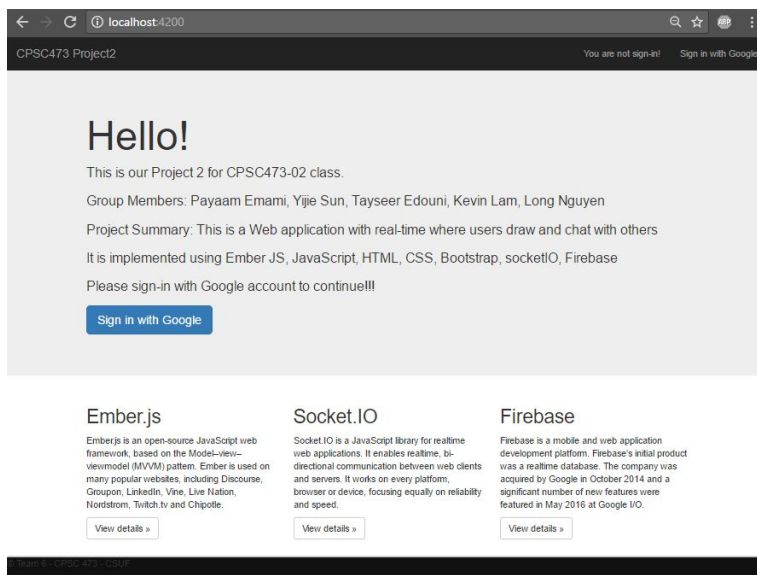


# Project 2

## Documentation



By:

**Team 6**

Payaam Emami

Yijie Sun

Tayseer Edouni

Kevin Lam

Long Nguyen

Submit date: May 15, 2017.

## Table of Contents

<b>1. Introduction</b>	<b>3</b>
1.1 Purpose and Scope	3
1.2 System Description	
<b>2. Installation/Configuration</b>	<b>4</b>
2.1 Download	4
2.2 Running	5
<b>3. Application Architecture and Workflow</b>	<b>6</b>
<b>4. User Manual</b>	<b>8</b>
4.1 Main Page	8
4.2 Log-in with Google account	9
4.3 Drawing with Socket.IO	10
4.4 Chat with Firebase	12
4.5 Firebase Database	13

# 1. Introduction

## 1.1 Purpose and Scope

This application is a fun real-time drawing/guessing game where users can draw onto the canvas in real time while other users can guess what the drawing is using the chat. The first prerequisite is that users login through google login (using their web service api).

## 1.2 System Description

This web application allows users to draw and chat with others in real time. It is implemented using Ember.JS, JavaScript, HTML, CSS, Bootstrap, socket.IO, and Firebase.

Main functions:

- Login with Google Account
- Real-time drawing with Socket.IO
- Real-time chat with Firebase

## 2. Installation/Configuration

### 2.1 Download

#### Git

Install git if it is not already installed

Then in terminal:

- ***git clone [https://github.com/PayaamEmami/CPSC\\_473\\_Project\\_2](https://github.com/PayaamEmami/CPSC_473_Project_2)***

#### Node.js

Make sure you have Node.js installed, and you are using the latest version of Node.js (>0.12.0). You can check your version with the terminal command

- ***node --version***

#### Ember.js

Install Ember CLI if it is not already installed

- ***npm install -g ember-cli@2.4***

#### Bower and Watchman

Next, install Bower, another asset management tool.

- ***npm install -g bower***
- ***npm -g install fb-watchman***
- ***npm -g install watchman***

