Technical Memo

From the “Documentation.html” file

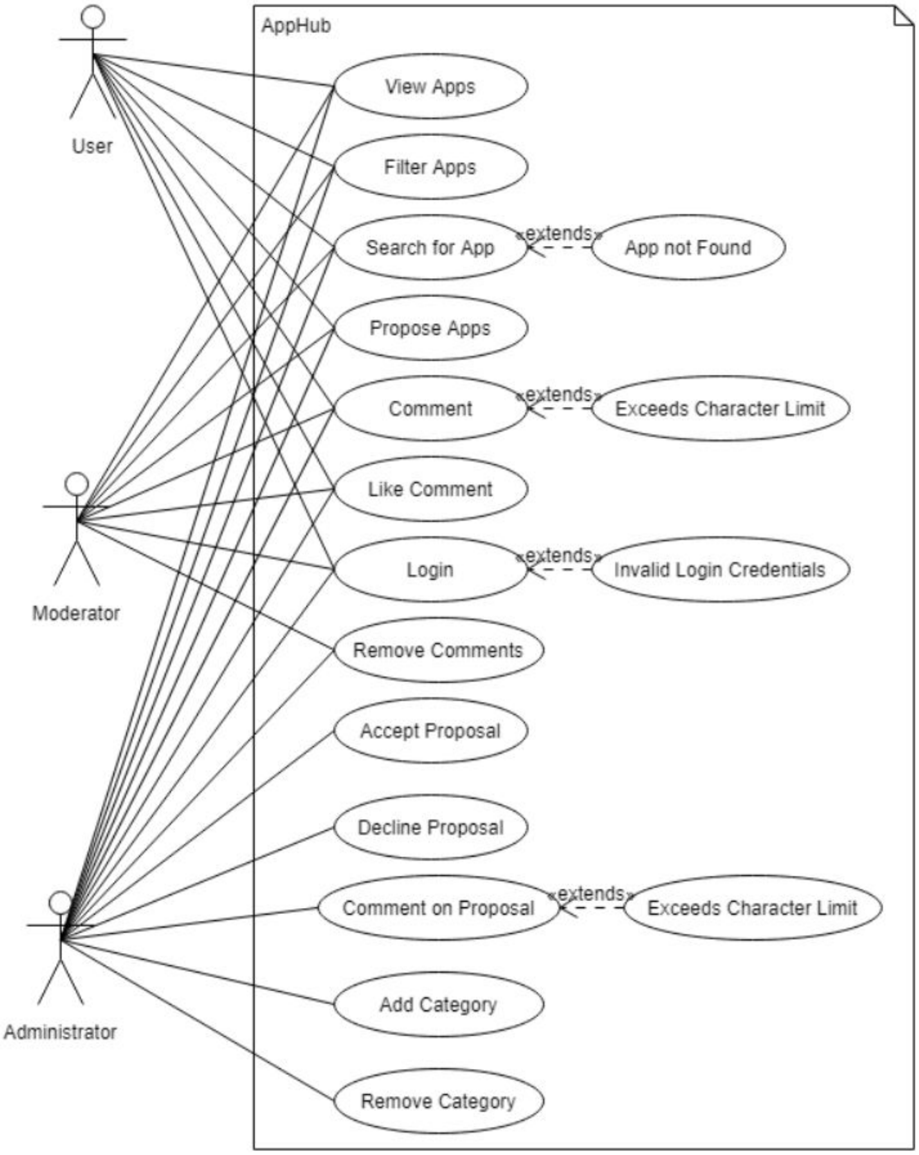
```
<h1>Documentation for this AppHub project</h1>


<p>(Open this file in your browser)
There are a lot of files here so this document will try to explain what they all
do.</p>
<br><br>
<h2>.html files</h2>


<h3>index.html</h3>


<p>This is the file that is immediately loaded by the browser.</p>


<p>In every .html file, the <code>&lt;head&gt;</code> contains all of the .js and
.css files needed. </p>


<p>The <code>&lt;header&gt;</code>.
is also common in index.html and appProposal.html. </p>


<p> There is a <code>&lt;script&gt;</code> section and the bottom of the file that
contains the script to display all of the apps from firebase. This creates a new
<code>AppItem</code> (in appItem.js) object and gets the <code>element</code> field
from it.</p>




<h3>login.html</h3>


<p>Pretty basic page. It contains all of the .js and .css files</p>
```

```html
<p>Contains a form that calls <code>logIn()</code> function in buttonFunction.js
</p>




<h3>createAccount.html</h3>


<p>Similar to the login page, it calls <code>createAccount()</code> in the
buttonFunctions.js </p>




<h3>appProposal.html</h3>


<p>This page includes a form for proposing apps. (Not implemented yet)</p>
<br><br>


<h2>.css files</h2>


<p>This file contains all of the CSS rules for the whole website. It isn't
organized well but we can worry about that later.</p>


  <h2>.js files</h2>


  <h3>firebase.js</h3>


  <p>This file includes the Firebase API. <b>DO NOT CHANGE</b> </p>


  <h3>session.js</h3>


  <p>The idea behind this file is that it basically maintains a user's session
  on the website. </p>


  <p>It contains a function that gets run as soon as the page has loaded (the
  <code>$(document).ready(function(){</code>. This calls the
  <code>initializeFirebase()</code> that Firebase gave us. </p>
```

