

# “Sketchy” App Documentation v1.0

CMPT276 - Group NOIDEA [github.com/engichang1467/Sketchy](https://github.com/engichang1467/Sketchy) - UI BRANCH

<https://sketchy-app-production.herokuapp.com/>

Benjamin Katz, Zhifan Shen, Favian (Ian) Samatha, Daisy Xu, Michael Chang

---

## Project Abstract

Sketchy is an easy-to-play multiplayer browser game designed for anyone who enjoys drawing and guessing words. Built with education in mind, Sketchy also displays descriptions, images, and translations for each word. Players can also use different word bank presets if they’d like to focus on a certain subject. Users can create private or public rooms and invite others to join a game using a link. One game consists of several rounds. Each round, one player will draw their chosen word, and the other players will guess what the word is. The faster you guess, the more points you receive!

## Customer

Sketchy’s customers are those who have previously played casual online browser games or are looking for an alternative to similar existing games.

## Competitive Analysis

The most well known multiplayer drawing game is skribbl.io, but there are many other alternatives that have more features (the original developer quit so there have been no updates). The success of Skribbl.io is partly due to its simplicity, but the lack of certain features (like an “Undo” button) makes playing the game frustrating at some times. Further, there are many words that people do not understand and cannot draw. This leads to rounds where the drawer simply writes out the word using the brush, which isn’t very fun. Sketchy solves this by including a description and photo of the word to help people learn.

## Admin Account Details

username: master & password: imurfather

## Iteration 1 Stories

For iteration 1, our group focused on the fundamental features of the application that will lay the foundation for the multiplayer and game-logic stories that will be added in iteration 2.

### Actors:

- Players: These are regular users who wish to play the game
- Admin: This is a superior user who can view all users and their respective usernames and passwords

Iteration	Story Name	Actor(s)	Trigger(s)	Action(s)	Test Description / Expectations	Test Function
1	Player Login	Player	Player is on the application home page, and is currently logged out.	Player types in their username and password and clicks “login” to authenticate their information.	If I enter my username and correct password, the user status should be logged in, and they will be able to join one of the game rooms. If the password is incorrect, the page should ask me to re-enter the password.	<b><u>Inside “login-signup.js”</u></b> <code>It('Join game room: return successful(code 200) if user is logged in')</code>
1	Group Chat	Player	Player is in a chat room.	Player types a message in an input box and clicks a send button to broadcast this message to everyone in the lobby.	When a player clicks the send button, the chat message HTML should be appended to the chat message tag.	<b><u>Inside “socket-test.js”</u></b> <code>It('Users can send messages within the same gameroom')</code>

1	Registration	Player	New / logged out user is on the application home page and wishes to create an account for the application.	Player clicks “sign up” after entering the username and password to create an account in the user database and use the application.	If I enter a username and password, and the username is unique, it should create an account and direct me to the application home page. If the username has already been taken, the registration will fail, and it will redirect the user back to the registration page.	<b><u>Inside “login-signup.js”</u></b> <pre>it('Unique username: return error code 302 when name has already been taken')</pre> <pre>it('Log in: successful if and only if registration succeeded')</pre>
1	View Database	Admin	Admin logs in and wants to view list of players	When an admin logs in, they will be able to see an option to view the database that cannot be seen by regular players	If I access the admin page, then it will show all the user’s data including both username and password	<b><u>Inside “admin-tests.js”</u></b> <pre>it('Log in as admin: view user database --&gt; successful')</pre> <pre>it('Log in not as admin: view user database --&gt; should be denied')</pre>

1	Draw On Canvas	Player	Player is on the canvas	Player can use their mouse to click and draw on the canvas with my mouse	If I click and hold the mouse and try to draw a square on the canvas, it should produce an image that resembles a square.	Front-end test (see demo)
1	Eraser Tool	Player	Player is on the canvas	Player can erase part of my drawing using the eraser tool	If I select the eraser tool and click and drag over some previously drawn lines, it should erase them	Front-end test (see demo)
1	Change Colour	Player	Player is on the canvas	Player can change the color of the drawing tool	If the player selects a colour in the color selection tool with an rgb value of (255,0,0), the brush colour should change to an rgb value of (255,0,0).	Front-end test (see demo)
1	Change Size	Player	Player is on the canvas	Player can change the size of the drawing tool brush	If the player uses the brush size adjustment, the brush size variable should be equal to the size chosen.	Front-end test (see demo)

