<u>Progress Report</u>
- Increment 1 -
Group Streamster (Was StreamHub)

1) Team Members

Isaac Wolfe- iww15- IsaacWolfe

Laura Sarmiento - lss16b - Lulairu

Carlos Alfonso - caa16f - caa16f

Vanessa Myron - vim18 - Vanessa Myron

Ryan Diaz - rd18d- ryandiaz1090

2) Project Title and Description

Streamsters is a Chrome extension that is made to allow users to add both Twitch and Mixer accounts to view streamers they follow and get an extension that notifies them when the streamer is live. We support them having the ability to change their game immediately, get a view count, get length of strea, send out custom messages to followers, and most importantly accurately notify when streamers start streaming from both websites.

3) Accomplishments and overall project status during this increment

Our first iteration we were able to properly test most of the functional requirements of our app, with the exception of login to the users external accounts. We have a functional design made and are making progress towards completing the overall starting webpage.

4) Challenges, changes in the plan and scope of the project and things that went wrong during this increment

One of the challenges that we faced during the design stage of the project was getting the user icon.

Another issue that we discovered was getting "404" with respect to implementing the profile picture.

In respect to code, we had to learn new languages like JQuery and also external libraries. Initially, we planned on using Python to implement our code, but there was no direct way to interpret python code within the browser which prompted us to relearn new languages.

We also had to change our extension name from StreamHub to Streamsters as the name has already been claimed by another company.

5) Team Member Contribution for this increment

a) *the* ***progress report****, including the sections they wrote or contributed to*

i) *Ryan* - Added own contribution to the project, added to various other sections.

ii) *Isaac*- I have wrote most of the webpage folder with the base design for our web page itself with a settings tab, main page, and adding streamers page (this last page is being removed in favor of dynamically shifting between the two pages of home and adding streamers)

iii) *Laura* - I started a code graph for the project and designed the logo and the theme colors of the web page.

iv) *Vanessa* - Implemented the get user follows which gets a list of all channels followed by a specific user which will be sorted by date.

v) *Carlos* - Contributed to working on the overall increment project by making the Case diagram as well as working throughout the paper on various.

b) *the* ***requirements and design document****, including the sections they wrote or contributed to*

i) Ryan - Filled out the basic classes information and functional requirements for the project.
ii) Isaac - Created base web design layout document that was later replaced.
iii) Laura - I write the Sequence Diagram and help Issac with the web page base design
iv) Vanessa - I wrote the get followers function
v) Carlos- I wrote the case Diagram and helped Laura write the Sequence Diagram. Wrote Some of the non-functional requirements

c) *the* ***implementation and testing document****, including the sections they wrote or contributed to*

i) Ryan - Tested APIs using execution-based testing.
ii) Isaac - Tested base functionality of website and layout, still ongoing with debugging and fixing.
iii) Laura -
iv) Vanessa -
v) Carlos- Worked on the execution based non-functional testing section. As well as

d) the ***source code*** *(be detailed about* ***which*** *parts of the system each team member contributed to and* ***how****)*

i) Ryan - Wrote basic functionality of API calls.
ii) Isaac - Wrote all the webpages and files associated with the webpage directory
iii) Laura - I didn't wrote code
iv) Vanessa - Wrote getfollowers function
v) Carlos- Wrote the OAuth Twitch file to allow a user to log in to twitch.

e) the ***video or presentation***

i) Ryan - Walk through the parts I worked on for iteration 1.
ii) Isaac - Walked through web page layout and plan

iii) *Laura -*
iv) *Vanessa -*
v) *Carlos- Had speaking role.*

6) Plans for the next increment

Continue working on the settings webpage, prototyping UI/UX designs. We plan to have all of the functional requirements complete regarding dealing with the Mixer and Twitch APIs. A 3rd goal, if possible will be to have continuity throughout the program lifecycle.

