System and Unit Test Report

GrepThink LiveChat Fall 2017

Project Owners (Sponsors): Ryan Monroe, Sean Dougher
Team Members:

- Sean Gordon (skgordon@ucsc.edu) - Product Owner (in CS 115)
- Hugh Feng (hzfeng@ucsc.edu)
- Trevor Ching (ttching@ucsc.edu)
- Ali Alkaheli (aalkahel@ucsc.edu)
- Anjali Kanthilal (akanthil@ucsc.edu)

Goal: Building from GrepThink's existing website, our goal this quarter was to create a real-time chat application that allows individuals to message other GrepThink users. Additional functionality and usability aspects are detailed in our user stories.

All User Stories:
Sprint 1:

S1.1. As a developer, I want everyone (other developers) comfortable using git and GrepThink's GitHub so that we can seamlessly deploy our new product.
S1.2. As a developer, I want everyone (other developers) using the same Django and Python versions so that we do not get any code conflicts.
S1.3. As a developer, I want everyone to understand and go over the code given to us so we will know what we will be working on.
S1.4. As a scrum follower, I want everyone to know how to use the scrum board on Trello.

Sprint 2:

S2.1. As a user of GrepThink, I want to be able to have a text based real-time chat so that I can message my team and have my text show up on their screen immediately.
S2.2. As a user & web dev, I want chat to be in a better place so that it is easier to access (left side bar).
S2.3. As a user, I want chat to look functional.

Sprint 3:

S3.1. As a user, I want notifications so I can be alerted when someone posts a new message.
S3.2. As a user, I want to be able to reference someone's profile in chat with @ and give a notification.

S3.3. As a user, I want to be able to direct message another profile so that I can talk to someone one on one.

S3.4. As a user, I want chat to be functional.

Sprint 4:

S4.1. As a developer, I want to be able to use the new LiveChat on Grepthink's webhost Heroku so that when GrepThink uses our code, users won't have problems.

S4.2. As a user, I want to be able to create a channel for a new conversation in my chatroom so that I can separate topics my chatroom discusses.

S4.3. As a user, I want to be able to direct message another profile so that I can talk to someone one on one.

S4.4. As a user, I want chat to be functional and usable.

S4.5. As a user, I want images to show up when I link them, so that I can share more information with people in the chatroom.

Scenarios (System Tests):

Sprint 1:

S1.1:

- Each member clones the fork
- Each member makes a change to a file
- Each member adds, commits, and pushes their change.
- Each member shows they can solve merge conflicts

S1.2:

- Each member installs dependencies from requirements.txt
- Each member shows that they have all the dependencies installed

S1.3:

- Each member can explain where models and views are located.
- Each member can show where HTML and JS templates are.
- Each member can install and run the server.

S1.4:

- Each member creates a Trello account and joins our group.
- Each member can create a new scrum board.
- Each member can create/edit/delete a task.
- Each member can create categories the tasks will sit in (To Do, In Progress, Done, etc)

Sprint 2:

S2.1:

- Start server
- Login as admin to create Chatroom "room1"

- Add users “a” and “b”
- Open two web browsers
- Login as “a” and “b” on each browser
- Navigate to Chat url
- Send message as “a”
- User should see that it shows up immediately for “a” and “b”

S2.2:

- Start server
- Login as user “a”
- User should see a “Chat” element on the left side navigation bar
- User should be able to click on it and view their Chatroom(s)

S2.3:

- Start server
- Login as user “a”
- Navigate to Chat url
- Open a Chatroom
- User should see a box containing previous messages, a text input field, a send button.

<u>Sprint 3:</u>

S3.1:

- Start server
- Login as user “a”
- Navigate to Chatroom “room1” that contains both user “a” and “b”
- As user “a” type a message and send it.
- Login as user “b”
- Check the alerts icon in the top right of the main landing page.
- User should see an alert that new messages were added to “room1”

S3.2:

- Start server
- Login as user “a”
- Go to Chatroom “room1” that contains both user “a” and user “b”
- As user “a” send a message that contains the text “@b” somewhere in the message
- User “a” should see a hyperlink for user “b” in the text that takes them to “b”s profile
- Login as user “b”
- User “b” should see a notification in their alerts in the top right of the page mentioning that they were referenced in a Chatroom.

S3.3:

- Start server
- Login as user “a”

- Navigate to user “b”s profile page
- Click link to send a private message
- User should be directed to a Direct Messages page, where a new Direct messages room should appear with both “a” and “b” as members.
- As user “a” navigate back to “b”s profile page
- Click “send private message” again
- User should still see the same chatroom(direct message) as before with any previous messages. No duplicate chatroom should be created
- User should be able to leave the chatroom(direct message), deleting it.
- User should not be able to add anyone else to the direct message room.

S3.4:

- Start server
- Login as user “a”
- Navigate to main Chat page
- User should be able to create a new Chatroom via “Create Chatroom” button, choosing a unique name, and adding comma separated (valid) usernames. If no (valid) usernames are added, the user creating the room should be added by default. Any non-existing usernames will be skipped over.
- Navigate already existing Chatroom “room1”
- User should be able to add additional users via “Invite Users” button in that chatroom (comma separated list of usernames, same functionality as Create Chatroom)
- User should be able to leave Chatroom via “Leave Chatroom” button. Chatroom should also be deleted when no users are apart of the room.

<u>Sprint 4:</u>

S4.1:

- Connect to the URL for Beta Grepthink, it must not connect to localhost
- Create an account and login as user “a”
- Navigate to main chat page
- User should be able to create a chatroom in the same fashion as system test 3.4
- Navigate to chatroom “room1”
- User should be able to see and use chat in the same method as system test 2.1 and 2.3

S4.2:

- NOTE: We could not get this functionality working (also noted in “Known Problems Report”)

S4.3:

- Start server
- Login as user “a”
- Navigate to user “b”s profile page

- Click link to send a private message
- User should be directed to a Direct Messages page, where a new Direct messages room should appear with both “a” and “b” as members.
- As user “a” navigate back to “b”s profile page
- Click “send private message” again
- User should still see the same chatroom(direct message) as before with any previous messages. No duplicate chatroom should be created
- User should be able to leave the chatroom(direct message), deleting it.
- User should not be able to add anyone else to the direct message room.

S4.4:

- Start server
- Login as user “a” and navigate to Chatroom “room1”
- Send a regular text message.
- User should see their profile picture (or a default one) next to their username and message. They should see a timestamp on the right of the message.
- Click load previous messages button.
- User should see 10 previous messages in the correct order.

S4.5:

- Start server
- Login as user “a” and navigate to any Chatroom
- Send a message with a link to a image/gif/video (granted the format is supported, see user guide).
- User should see the image/gif/video load properly.

<u>Unit Tests:</u>

Our unit tests are located in the following directories:

- /docs/fall17/tests/* (Frontend, functionality tests)
- /teamwork/apps/chat/tests.py (Backend, model tests)