1	Undo Changes	Player	Player is on the canvas	Player can undo the changes made to the drawing	If the player presses the undo button, the stack of changes should have it's top most entry popped off.	Front-end test (see demo)
---	--------------	--------	-------------------------	---	---	---------------------------

## Iteration 2 Stories

**NOTE 1: For iteration 2, what's on our heroku link is just the UI branch. NOT the master branch**

**NOTE 2: Due to updates to the login system, the admin page is not available on this iteration**

Iteration	Story Name	Actor(s)	Trigger(s)	Action(s)	Test Description / Expectations	Test Functions
2	Redo Button (on fabricJs branch)	Player	Player is on the canvas	Players can redo the changes made by the undo button.	If the player presses the redo button, it should undo the changes the “undo” button made. (Provided that the user didn’t make any other changes).	Front-end test (see demo)
2	Store user info in session	Player	Player successfully logs in	Player enters other rooms, or refreshes the page and the server will store the user’s session and information.	If the player logs in successfully, when they refresh the page, or exit and return to the site, they do not need to log in again unless they logged out.	Front end test (see demo). Look at <b><u>“login-signup.js”</u></b> for session login test <pre>It('Join game room: return successful(code 200) if user is logged in')</pre>
2	Create game sessions	Player	Player is logged in and clicks on one of the game rooms	When a player enters a room, they should only see other people who are in the same room as them.	If the player clicks “join game”, they will be directed to a game session that contains players’ information and the canvas.	<b><u>Inside “socket-tests.js”</u></b> <pre>It('Users can successfully join a game room')</pre>

2	See other people's drawings	Player	Player is in the game room and draws an image on the canvas	Players can see the drawings of other users.	When two different users are in the same game room, they can see what the other person drew.	<b><u>Inside "socket-tests.js"</u></b> <code>It('Users will be able to send their canvas data to others in the same room')</code>
2	Type messages that other users can see in the same game room	Player	Player is in a game room	Players can type messages in the chat. After pressing enter, their message will be visible by other players in the same game room.	If two players enter a room and one player enters a message in the chat, both players should be able to see it.	<b><u>Inside "socket-tests.js"</u></b> <code>It('Users will not receive messages from other rooms')</code>
2	Generate a list of random words for the drawer to choose	Player	Player starts a game session	At the beginning of each round, the drawer can choose a word that they want to draw, and the guesser(s) will guess what the word is.	Once a new round begins, one of the players should see a page that contains a list of three words, and they will be able to select one of the words to draw.	<b><u>Inside "wordlist-api-test.js"</u></b> <code>it('Return type: not null')</code>  <code>it('Return type: returned words are valid strings')</code>  <code>it('Return type: returned strings are valid English words')</code>

2	Generate the link to access the wiki article of the words from each word card	Player	Player starts a game session	When the player is choosing the word to draw from, they can click on the link below those words to take them to its wiki page.	Once a new round begins, one of the players should see a page that contains wiki links for each word, and they can click on it to see further explanations	<b><u>Inside “wordlist-api-test.js”</u></b> <pre>it('Fetching API link: API link successfully fetched and returns content')  it('Fetching API link: returned array contains a valid Wikipedia link')</pre>
---	---	--------	------------------------------	--	--	---



## Iteration 3 Stories

**Note 1: the test files are located in the “test” directory. To review the testing function(s) of a specific story, please look for the test case with the description in pink (inside the it() functions).**

**Note 2: to run the test cases/view results, run the following command:**

**>npm test**

Iteration	Story Name	Actor(s)	Trigger(s)	Action(s)	Test Description / Expectations	Test Function
3	Common Words (Make our words more “drawable”)	User	Player is in a game and it is now their turn as a drawer.	When a player sees the words that they can pick, the list needs to be “drawable”.	When a round begins and a user is able to pick out some words, those words must be “drawable”.	Front-end test (see demo). However, in our <b><u>“wordlist-api-test.js”</u></b> test function, we created a call from our custom npm package to grab some words from our custom list. <code>it('Return type: returned strings are valid English words')</code>