7) Link to video

Paste here the link to your video (only for increment 1 and 2).
https://youtu.be/E1EKHloMwdA

R&D

Overview (5 points)

We are proposing a chrome extension to manage your favorite Twitch and Mixer streamers. Users will receive notifications when their followed streamer goes online, what platform they're on, and even what game they're playing. Streamster will consolidate your favorite games and streamers into one easy to use button.

Functional Requirements (10 points)

1. Darkmode

2. Setting pane on extensions homepage

3. Saved clip libraries

4. Notifications for when streamers go live

5. Clickable window from extension to view current streams with data such as viewers, length of stream, game, and streamer

6. Custom sort methods of how one wants their list of active streams sorted (such as length of stream, viewer count, game, or streamer name)

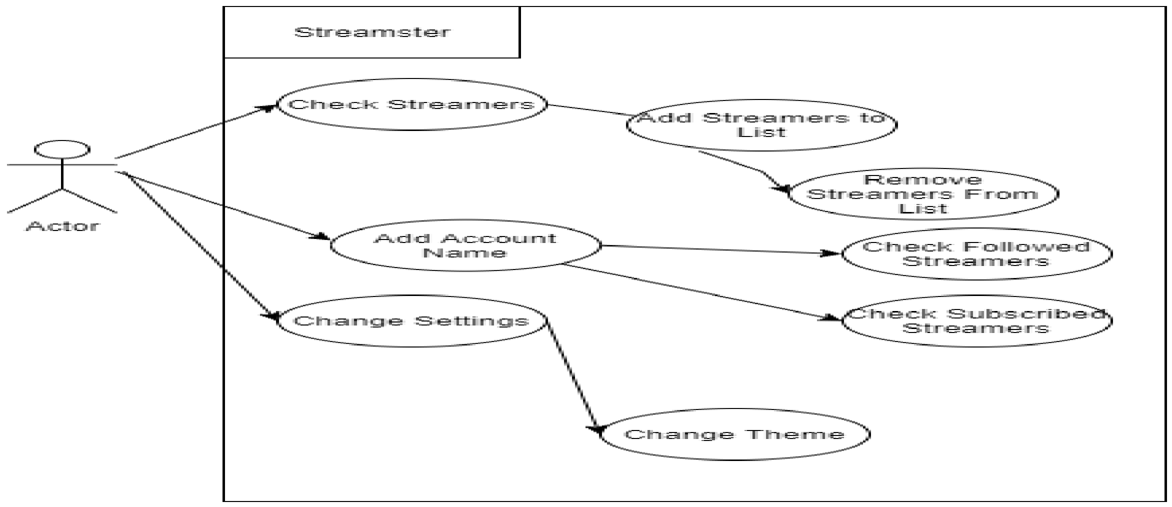
7. Notifications can be customized when going live so watchers can know what specifically is being streamed and ease communication.

8. Possible way to manually change game within the extension

Non-functional Requirements (10 points)

Perfectly optimized design, the app doesn't require much security besides hiding our own API client key unless we decide to store user login data. Webpage

Use Case Diagram (10 points)



Class Diagram and/or Sequence Diagrams (15 points)

Classes:

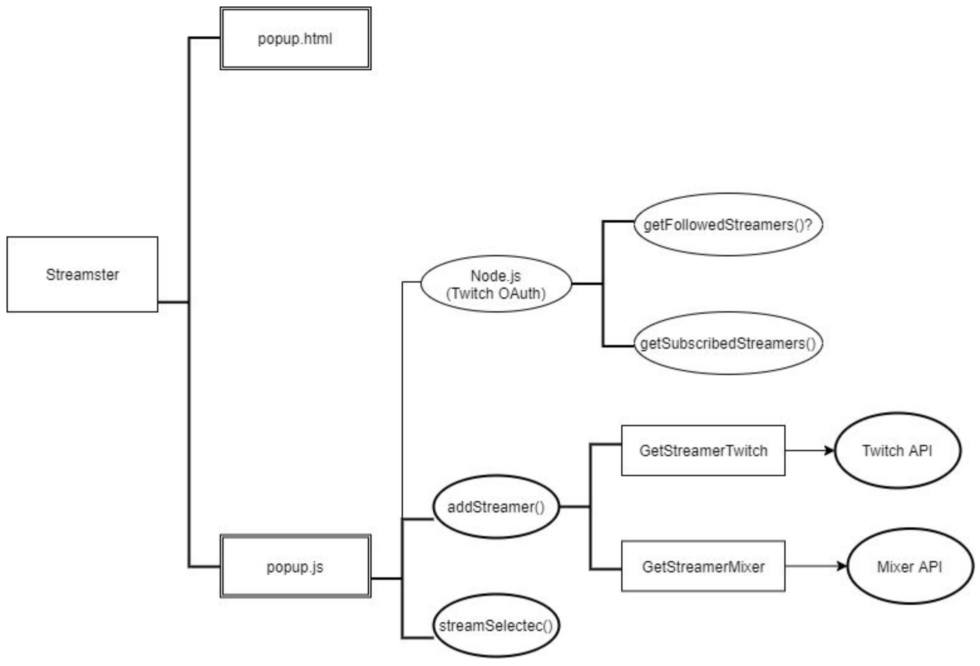
Settings - *Handles user settings such as themes, adding their account names to get their follows*

Popup - *The html page that pops up when the chrome extension icon is clicked. Will show a minified version of the following class*

Following - *A full webpage that will display more in depth information about who the user is following*

Background - *Place to store user data either through the chome.storage api or through our own backend. This will handle continuity to the program.*

A ***Sequence Diagram*** *simply depicts* ***interaction between objects*** *(or* ***functions -*** *in our case - for non-OOP systems) in a sequential order, i.e. the order in which these interactions take place. Sequence diagrams describe how and in what order the objects in a system function.*



Operating Environment (5 points)
The software will run within the Google Chrome browser for each user that installs it. All user interaction will be through the Google Chrome browser.

Assumptions and Dependencies (5 points)
Assumed factors -> We assume that the API will not change in the future. We also assume that none of the libraries will depreciate. We also assume that our operating system (in this case Google Chrome) can handle all the requirements set forth in our project.

Software Implementation and Testing

Programming Languages (5 points)

List the programming languages use in your project, where you use them (what components of your project) and your reason for choosing them (whatever that may be).

Javascript, CSS , HTML are our primary languages. We will be using Javascript for the back end and CSS + HTML to handle our web applications. Our reasons for choosing these languages stems from our need to use certain API's like Twitch and Mixer

Platforms, APIs, Databases, and other technologies used (5 points)
List all the platforms, APIs, Databases, and any other technologies you use in your project and where you use them (in what components of your project).

We will be using the Twitch and Mixer API to make calls so that we can get up to date stream information. We will also be using twitch OAuth so that a user can log in to their twitch account. We will consider using a Database most likely MongoDB if the need arises.

Execution-based Functional Testing (10 points)
*Describe how/if you performed functional testing for your project (i.e., tested for the **functional requirements** listed in your RD).*

We tested API calls to both Twitch and Mixer and made sure that we could deliver on our most important feature to show a streamers name, if they are online, and how many viewers they currently have.

Execution-based Non-Functional Testing (10 points)
*Describe how/if you performed non-functional testing for your project (i.e., tested for the **non-functional requirements** listed in your RD).*

One way we could perform non-functional testing is to test exclusively the calls and commands we are doing from the API. We consider the API to be our weakest link in the stability of our program, therefore Testing the security and speed of our API uses will allow us a more concrete idea of the overall efficiency of our project.

Non-Execution-based Testing (10 points)
Describe how/if you performed non-execution-based testing (such as code reviews/inspections/walkthroughs).

We review the code posted on github to check for invalid packaging or other possible errors having to do with how our code fits together. We also demonstrate any new code so that we are all up to date on it.