (c) 2012 ivleHangout Team

Figure 7: Private Chat with user

The Dashboard – Create Channels

A user can create a channel that does not exist yet but filling the desire channel name in the "Create Channels" field. A notification will be shown in **Figure 8** (Highlighted in **red**) to indicate the successful creation of channel. If a user attempt to create a channel that already exist and he/she is in the channel, error notification message will be shown. On the other hand, if a user attempt to create a channel that already exists but he/she is **not** in the channel, the user will be automatically set to join the channel as shown in **Figure 4.**



join the discussions!

privately / chat with them!

Cannot find a channel that you like? Join a **Channel** directly!

Notice
Join Channel: cs3213-team09

Figure 8: Join Channel directly

The Dashboard – User's Profile

If the user clicks on the top right button that displays his/her IVLE ID, a dropdown as shown in **Figure 9** will be shown.

1. Highlighted in **red** are the details pertaining to the user.
2. Highlighted in **yellow** is the user's **nickname** and can be change by double clicking on the nickname.
3. Highlighted in **green** is the "Logout" button to logout from IVLE Hangout system as well as a "Help" button to display other available command that a user can use.

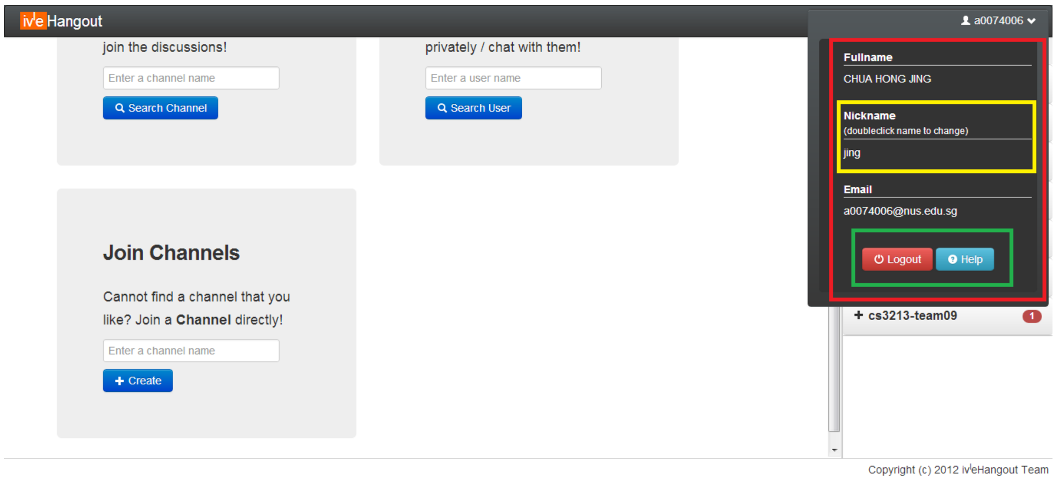


Figure 9: User Profile

The Dashboard – Where the Chat Begin

As shown in **Figure 10** is the overview of the chat area.

1. Highlighted in **red** is where the chat message will be displayed. Each chat message consist of 3 component namely the sender's nickname, the message as well as the time the message is send.
2. Highlighted in **orange** is where the user will type his/her chat message.
3. Highlighted in **yellow** is the list of user(s) that is currently in the selected channel. The currently selected channel will have its channel name at the top left hand corner.
4. Highlighted in **green** will show the number of **unread** messages for that particular channel. E.g. there are **two unread** messages in channel "team9" and "cs3235".
5. Highlighted in **blue** is the icon for actions that can be perform other than sending chat messages. **Starting from left**, the icon uniquely identify to be "**Drawing Canvas**", "**Files**", "**Settings**", "**Favourite**", "**Archive**", "**Close**", "**Channel User List**".