### 2.2 Running

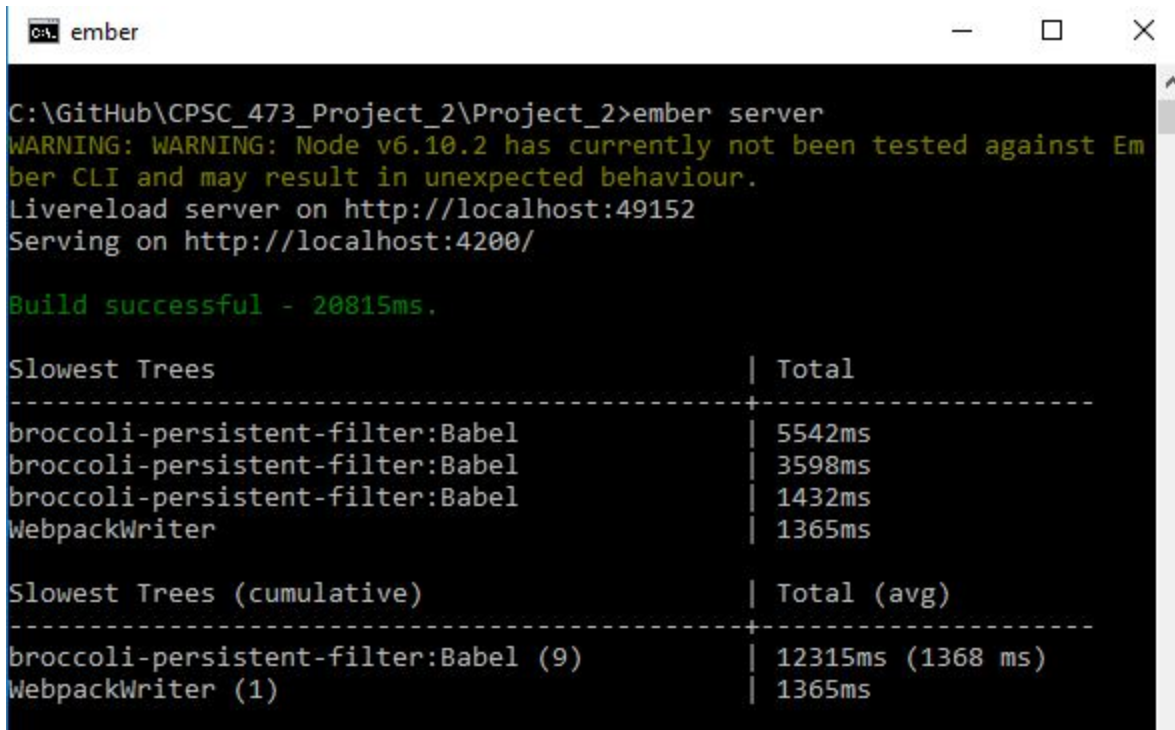
Installation: From the project directory you download (CPSC\_473\_Project\_2):

- ***cd Project\_2 (if not already in Project\_2 directory)***
- ***npm install***
- ***bower install***

Running:

- *ember server*
- *Visit the app at <http://localhost:4200>.*

Note: Installation only need for the first time after download from GitHub (npm install, bower install). After that, it can run by using ‘ember server’ command.



```
C:\GitHub\CPSC_473_Project_2\Project_2>ember server
WARNING: WARNING: Node v6.10.2 has currently not been tested against Em
ber CLI and may result in unexpected behaviour.
Livereload server on http://localhost:49152
Serving on http://localhost:4200/

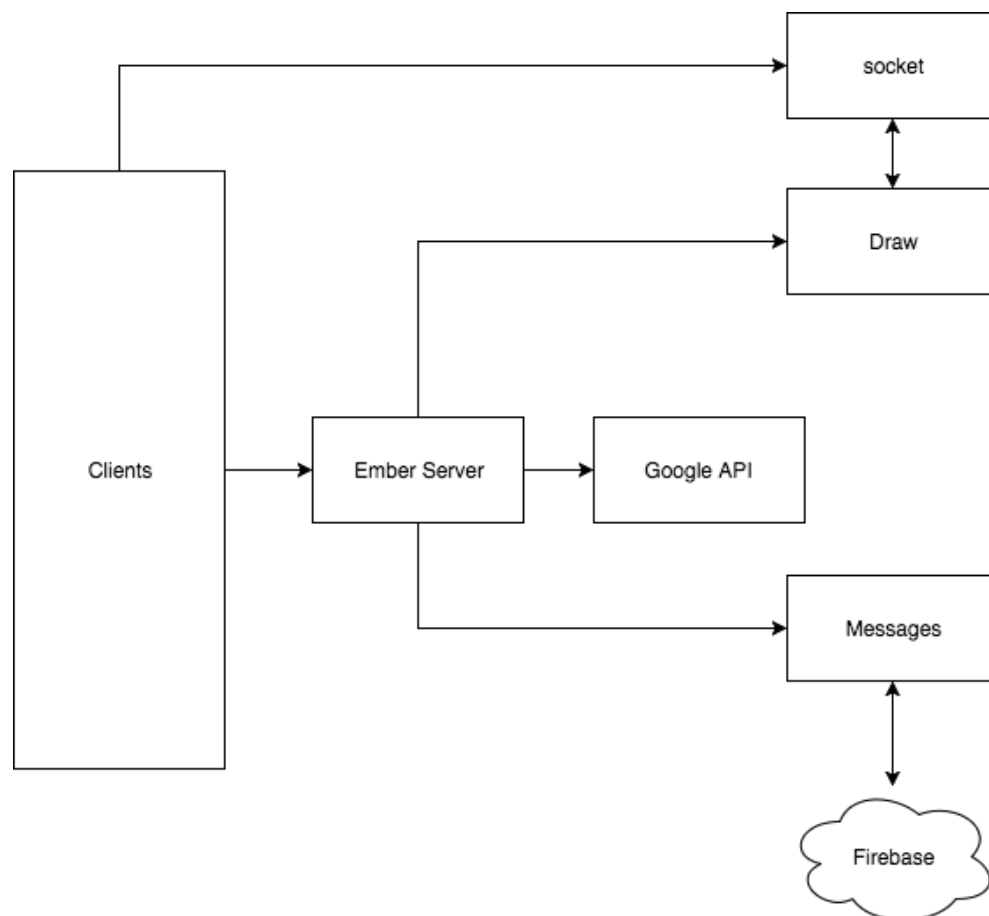
Build successful - 20815ms.

Slowest Trees | Total
-----+-----
broccoli-persistent-filter:Babel | 5542ms
broccoli-persistent-filter:Babel | 3598ms
broccoli-persistent-filter:Babel | 1432ms
WebpackWriter | 1365ms

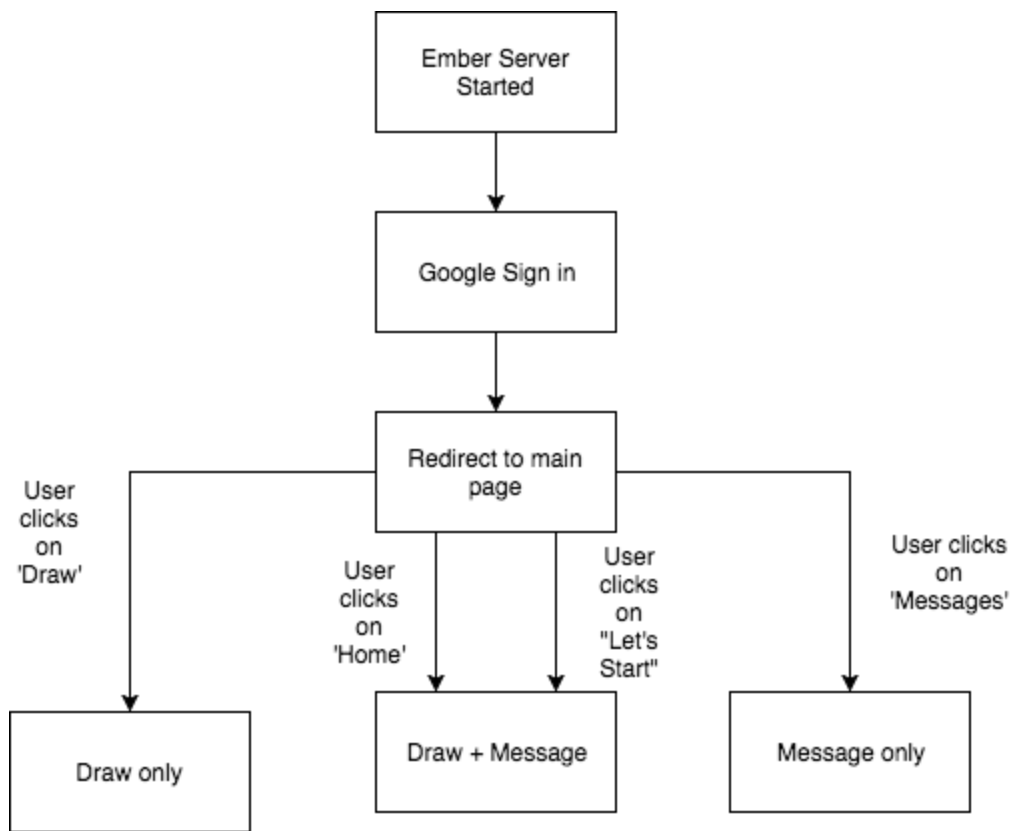
Slowest Trees (cumulative) | Total (avg)
-----+-----
broccoli-persistent-filter:Babel (9) | 12315ms (1368 ms)
WebpackWriter (1) | 1365ms
```

### 3. Application Architecture and Workflow

This project runs on an Ember Server. Ember is a Model-View-Controller framework. The clients will initiate the ember server (run 'ember server' in terminal within root directory of 'tracker') then will need to login using their Google account. The user cannot progress in using the application without logging in. Once the user logs in, Ember will reroute them back to the same page but now the browser will show that the client is logged in and will provide options such as "Let's Start", "Home", "Draw", "Messages". When the user click's on "Home" (main application), the user will be rerouted to localhost:4200/home by Ember. On this page there were will be two components (Draw and Messages). Draw will open a socket connection with the client (this way if other clients join they can see what each other are drawing). The messages component connects to Google's Firebase (therefore there's no need for another socket). Each client's browser will send data and retrieve data from Google's Firebase server.



**Figure 1:** Architecture



**Figure 2: Workflow of Application**

## 4. User Manual

### 4.1 Main Page

When the user first loads the webpage (localhost:4200), they are prompted to sign in with Google. The user cannot navigate to the drawing/messages portion of the project without logging. The index page gives a brief introduction to the project (group members and technologies used).