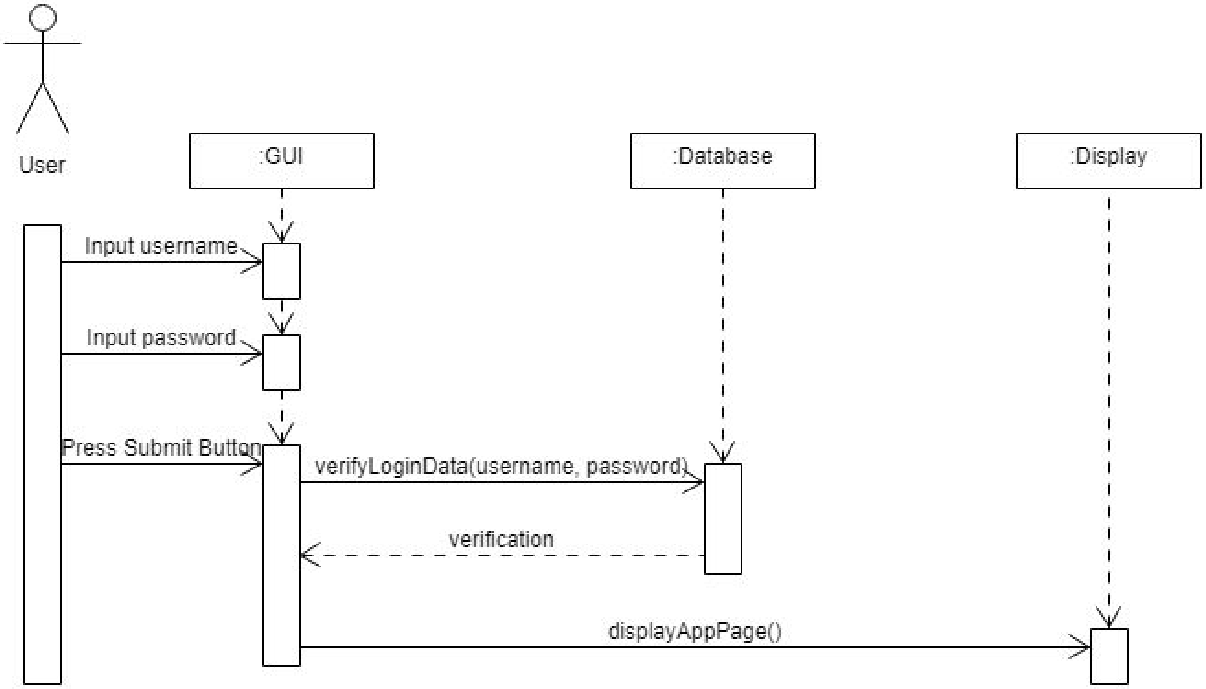
```html
<p>It also controls the login system. When the page loads, it checks the
<code>sessionStorage</code> for a current user.
<code>sessionStorage</code> is a data structure provided by the browser
using a key (in this case <code>user</code>). If the
<code>sessionStorage</code> returns nothing for <code>user</code>, then no
one is logged in. It logs this to the console. Otherwise, it sets the
<code>activeUser</code> global variable to whatever is in there.</p>


<p>Another thing about the <code>sessionStorage</code>. When a user logs
in, the system stores a string of a JSON object. JSON is JavaScript Object
Notation</p>
```