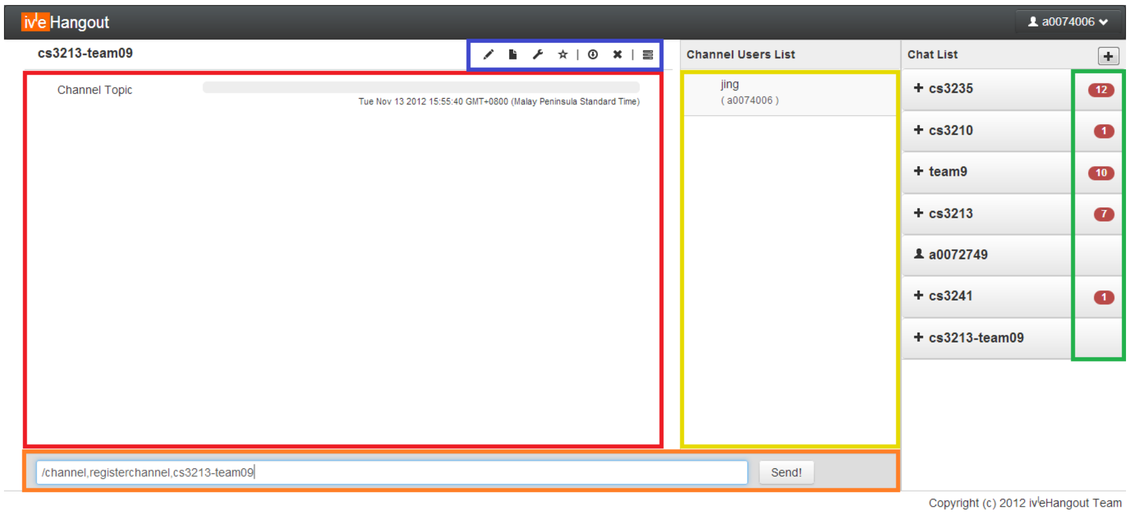
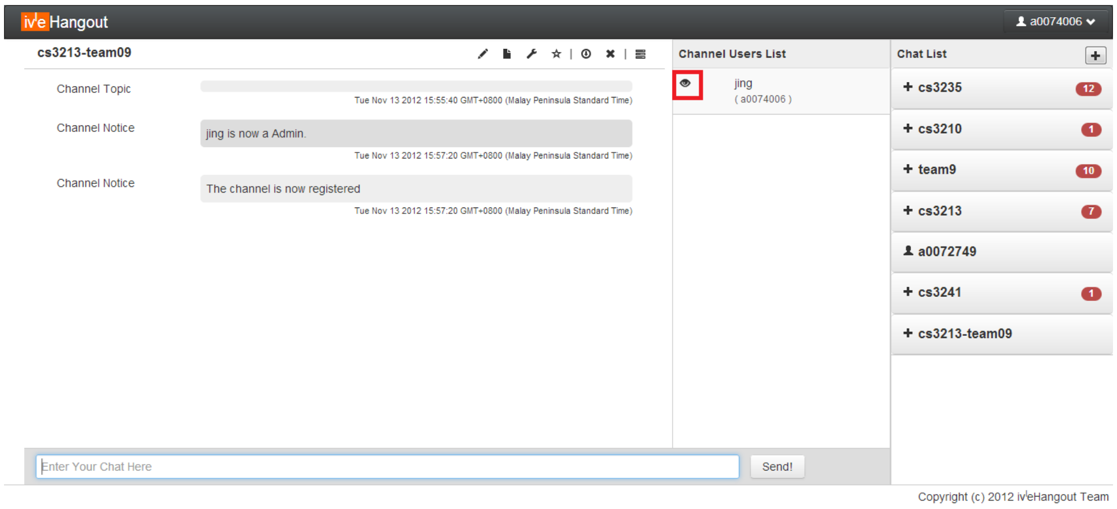


Figure 10: Channel Chat

User access levels are also reflected in the user list. The “eye” icon (**Figure 11**) indicates admin and co-admin. If it is a member of the channel, it will have a human like shape. If it is a guest/visitor of the channel, it will have no icon as shown in **Figure 10**.



iv'e Hangout — a0074006

cs3213-team09

Channel Topic
Tue Nov 13 2012 15:55:40 GMT+0800 (Malay Peninsula Standard Time)

Channel Notice — jing is now a Admin.
Tue Nov 13 2012 15:57:20 GMT+0800 (Malay Peninsula Standard Time)

Channel Notice — The channel is now registered
Tue Nov 13 2012 15:57:20 GMT+0800 (Malay Peninsula Standard Time)

Channel Users List
- jing (a0074006)

Chat List
- cs3235 12
- cs3210 1
- team9 10
- cs3213 7
- a0072749
- cs3241 1
- cs3213-team09

Enter Your Chat Here — Send!