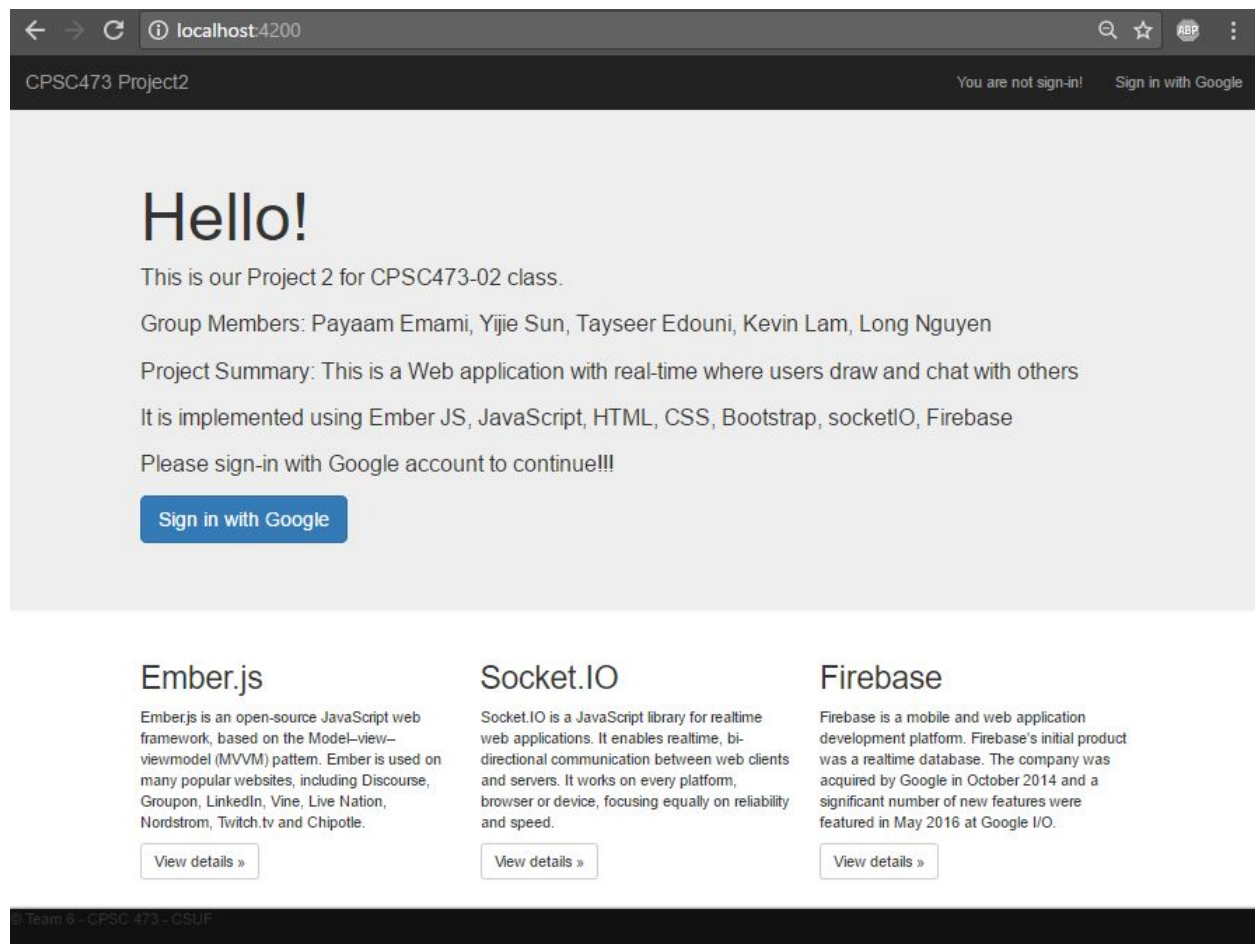
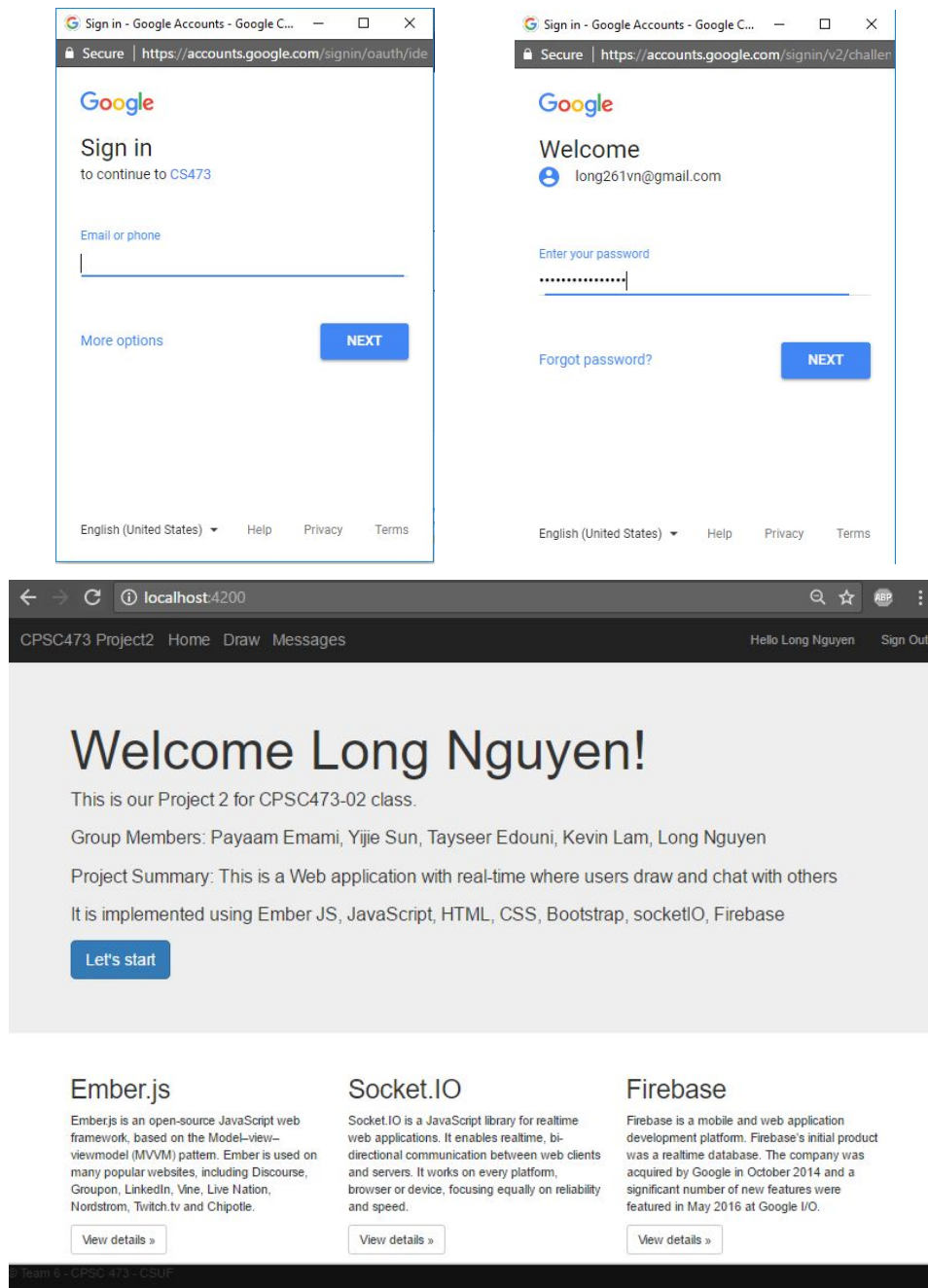