Diagrams

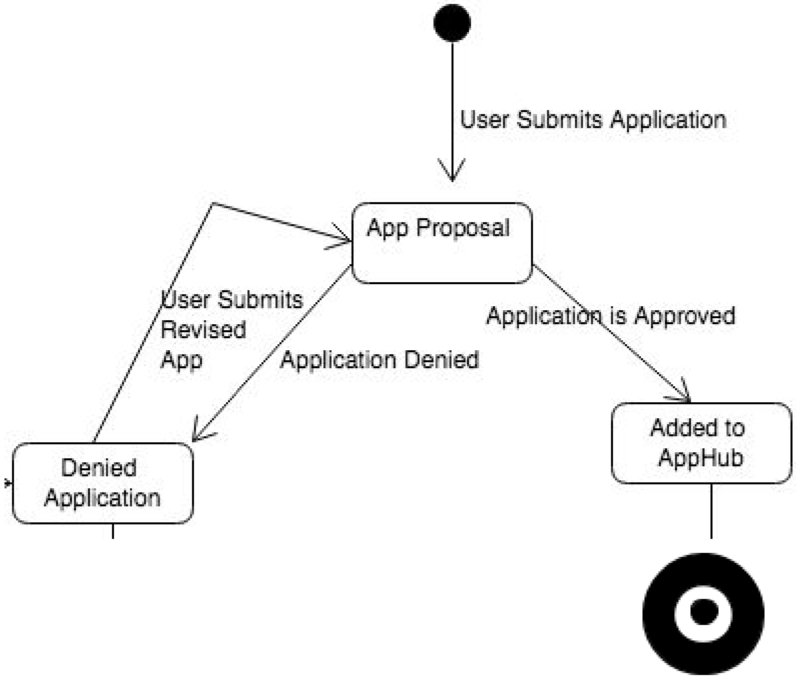
Use Case Diagram



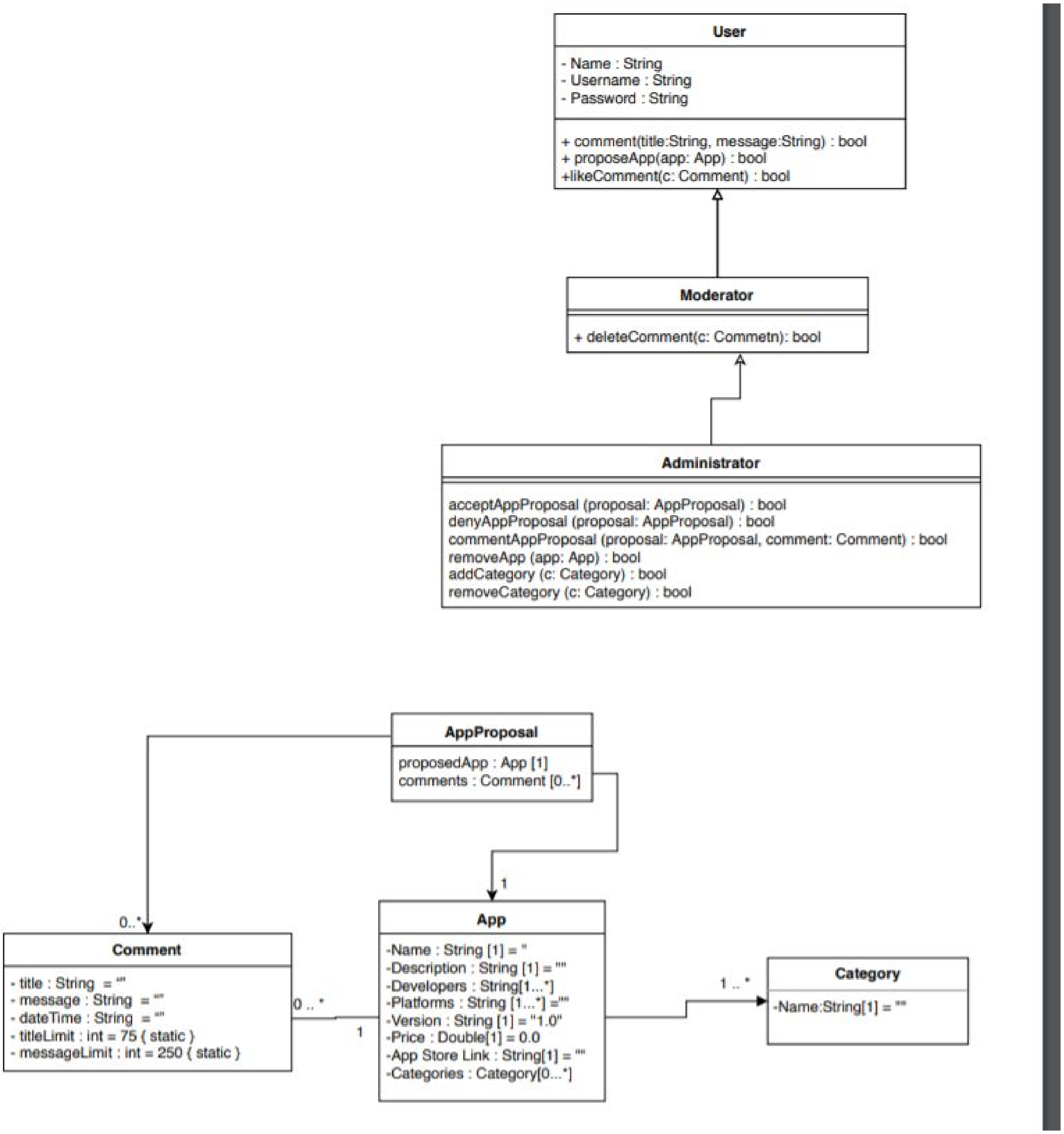
UML Sequence Diagram of Login Process



State Diagram of App Proposal



Class Diagram


User
- Name : String
- Username : String
- Password : String
+ comment(title:String, message:String) : bool
+ proposeApp(app: App) : bool
+likeComment(c: Comment) : bool
Moderator
+ deleteComment(c: Commetn): bool
Administrator
acceptAppProposal (proposal: AppProposal) : bool
denyAppProposal (proposal: AppProposal) : bool
commentAppProposal (proposal: AppProposal, comment: Comment) : bool
removeApp (app: App) : bool
addCategory (c: Category) : bool
removeCategory (c: Category) : bool
AppProposal
proposedApp : App [1]
comments : Comment [0..*]
1
0..*
Comment
- title : String = ""
- message : String = ""
- dateTime : String = ""
- titleLimit : int = 75 (static)
- messageLimit : int = 250 (static)
0 .. *
1
App
-Name : String [1] = "
-Description : String [1] = ""
-Developers : String[1...*]
-Platforms : String [1...*] =""
-Version : String [1] = "1.0"
-Price : Double[1] = 0.0
-App Store Link : String[1] = ""
-Categories : Category[0...*]
1 .. *
Category
-Name:String[1] = ""