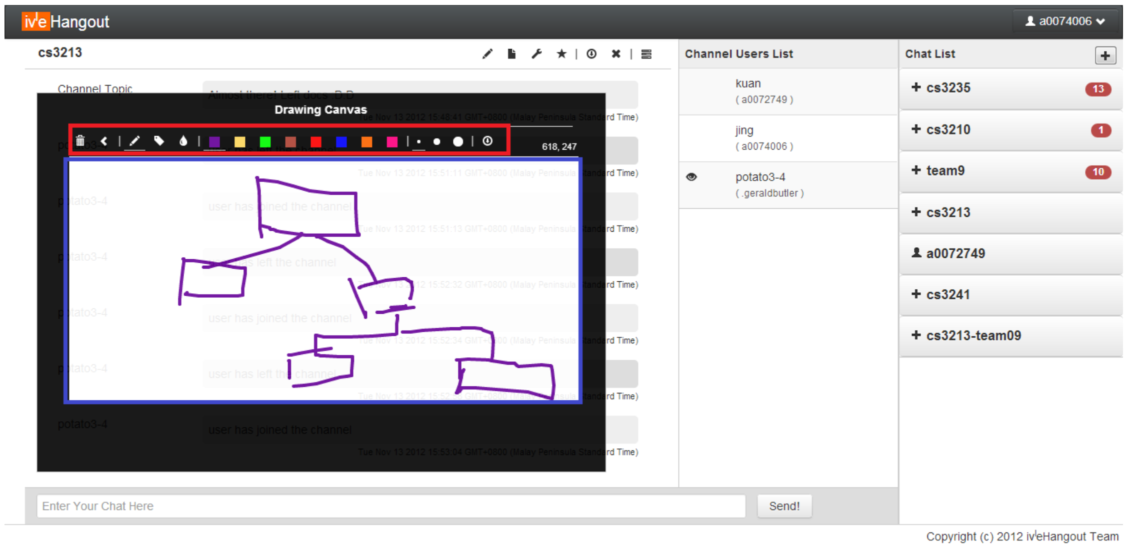
Copyright (c) 2012 iv'eHangout Team

Figure 11: Channel Access

The Dashboard – Canvas

This is a special feature that allow different user to draw on a selected area. It will display the drawing in real time across all the users in the channel. A description of **Figure 12** is as follows:

1. Highlighted in **red** is the set of option a user can choose.
2. Highlighted in **blue** is the area the user can draw.
3. Users can also download a copy of the canvas screenshot.



iv'e Hangout — a0074006

cs3213

Channel Topic — Almost there! Left docs :D:D

Drawing Canvas — 618, 247

potato3-4 — user has joined the channel
Tue Nov 13 2012 15:53:04 GMT+0800 (Malay Peninsula Standard Time)

Channel Users List
- kuan (a0072749)
- jing (a0074006)
- potato3-4 (.geraldbutler)

Chat List
- cs3235 13
- cs3210 1
- team9 10
- cs3213
- a0072749
- cs3241
- cs3213-team09

Enter Your Chat Here — Send!