Figure 3: Main Page



## 4.2 Log-in with Google account

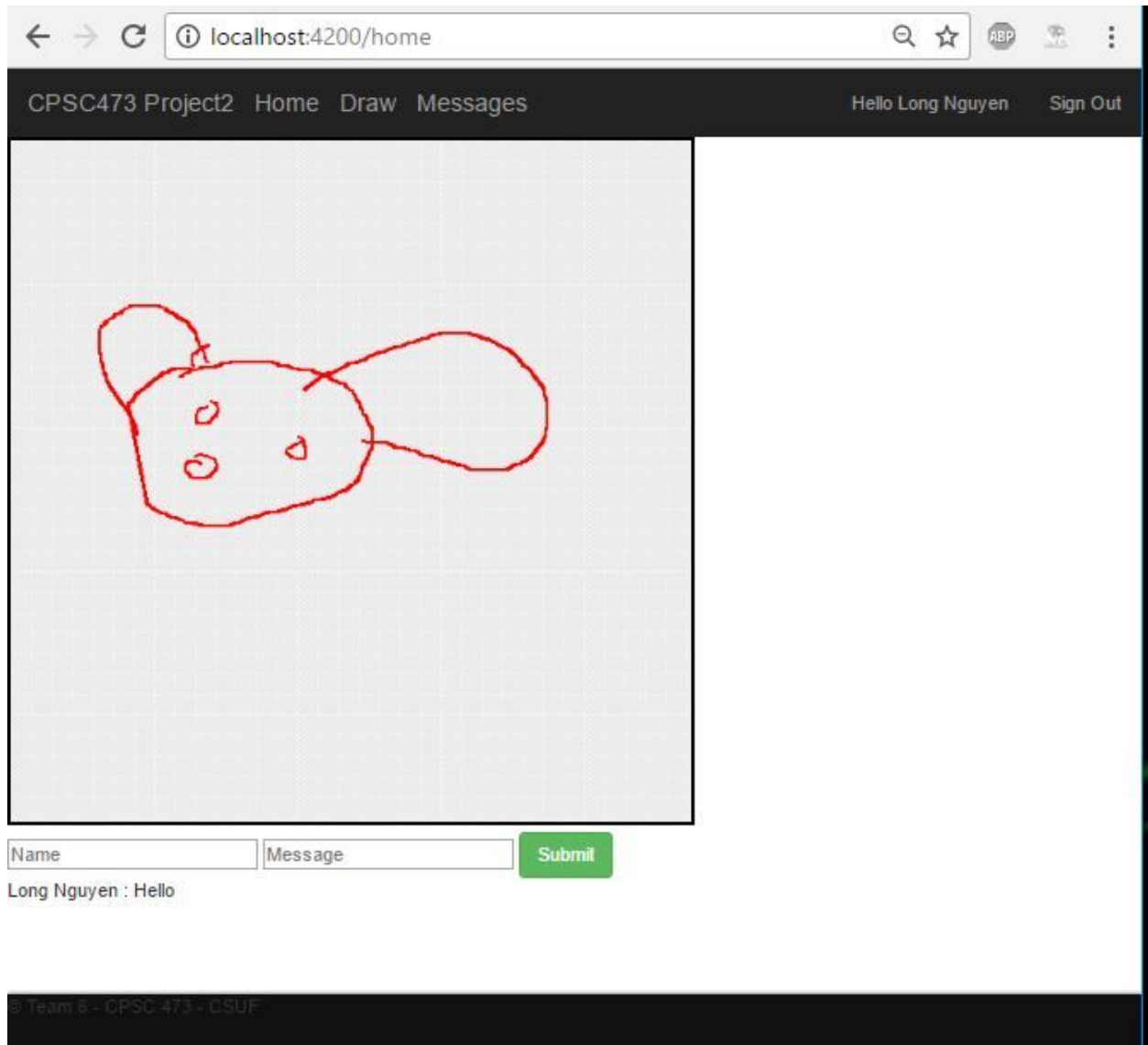
After the user has signed in via google they will be redirected to the same landing page however with noticeable changes signifying that the user is logged in. The user can click on “Let’s start” or “Home” and will be navigated to the drawing web page with the chat server.