3	Admin Page on a route	Admin	As an admin, if I go to .../admin, I can see the database	Admin logs in with the admin account and goes to .../admin	When logged in as an admin and only an admin, I can go to /admin and access the username database.	<b><u>Inside “admin-test.js”</u></b> <pre>it('Log in as admin: view user database --&gt; successful')</pre> <pre>it('Log in not as admin: view user database --&gt; should be denied')</pre>
3	Logout	User	Player is logged in and on the home screen	When a player is logged in, they can click on a button to logout	When a user is logged in, they can press a button which will log them out	<b><u>Inside “logout.js”</u></b> <pre>it('Logging out function: returns successful and redirect to the home_page')</pre>
3	Create game timer	User	User is currently in a game	The server should display and keep track of a timer of when each phase or round ends	When a user presses the game start and there are at least 2 players in the lobby, a timer should be created.	<b><u>Inside “game-logic.js”</u></b> <pre>It('Checks if timer and rounds are created')</pre>

3	Update page when timer counts down or when round/phase ends	User	User is currently in a game	The page will automatically update when a round or phase ends or when the timer is ticking down	When users are in an active game, they should be able to see the timer counting down to indicate the end of a phase or round. UI should update accordingly to rounds and phases	Front-end test. See demo
3	Add and record points for guesser(s) based on how fast they have guessed the word	User	User is currently in a game	When a player guesses the right word, they will receive points for that word based on a point system and the scoreboard will be updated	When a user guesses	<b><u>Inside “game-logic.js”</u></b>  <pre>It("Users will be acknowledged to get the correct answer and points function will be true")</pre>
3	Choose words	User	User is in a game and they are the artist of that turn.	Players will be able to select 1 of 3 words on the screen.	When a player selects a specific word, that word should be captured in the turn variable.	<b><u>Inside “game-logic2.js”</u></b>  <pre>It('User will choose a word')</pre>

3	Check if any player has guessed the word correctly	User	User is currently in a game	When a player guesses the right word, the chat will be notified, it will NOT display the correct guess to others in the chat, and will receive points.	When a guesser types the right word in the chat, they should be notified	<b><u>Inside “game-logic.js”</u></b> <pre>It("The user should guess the right word" )</pre>
3	Determine the winner of the game	User	User is currently in a game	When the game ends, the winner (player with highest points) will be announced	When the game ends, the person with the highest points should be declared the winner	<b><u>Inside “game-logic2.js”</u></b> <pre>It("Determine the winner of the game of these 2 players")</pre>

## **Admin View Explanation/Instructions**

When logged in as an admin (username: master & password: imurfather), you will have the privilege of seeing a database of users (just their usernames for protection). To do this, simply type “/admin” on the browser. If you are not an admin or not logged in, you will not be able to use this feature and will be redirected to the home page

## **API Uses**

In index.js or in the wordlist-api-test.js file, we are using two API's when we grab words for our artist to see. The first API is the wiki API which lets us retrieve a wikipedia link related to that word, and the second API is a dictionary API which lets us retrieve the definition of that particular word. The purpose of using these API's is to ensure that the drawers understand all of the words, and therefore have more options to choose from. It also provides an opportunity for the players to learn new vocabulary.

## **Real Time Features**

We have many real-time features in our application. For example, the game timers, the chat feature, and the transmissions of the canvas during games.

## **Velocity Discussion**

**General overview:** Our velocity throughout the iterations have definitely been an upward trend. However, it wasn't due to the fact that we weren't as productive in the first two iterations, but a misunderstanding of the scope of this project (*in other words, our project was a lot heavier than we expected!*). However, with each iteration, we would like to say (though no way of actually determining this) that we did at least the same or more than the average amount of work a normal group would produce in each iteration.

## **Velocity Discussion Cont.**


**Iteration 1:** In this iteration we were able to develop the features (or at least a prototype) of the main canvas, a working login feature, and a view database page that only admin users can see. We were also researching and testing chat features with socket.io, but it wasn't submitted to the heroku server. All together, it was about 9 simple user stories that we fulfilled. At the end of the iteration, we were also able to develop a prototype for the UI, but this was not pushed to Heroku.