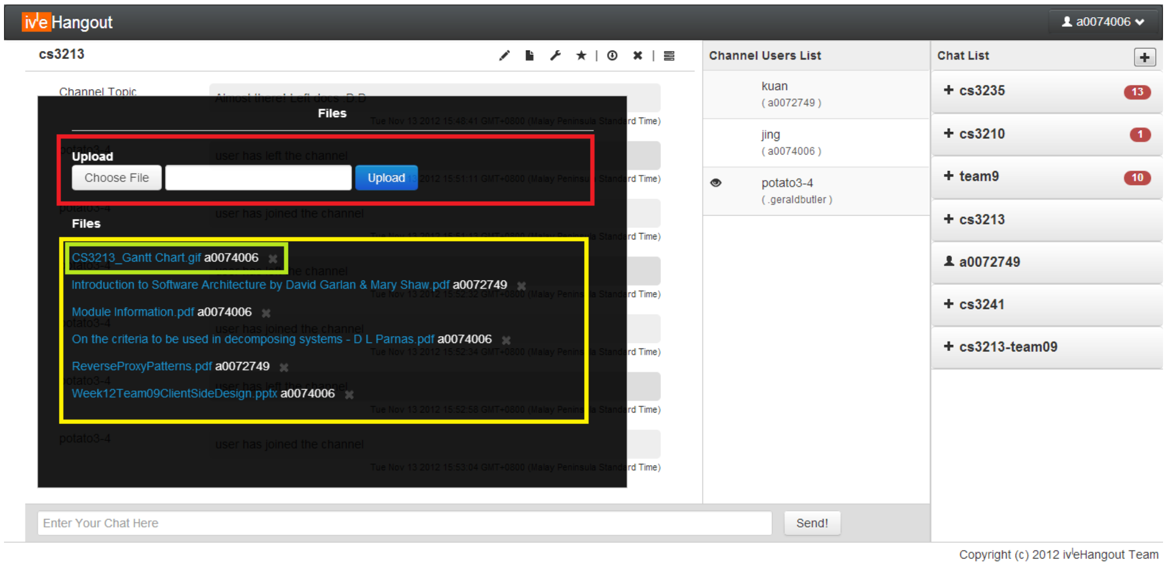
Copyright (c) 2012 iv'eHangout Team

Figure 12: Canvas

The Dashboard – Files in a Channel

This is another special feature that allows different user to shares files within the channel. User will be able to upload his/her files as well as see all other files that are uploaded by other user of that channel. A description of **Figure 13** is as follows:

1. Highlighted in **red** is where the user can choose which file to upload by clicking on the “Choose File” button. A regular window file selection pop up will appear. After which, the user can click on the “Upload” button to upload the file to server.
2. Highlighted in **yellow** is where the files pertaining to the current channel are displayed.
3. Highlighted in **green** contained information such as the file name, the user id that uploaded the files and an “x” button to delete the file.



CS3213_Gantt Chart.gif a0074006
Introduction to Software Architecture by David Garlan & Mary Shaw.pdf a0072749
Module Information.pdf a0074006
On the criteria to be used in decomposing systems - D L Parnas.pdf a0074006
ReverseProxyPatterns.pdf a0072749
Week12Team09ClientSideDesign.pptx a0074006

kuan (a0072749)
jing (a0074006)
potato3-4 (.geraldbutler)

cs3235	13
cs3210	1
team9	10
cs3213	
a0072749	
cs3241	
cs3213-team09	

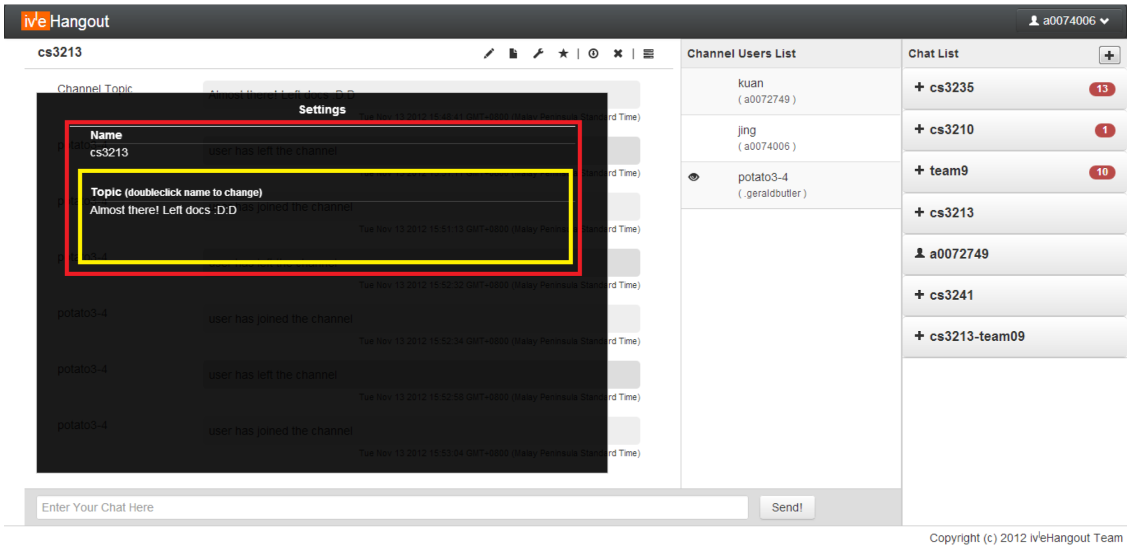
Copyright (c) 2012 iv'eHangout Team

Figure 13: Document Repository

The Dashboard – Settings of a Channel

Allow the user to view the topic as well as changing the topic (if permission granted by channel admin). A description of **Figure 14** is as follows:

1. Highlighted in **yellow**, is where the user can change the topic pertaining to the channel.
2. Highlighted in **red**, is shows the overall information pertaining to the channel. (More details can be added in the future such as Channel admin/co-admin as well as Channel access level.



iv'e Hangout
a0074006
cs3213
Channel Topic
Almost there! Left docs :D:D
Settings
Name
cs3213
Topic (doubleclick name to change)
Almost there! Left docs :D:D
Tue Nov 13 2012 15:48:41 GMT+0800 (Malay Peninsula Standard Time)
potato3-4
user has left the channel
user has joined the channel
Tue Nov 13 2012 15:51:13 GMT+0800 (Malay Peninsula Standard Time)
Tue Nov 13 2012 15:52:32 GMT+0800 (Malay Peninsula Standard Time)
potato3-4
user has joined the channel
Tue Nov 13 2012 15:52:34 GMT+0800 (Malay Peninsula Standard Time)
potato3-4
user has left the channel
Tue Nov 13 2012 15:52:58 GMT+0800 (Malay Peninsula Standard Time)
potato3-4
user has joined the channel
Tue Nov 13 2012 15:53:04 GMT+0800 (Malay Peninsula Standard Time)
Enter Your Chat Here
Send!
Channel Users List
kuan (a0072749)
jing (a0074006)
potato3-4 (.geraldbutler)
Chat List
cs3235 13
cs3210 1
team9 10
cs3213
a0072749
cs3241
cs3213-team09
Copyright (c) 2012 iv'eHangout Team

Figure 14: Channel Settings

The Dashboard – Favorite (Bookmark) a Channel

This feature allows the user to set a particular channel as default/auto join channel upon login. A description of **Figure 15** is as follows:

1. Highlighted in **red**, is the star icon that is shaded indicates that the channel is set to be a default/auto join channel. A notification will be shown as highlighted in **red** at the bottom right of the screen. If the star icon is not shaded, it indicates that the channel is either not set to be default/auto join channel or it has been removed from the user default/auto join channel.