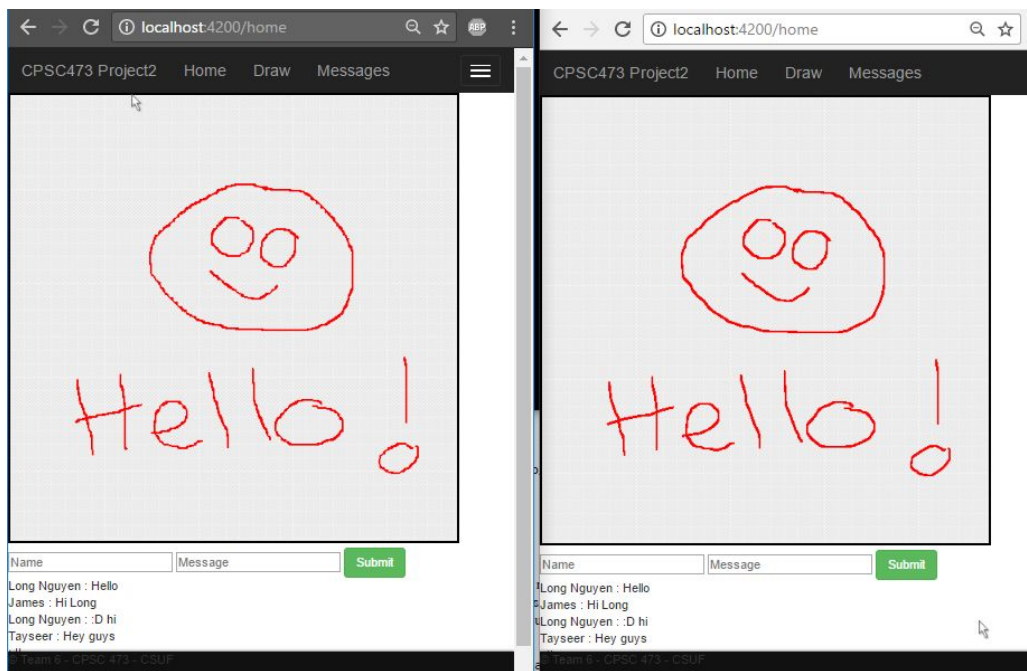
**Figure 4:** After User logs in via Google