**Iteration 2:** In this iteration we were able to complete a total of 7 user stories. However, these stories were more complex than our first iteration especially since most of us didn't have experience with socket programming. In this iteration, we were able to improve on the canvas and added a redo feature. Furthermore, we implemented the UI with our features in iteration 1. This includes the chat prototype we had. We were able to make separate game rooms in which people in that room could see drawings and chat messages of other people in that same room. We were also able to incorporate an API feature from wikipedia to show a wiki link for our random words. Although we did complete less stories compared to iteration 1, these stories were far more complex and we felt like it was more work than iteration 1. The majority of the game logic has not been implemented in this iteration.

**Iteration 3:** In this iteration, we were able to complete a total of 9 **complex** user stories. Most of the work in this iteration was related to game logic and creating test functions. We found these stories to be a lot more work since it incorporated the use of many real time features. Other than the game logic, we did have adjustments to our features in the previous iterations. For example, we implemented the fabricJs library for our canvas for added simplicity with our sockets and canvas, and we also added a new dictionary api to grab the definitions of words. We also created a custom npm package for the word list so we don't receive abstract and "undrawable" words. Furthermore, we also re-implemented the functionality of our admin view and decided to use a route instead of a physical button (for security reasons). There's some minor features we wish to include in our final project (like updating the scoreboard as soon as someone guesses rather than the end of the round), but we decided to focus on the main functionalities to save some time. Hopefully in the future we'll continue to work on this since it was such a blast!

## User Interface Requirements

### Logged out homepage

[Register](#)[Login](#)

### Welcome!

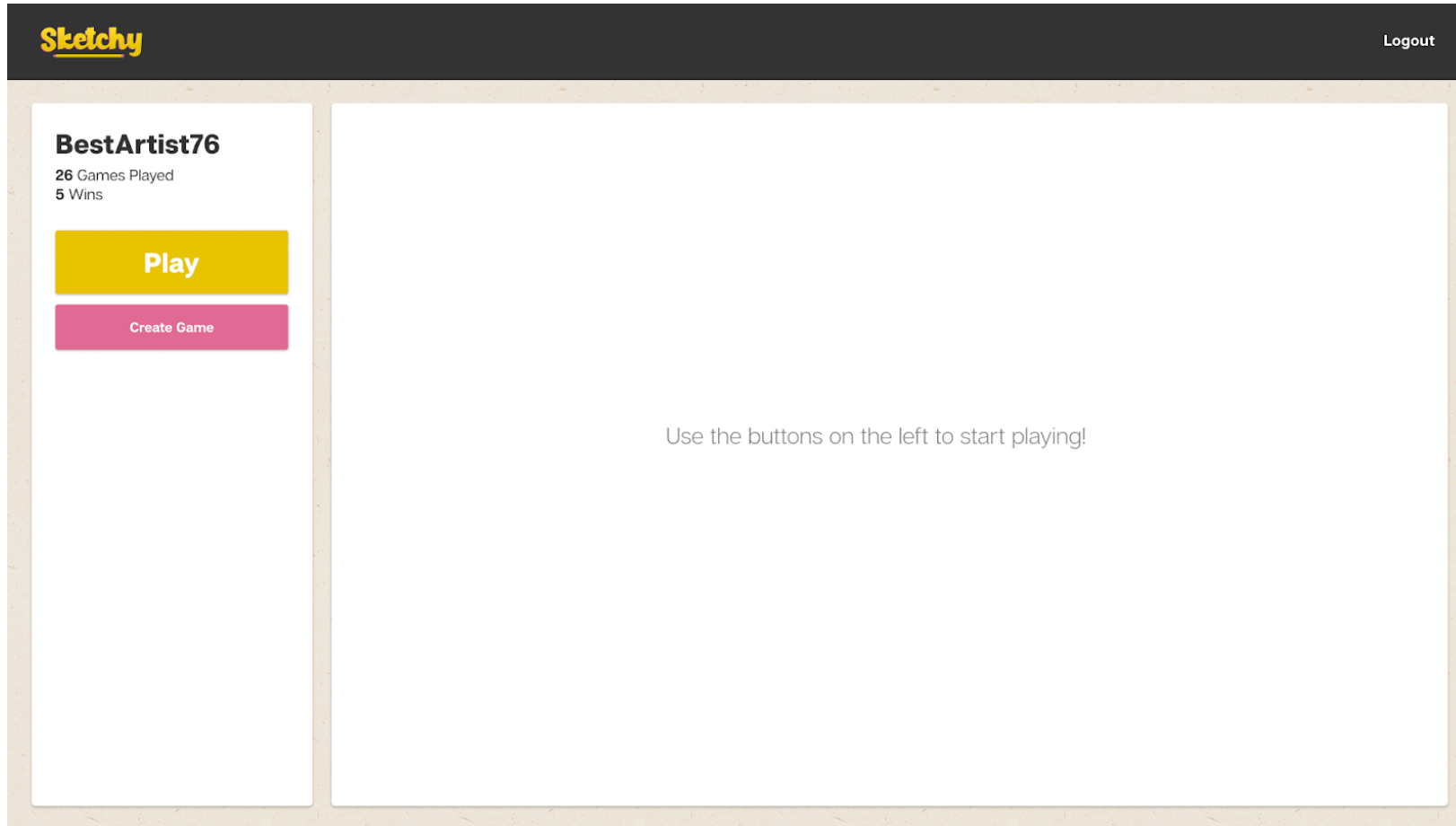
Login or Register to start playing.