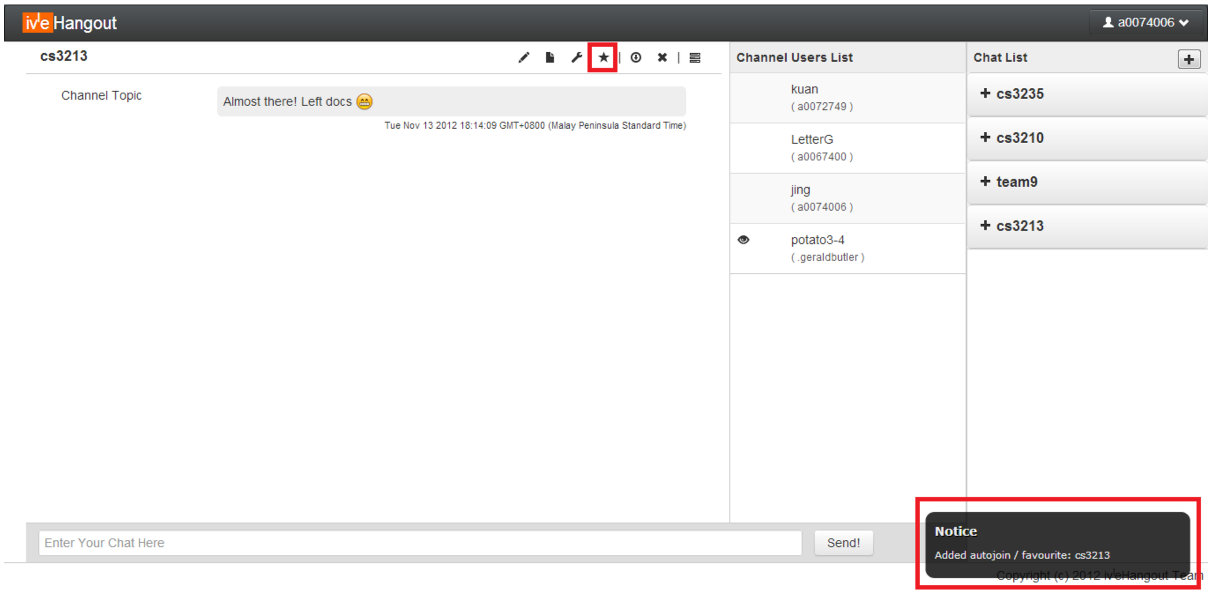
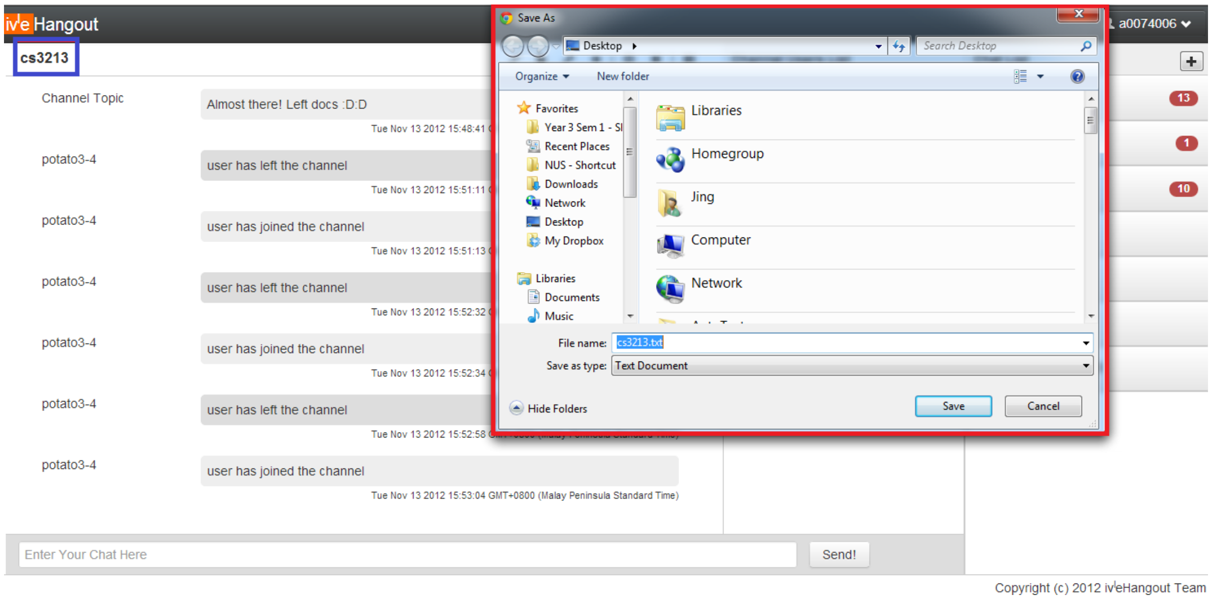

iv/e Hangout
a0074006
cs3213
Channel Topic
Almost there! Left docs 😁
Tue Nov 13 2012 18:14:09 GMT+0800 (Malay Peninsula Standard Time)
Channel Users List
kuan (a0072749)
LetterG (a0067400)
jing (a0074006)
potato3-4 (.geraldbutler)
Chat List
+ cs3235
+ cs3210
+ team9
+ cs3213
Enter Your Chat Here
Send!
Notice
Added autojoin / favourite: cs3213
Copyright (c) 2012 iv/eHangout Team

Figure 15: Favorites

The Dashboard – Saving the Chat Log

This is a chat log function that allows the user to save the chat history pertaining to the current selected channel/private chat. A description of **Figure 16** is as follows:

1. Highlighted in **red**, is a windows pop up file selection. This is where the user can choose to save its chat history file.
2. Highlighted in **blue** is the channel name, the name of the file will be automatically set to be the channel name. Nonetheless, the user is allowed to change the file name.