## 4.3 Drawing with Socket.IO

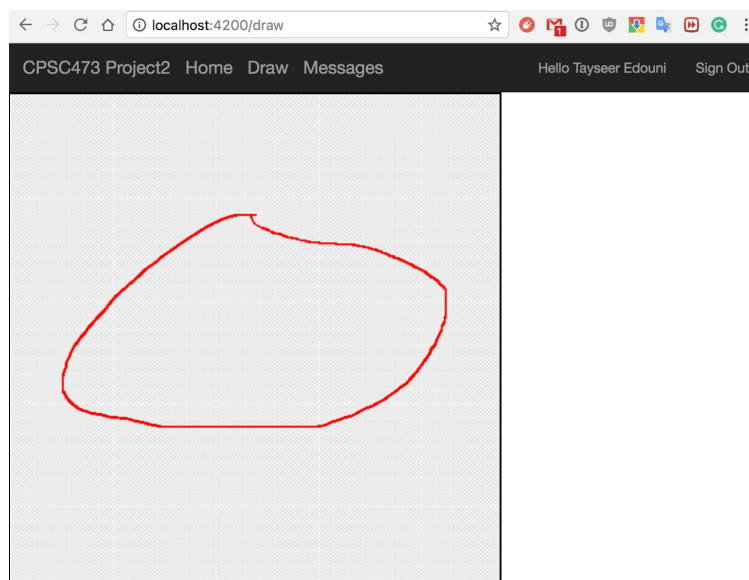
Users can either opt to draw with the messages server or can chose to draw without the chat server. First the user needs to be logged in via Google. Then the user can click on “Home” or “Let’s Start” to be redirected to localhost:4200/home which contains the drawing canvas and the chat server on the same page as shown in



**Figure 5:** Drawing Canvas and Chat server on same page



If users chose to click on “Draw” they will be redirected to localhost:4200/draw and will only see the drawing canvas on the web page.



Canvas drawing

**Figure 6:** localhost:4200/draw

## 4.4 Chat with Firebase

Similar to the drawing aspect to the project, users have the option of drawing and chatting at the same time or have the option to only chat. If users select “Home” or “Let’s Start”, they will be redirected to the localhost:4200/home page which is seen in Figure 3. If the user selects “Messages”, they will be redirected to localhost:4200/messages and will only be able to chat as shown in Figure 7.