Login

Register

Use the buttons on the left to start playing!

## Logged in homepage





## Mid-game page

**Sketchy**

Logout

59

Round 1 of 3

1


BestArtist76 (You)

150 points

3

PlayerB

150 points



2

PlayerC

150 points

4

PlayerD

150 points

6

PlayerE

150 points

Leave Game

BestArtist76: hey guys!

PlayerB: hi!

PlayerC guessed the word!

PlayerD: how did you guess that!?

PlayerB: desert

PlayerE: snowcone

PlayerE guessed the word!

"icecreamb" is close!

PlayerB: snowcones

NewPlayer22 joined the game!

## Game Logic Documentation

- - run `gameStart()`
  - **Create next round**
  - **Round constructor: turns: {t, t, t, t, t}**
    - **Turn 1**
      - (1) Artist
      - (N - 1) Guessers
      - Turn begins in the **“choosing” phase**, where the artist can pick 1 of 3 given subjects to draw. If a subject is not picked within 15 seconds, it is picked randomly.
      - Once a subject is picked, the turn transitions to the **“guessing” phase**.
        - During the guessing phase, players receive points if they guess the subject correctly. A player receives less points depending on how much time has passed or how many players have guessed before them.
        - **Math for determining points TBD.**

- Turn transitions to the **“end” phase** when the time limit is reached or when all guessers have successfully guessed the subject.
  - During the “end” phase, the results of the round (the subject, points per player, who the next artist is) are displayed on screen for all players.
- After the turn ends, the next of N turns begin.

- Turn 2
- ...
- Turn N
- Round 2
- ...
- Round X

## Demo Ideation

As part of your iteration 3 mark, you will be required to give a very short  $\leq 15$  minute demo on your project. You should view me as an interested (semi-technical) user of your system. This will give you good practice on presenting your work in a limited amount of time. Concentrate on the main features to optimize impact of your presentation (as mentioned, you really don't need to demo your login, unless you have a very special login workflow, you should probably concentrate on your bigger features, especially the real-time features :P). In addition to this, in your presentation, you may also want to include (but are not limited to) things like:

- What web APIs did you use?
  - What are some features that you would theoretically like to implement in the future?
  - What features did you try to implement but failed?
  - What are advantages you have over your competitors?
- 
- Intro
    - Show logo
    - Explain briefly what sketchy is
  - Game Demo
    - Go through a game round (or a few turns) - show both artist and guesser perspective
    - Talk about drawing tools and guessing features
    - Show win state
  - Technical Showcase
    - Live chat and drawing
    - Custom NPM word package