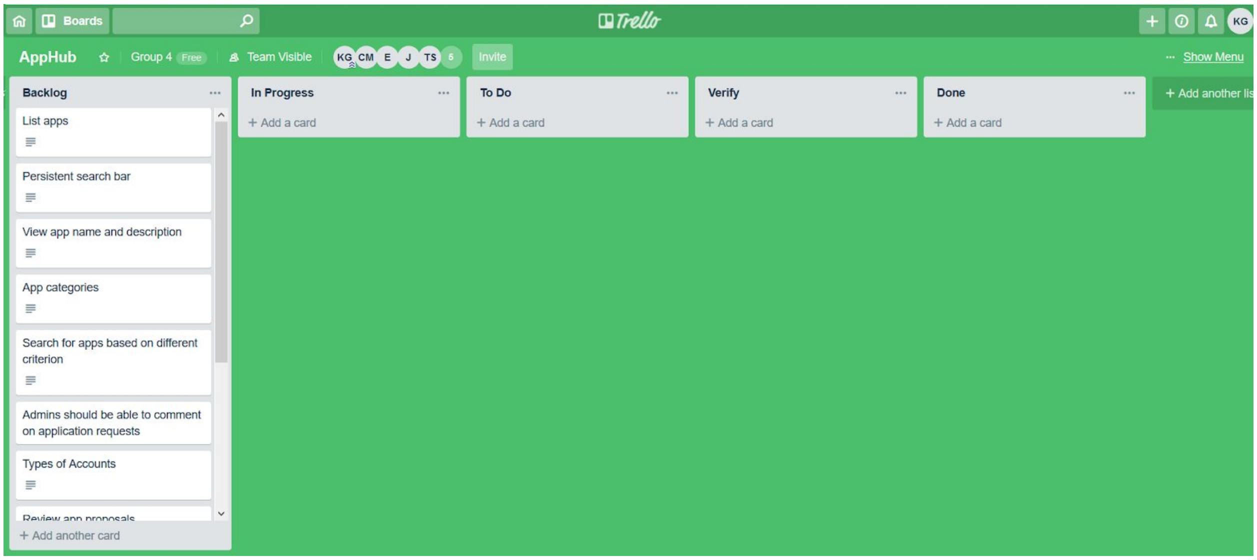
Project Timeline/Weekly Updates

Project Manager: Tyler Sowards
Technical Manager: Kaitlin Gray
Data layer: Ian Albert
Developers: Collin Murdock, Jacob Freedman
Testers: Tyler Sowards, Kaitlin Gray
Domain: Social Media App Catalog
Application Name: AppHub
Software Development Company Name: AppHub Industries

Initial Requirements:

1. List apps
2. Categorize apps
3. View app name and description
4. Search for apps based on different criterion
5. Persistent search bar
6. User and administrative accounts
7. Review app proposals
8. Admins should be able to comment on application requests
9. Comment section - users can post and owners can edit posts
10. Persistent login information and app data
11. Graphical web based interface

Task Board: Trello, https://trello.com/b/p4xq5Juk/apphub
Code repository: https://github.com/Jsf499/201-Project



Backlog
- List apps
- Persistent search bar
- View app name and description
- App categories
- Search for apps based on different criterion
- Admins should be able to comment on application requests
- Types of Accounts
- Review app proposals

+ Add another card

In Progress

+ Add a card

To Do

+ Add a card

Verify

+ Add a card

Done

+ Add a card

From Week 3 Deliverables:

Deliverable: Have a working/running program - optionally with skeleton/dummy objects.
https://editor.p5js.org/sterlingsowards/present/zG-S5_NQ_
https://p5js.org/learn/tdd.html