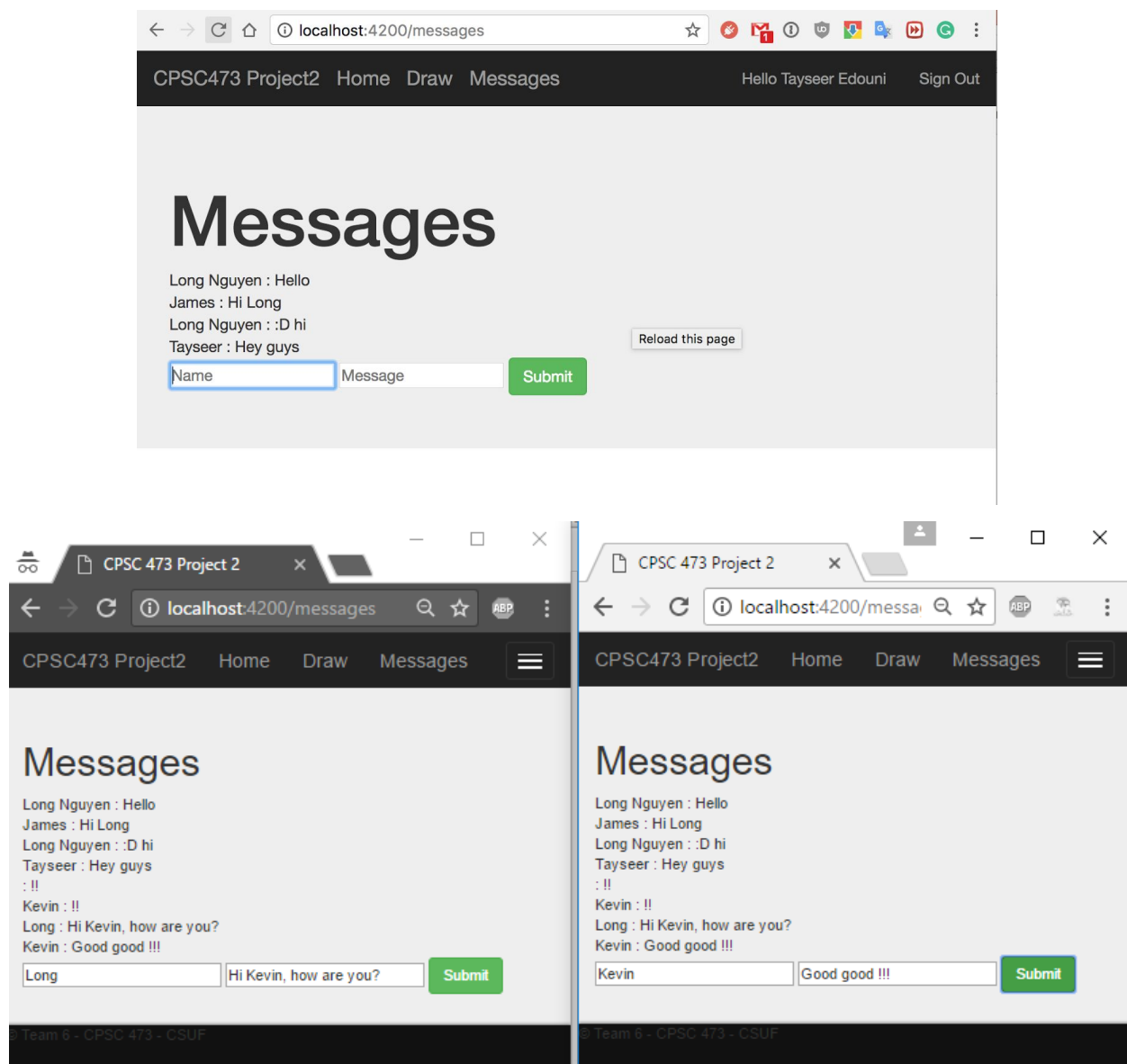
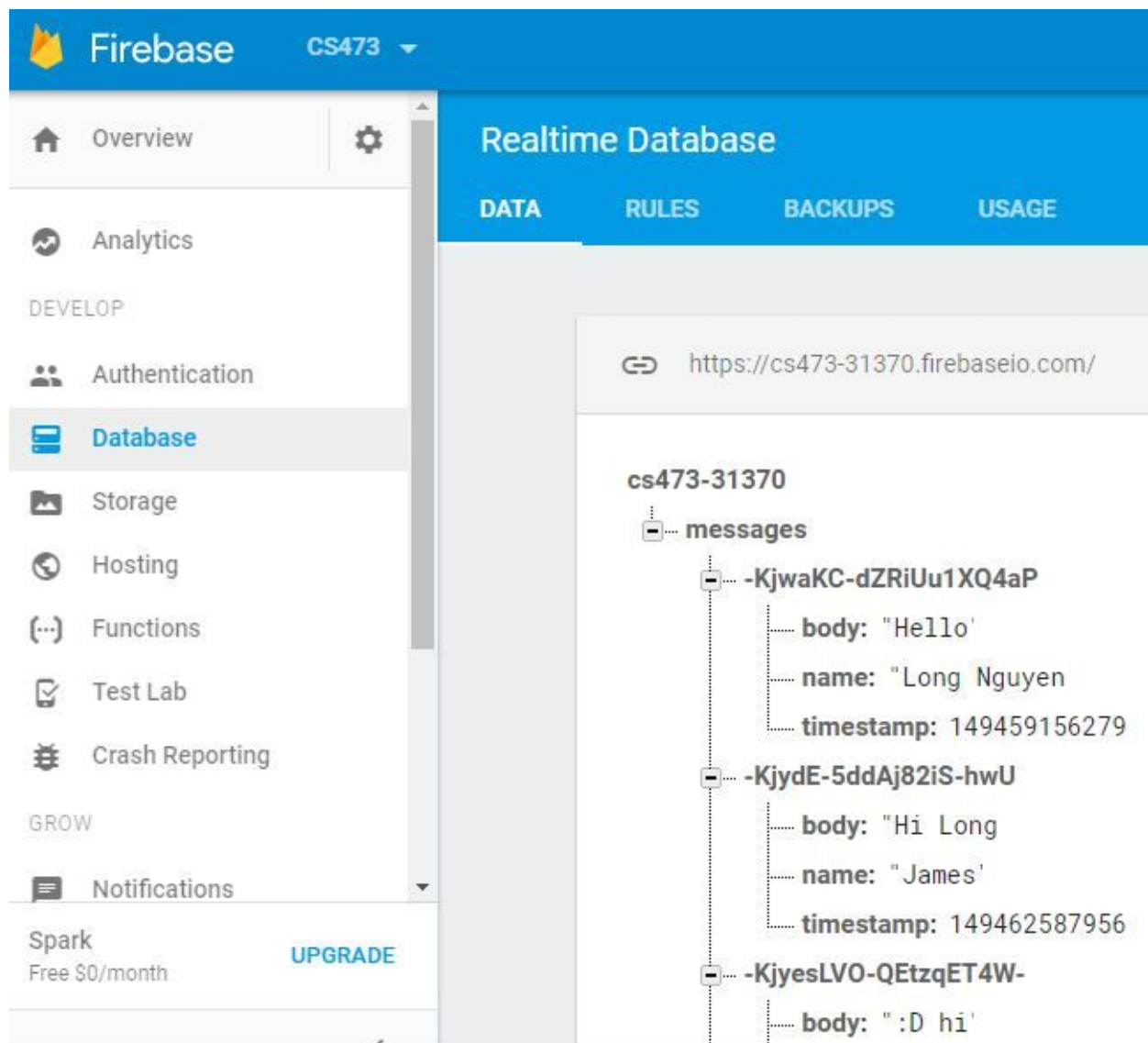


Figure 7: localhost:4200/messages

## 4.5 Firebase Database

The Firebase Realtime Database is a cloud-hosted NoSQL database. Data is stored as JSON and synchronized in realtime to every connected client. All clients share one Realtime Database instance and automatically receive updates with the newest data.



**Figure 8:** Firebase database