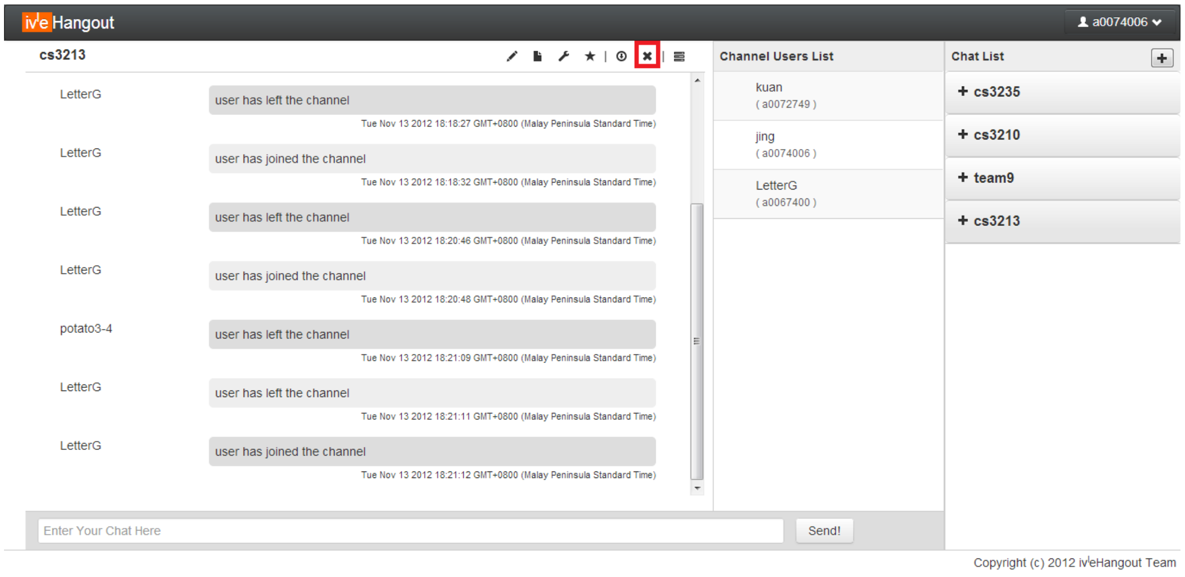

iv/e Hangout
a0074006
cs3213
Channel Topic
Almost there! Left docs :D:D
Tue Nov 13 2012 15:48:41
potato3-4
user has left the channel
Tue Nov 13 2012 15:51:11
potato3-4
user has joined the channel
Tue Nov 13 2012 15:51:13
potato3-4
user has left the channel
Tue Nov 13 2012 15:52:32
potato3-4
user has joined the channel
Tue Nov 13 2012 15:52:34
potato3-4
user has left the channel
Tue Nov 13 2012 15:52:58
potato3-4
user has joined the channel
Tue Nov 13 2012 15:53:04 GMT+0800 (Malay Peninsula Standard Time)
Enter Your Chat Here
Send!
Copyright (c) 2012 iv/eHangout Team

Save As
Desktop
Search Desktop
Organize
New folder
Favorites
Year 3 Sem 1 - S
Recent Places
NUS - Shortcut
Downloads
Network
Desktop
My Dropbox
Libraries
Documents
Music
Libraries
Homegroup
Jing
Computer
Network
File name: cs3213.txt
Save as type: Text Document
Hide Folders
Save
Cancel

Figure 16: Channel Chat Archive

The Dashboard – Leaving a Channel

This is to leave the current channel to stop receiving messages from the channel. User can simply click on the button highlighted in **red** in **Figure 17** to leave the channel.



LetterG — user has left the channel — Tue Nov 13 2012 18:18:27 GMT+0800 (Malay Peninsula Standard Time)

LetterG — user has joined the channel — Tue Nov 13 2012 18:18:32 GMT+0800 (Malay Peninsula Standard Time)

LetterG — user has left the channel — Tue Nov 13 2012 18:20:46 GMT+0800 (Malay Peninsula Standard Time)

LetterG — user has joined the channel — Tue Nov 13 2012 18:20:48 GMT+0800 (Malay Peninsula Standard Time)

potato3-4 — user has left the channel — Tue Nov 13 2012 18:21:09 GMT+0800 (Malay Peninsula Standard Time)