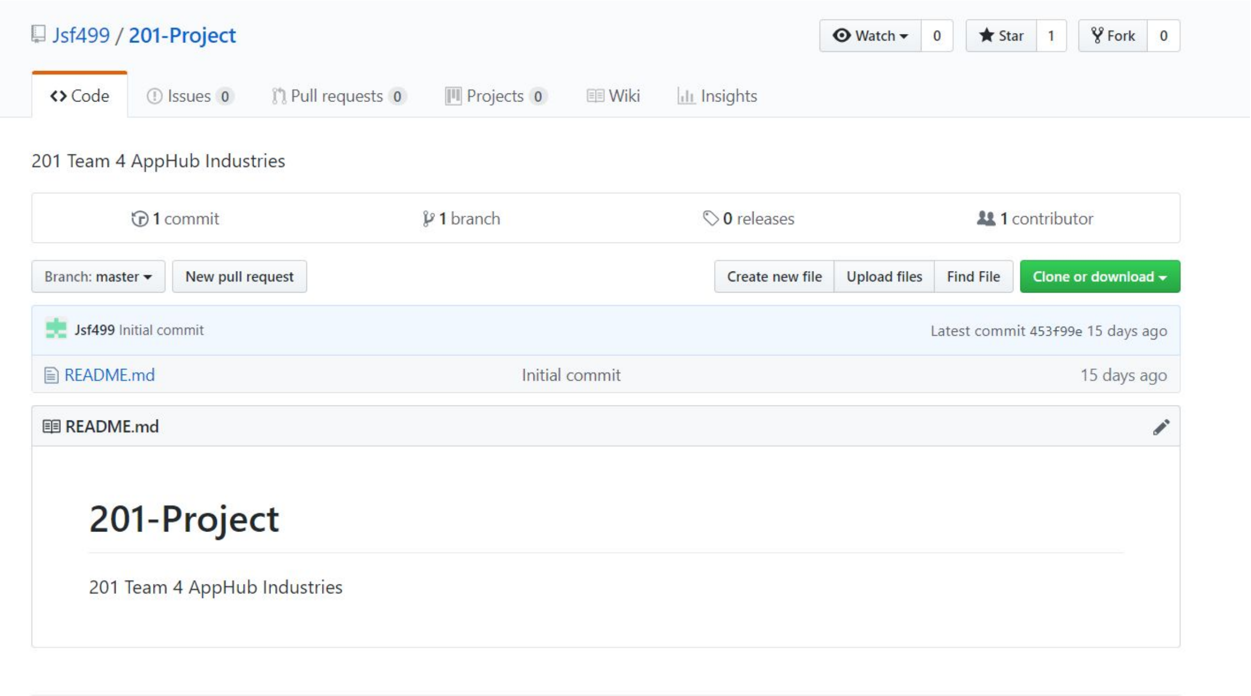
1. Identify and mark clearly on task board tasks for iteration 1.

  - Assign estimate value to each task (hours?)
  - Assign every iteration 1 task to a person.

2. Clearly identify which requirements will be/will not be worked on in iteration 1

- Iteration 1:
  - Persistent login information and app data
  - Graphical web based interface framework
  - Types of accounts
  - List apps
  - Submit app proposal
- Not on Iteration 1:
  - View app name and description
  - Persistent search bar
  - App categories
  - Filter apps based on different criterion
  - Admins should be able to comment on application requests
  - Review app proposals
  - Comment section
  - Accept or deny application proposals
  - Moderation of comment section

5. Setup (SVN, GitHub, etc.) repository. SVN will be taught later.



Jsf499 / 201-Project

201 Team 4 AppHub Industries

1 commit	1 branch	0 releases	1 contributor

Jsf499 Initial commit		Latest commit 453f99e 15 days ago
README.md	Initial commit	15 days ago

README.md

201-Project

201 Team 4 AppHub Industries

6. Prepare agenda for next 2 customer meetings (this one and the next)

Week 3:

1) Prototype version of the web application

2) Changes to requirements

3) Iteration 1 progress on Trello

4) Requirements that will be part of iteration 1

5) Confirm the priorities set on the iteration 1 requirements

6) Burn down chart progress

7) Github progress

Week 4:

1) Implement the iteration 1 requirements

2) Create a use case diagram for an important use case

3) Create a sequence diagram for the use case

4) Keep the burndown chart updated

5) Commit code to GitHub

6) Determine our group's velocity

7) Iteration 1 retrospective

7. Start developing and track your progress using the task board.

Demonstrate some working application project beyond “Hello World”. Ideally a skeleton/first iteration version of your application since your presentations.

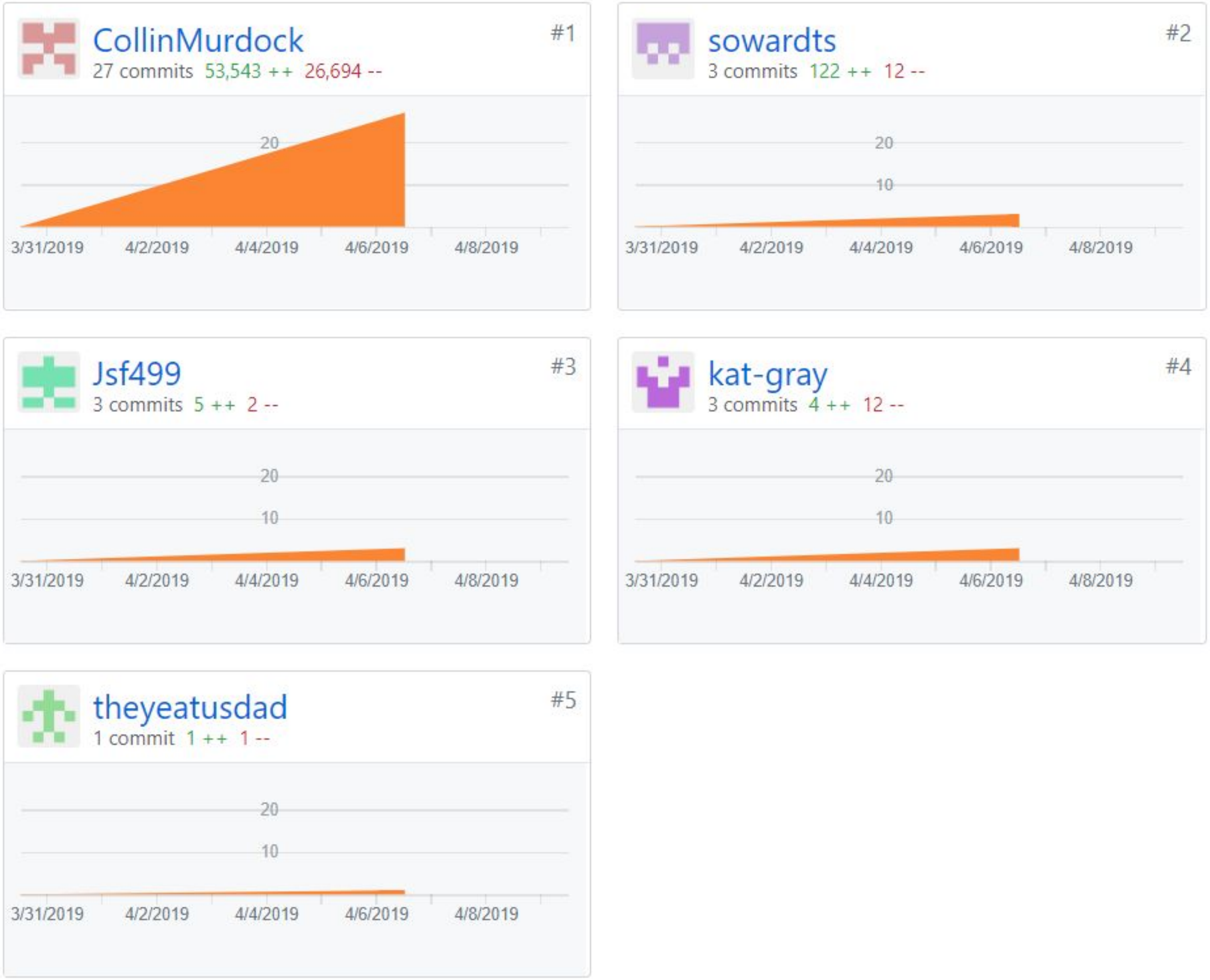
http://ceclnx01.cec.miamioh.edu/~murdoccr/test/

Based on your requirements, devise one UML use case diagram for an “important” use case of your application.

Explicate your current requirement list and priorities for each. Ensure customer is content with them.

Show your customer your updated/tracked burndown chart with progress

Every team member must commit something to the repository (can be documentation, source code, etc)



CollinMurdock #1
27 commits 53,543 ++ 26,694 --

sowardts #2
3 commits 122 ++ 12 --

Jsf499 #3
3 commits 5 ++ 2 --

kat-gray #4
3 commits 4 ++ 12 --

theyeatusdad #5
1 commit 1 ++ 1 --

Prepare the agenda for the next 2 meetings.

Brian,

We will discuss the following at tonight's meeting:

1) Updated demonstration of the application

2) UML Use Case and Sequence Diagrams

3) Requirements and Priorities

4) Burndown chart and Trello progress

5) GitHub contribution of all members

-Group 4

Brian,

According the week 5 deliverables on canvas, here is what needs to be done:

1) Iteration 1 retrospective

2) Create unit test for at least 2 classes

3) Show the final iteration 1 product

4) Burndown chart

5) Update time estimates

6) Prioritize and pick iteration 2 requirements

-Group 4

Iteration 2 continues...

Show your customer your progress.

http://ceclnx01.cec.miamioh.edu/~murdoccr/to_server/

Create build files and scripts for your project.

- For
    - Those using Java, create an ANT build script that will build, run, and clean your project. You can use/test it in Eclipse!
    - Others, write a script (windows or Unix) that allows your project to be built, ran, deployed on the server, or some other aspect of setup.