LetterG — user has left the channel — Tue Nov 13 2012 18:21:11 GMT+0800 (Malay Peninsula Standard Time)

LetterG — user has joined the channel — Tue Nov 13 2012 18:21:12 GMT+0800 (Malay Peninsula Standard Time)

Channel Users List: kuan (a0072749), jing (a0074006), LetterG (a0067400)

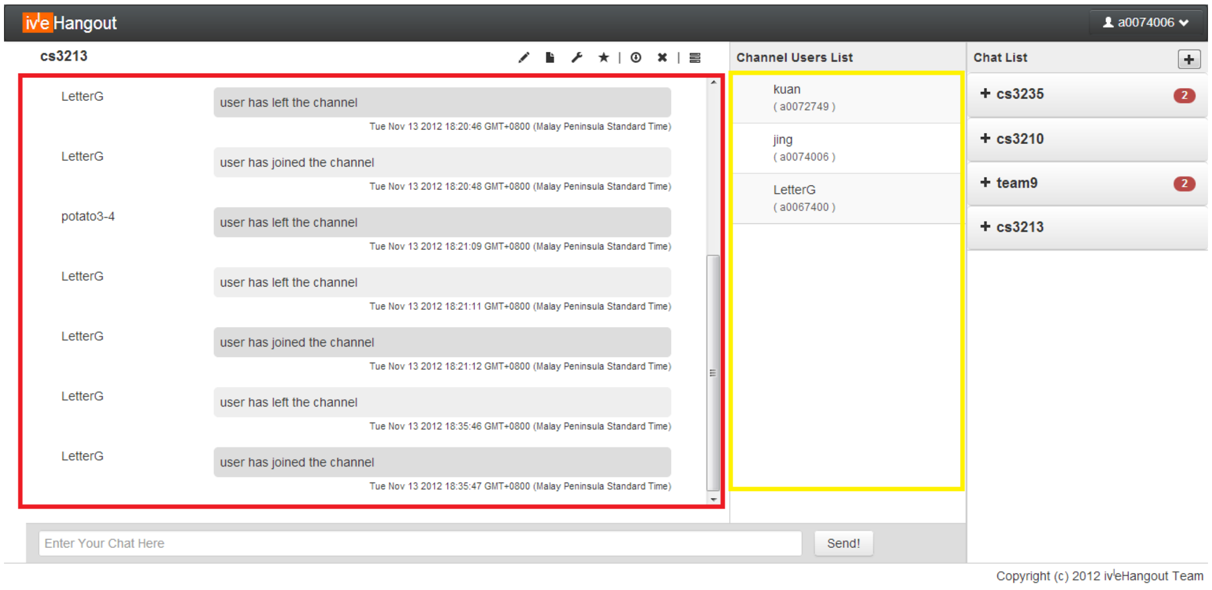
Chat List: cs3235, cs3210, team9, cs3213

Copyright (c) 2012 iv'eHangout Team

Figure 17: Leaving a Channel

The Dashboard – Channel User List

Upon clicking this icon, user is able to minimum the user list (highlighted in **yellow**) and at the same time maximizing the chat area (highlighted in **red**).



LetterG — user has left the channel — Tue Nov 13 2012 18:20:46 GMT+0800 (Malay Peninsula Standard Time)

LetterG — user has joined the channel — Tue Nov 13 2012 18:20:48 GMT+0800 (Malay Peninsula Standard Time)

potato3-4 — user has left the channel — Tue Nov 13 2012 18:21:09 GMT+0800 (Malay Peninsula Standard Time)

LetterG — user has left the channel — Tue Nov 13 2012 18:21:11 GMT+0800 (Malay Peninsula Standard Time)

LetterG — user has joined the channel — Tue Nov 13 2012 18:21:12 GMT+0800 (Malay Peninsula Standard Time)

LetterG — user has left the channel — Tue Nov 13 2012 18:35:46 GMT+0800 (Malay Peninsula Standard Time)

LetterG — user has joined the channel — Tue Nov 13 2012 18:35:47 GMT+0800 (Malay Peninsula Standard Time)

Channel Users List: kuan (a0072749), jing (a0074006), LetterG (a0067400)

Chat List: cs3235 2, cs3210, team9 2, cs3213

Copyright (c) 2012 iv'eHangout Team

Figure 18: Channel User List