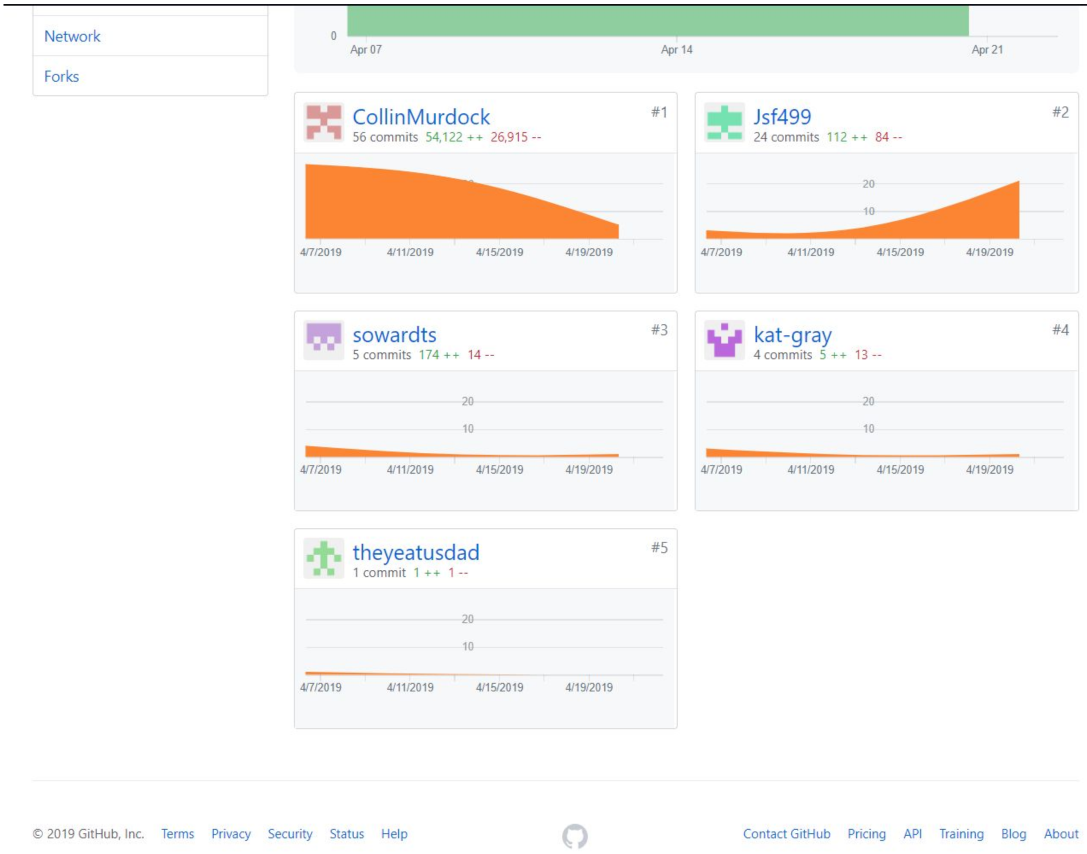
Include documentation on how to run your build scripts.

We do not have a build script, as our project is web-based. To run the program, all you need to do is run the index.HTML file.

Demonstrate your fully functional version control system to your customer. GIT or SVN?

- Every member of the group must have checked in something.
- Demonstrate the main trunk of work to the customer and any branches.

We’re using GIT



Network

Forks

CollinMurdock #1
56 commits 54,122 ++ 26,915 --

Jsf499 #2
24 commits 112 ++ 84 --

sowardts #3
5 commits 174 ++ 14 --

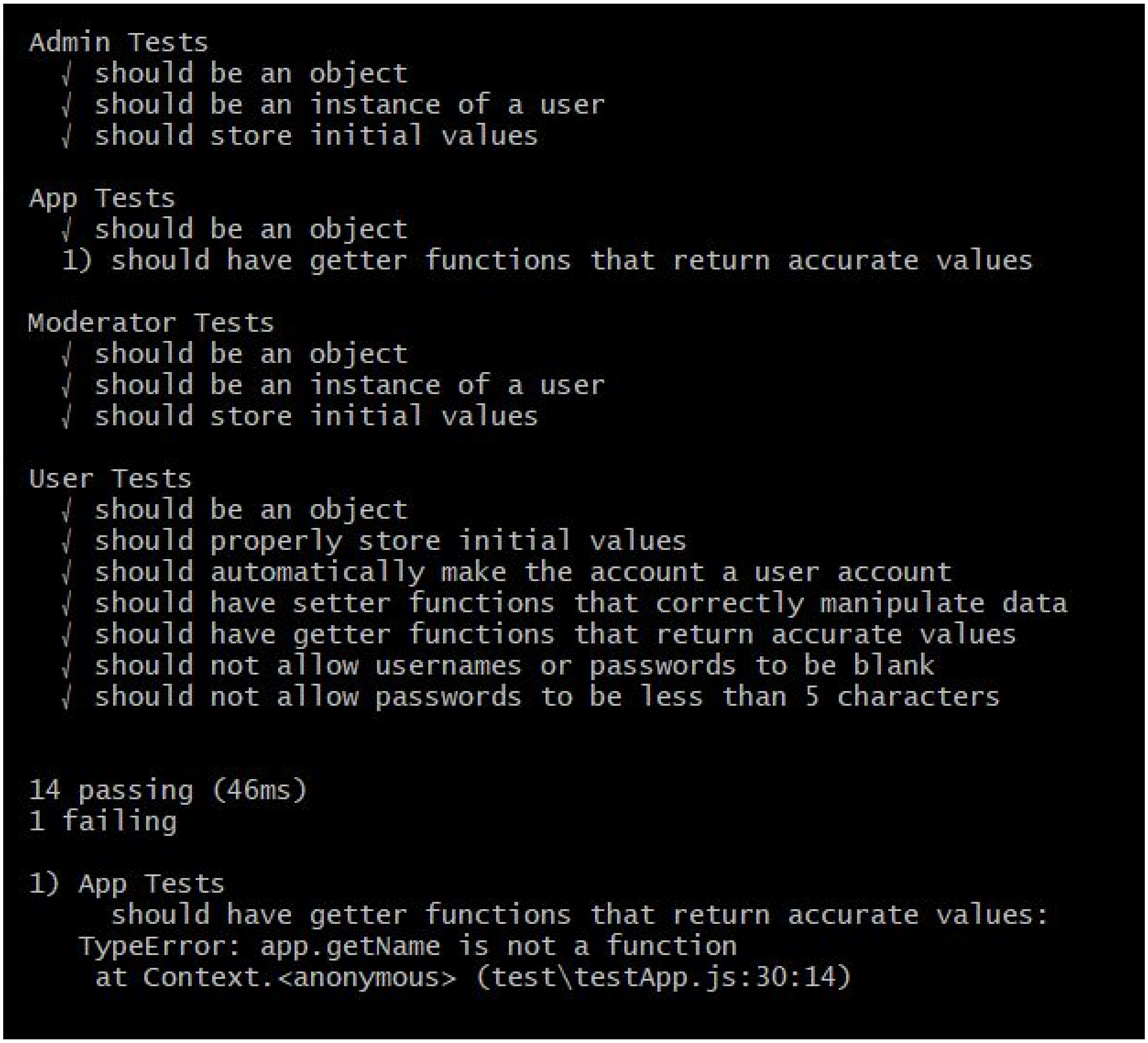
kat-gray #4
4 commits 5 ++ 13 --

theyeatusdad #5
1 commit 1 ++ 1 --

© 2019 GitHub, Inc. Terms Privacy Security Status Help Contact GitHub Pricing API Training Blog About

For a feature that you have yet to develop, write a Test for it first as described in Test Driven Development (Refer Lecture 13). Demonstrate it to the customer. Run the test showing that it is "Red".

- For Java users, create a JUnit test before developing the code.
- For others, write a test suite that covers your new feature.



```
Admin Tests
  √ should be an object
  √ should be an instance of a user
  √ should store initial values

App Tests
  √ should be an object
  1) should have getter functions that return accurate values

Moderator Tests
  √ should be an object
  √ should be an instance of a user
  √ should store initial values

User Tests
  √ should be an object
  √ should properly store initial values
  √ should automatically make the account a user account
  √ should have setter functions that correctly manipulate data
  √ should have getter functions that return accurate values
  √ should not allow usernames or passwords to be blank
  √ should not allow passwords to be less than 5 characters


14 passing (46ms)
1 failing

1) App Tests
     should have getter functions that return accurate values:
   TypeError: app.getName is not a function
    at Context.<anonymous> (test\testApp.js:30:14)
```

Don't forget your agendas for the next 2 meetings!

Brian,

Here is the agenda for tonight's meeting:

1) Present the updated product for iteration 2

2) Discuss how build scripts fit into our project

3) Show our GitHub repo

4) Demonstrate a test that is "red"

5) Find an alternate meeting time for next week

-Tyler

Brian,
Here is what we will do before next week's meeting:
1) Create a testing plan with unit tests, integration tests, and system tests.
2) Divide the tests in the testing plan among the testers
3) Gather final iteration requirements to propose during the meeting
4) Create a documentation plan and generate some initial documentation
-Group 4

- Present your proposed testing plan to the Customer. They will evaluate your plan and must agree/sign off on what they want. This must include
  - A list of all unit tests you plan on performing.
    - Getters and setters for app object
    - Adding app categories
    - Filtering apps
  - All your integration tests (integrating two components at a time)
    - App Categories
  - All your system tests.
    - User Feedback
- Who will be doing white box, grey box, and black box testing? How?
    - Create at least 10 test cases. However, more is better. You will have to implement the tests eventually, but for now, just describe them. Include at least
    - 3 White Box tests
      - For creating an account, give inputs that test all paths within the code (passwords do not match, empty username, username taken, etc..)
      - Displaying app proposal elements. If blocks for getAppProposalElements() in session.js
      - All branching code in firebaseFunctions.js for the login function
    - 3 Grey Box Tests
      - Accounts added to firebase without error
      - App Proposal data added to firebase without error
      - Accepting app proposals correctly moves the proposal to the app section in firebase
    - 4 Black Box (End user) Tests
      - Search results are accurate
      - Test all edge cases for leaving comments on apps
      - Test all edge cases for submitting app proposal objects
      - Testing the page wherein a specific apps data is displayed. Make sure all information is correct and formatted correctly

- Allow the customer to run/test your build scripts. They will evaluate how complete/automatic they are.

Index.html

- As this is the second last customer meeting, have the customer tell you the final feature list they REQUIRE by the time of your last customer meeting.
    - Propose something to them (which they will evaluate), but let them decide.

Done at meeting.

- Show the customer your documentation plan and initial documentation.
    - Who will be doing 1) developer documentation, 2) User/in application documentation?

Ian will do documentation. Documentation exists within the documentation.html page on the GitHub

- How will it be realized? Where will it go and what form will it be in.

User Documentation: how to use your software, features of software, troubleshooting

Developer Documentation: Flowcharts, Technical Brief

- Show the customer your completed Iteration 2 features/items for their approval.
- Don't forget your agendas for the next meeting!

Brian,

Here is the agenda for this afternoon’s meeting:

1) Outline our testing plan for this iteration

2) Demonstrate the automatic build process of our software

3) Discuss documentation tasks

4) Show our final iteration 2 version of the code

5) Determine the requirements for the final iteration

We’ll see you at 4:20

-Group 4

Brian,

Here’s what needs to be done for the next week:

1) Implement the remaining requirements

2) Perform the tests detailed in the week 7 testing plan

3) Finalize the project so that it is in a complete state by the next meeting

4) Finalize all project documentation

-Group 4