Simun Interim Report

Elias Kassell Raymond and James Adams

Simun Description

Simun, name based on the childhood game “Simon says”’, is a de-personalised social network. Users receive content (images, video or text) at random from other users (provided it is amiable) and can then either remove their copy from the network or forward it on to two other users at random. These content items are called **snippets**. Users can also view interesting metric or facts about what has been shared. This includes current most popular posts, type of posts that are most popular.

All accounts will be anonymous and discourage the sharing of personal information. Signing in will however be required for posting, in order to prevent spamming, bots, and potentially to moderate content. This will also provide the opportunity for interpreting trends in related amiability of posts to a viewer.

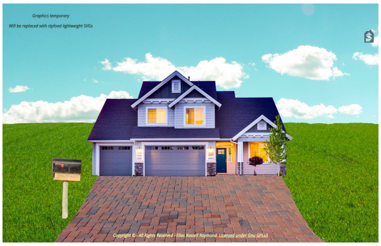
Deployment

- Unzip the package
- Navigate to the site folder
- Run `` `**npm install**` `` to install required node modules
    - For live development and testing, global installations (`` `**npm install -g**` ``) of **pug**, **standard**, **nodemon**, and **watchify** are required. Sass is also required which can be installed with gem via `` `**sudo gem install sass**` ``
- Run `` `**npm run dev**` ``. This will start the server, dynamically watch for updates to Pug or Sass files and open the login page in browser. This page will not be found, Refresh this page after a few seconds once the server has finished starting up
    - If the login page fails to open automatically then the address will be http://localhost:7000
- Run `` `**npm run tests**` `` to run automated tests
- Note that the login and register buttons haven’t been fully implemented yet, and although a lot of the routes for functionality have been set up

there is not much to see! Two things that can be seen are the basic login functionality (input 'Jadams' for the username, and 'Maybe' for the password), and the **receive** page which uses dynamic content loading

Assessment Details

- **HTML: A**
    - Generated via **Pug/Jade** and fed to the user when they connect to the website. Mixins are used to reduce verbosity
- **CSS: A**
    - Generated via **Sass**
- **Javascript: B**
    - The receive page has had select buttons implemented to request snippets from the server, and there is login functionality although it is restricted because cookies have not yet been implemented. There are also some animations
    - The stats page will require a significant quantity of javascript to implement
- **PNG: B**
    - There was an initial implementation of the site which has lots of formatted images (using **Gimp**), but it has been remade as it was too ugly. Here is a screenshot of it:


Graphics temporary
Will be replaced with stylised lightweight SVGs
Copyright © - All Rights Reserved - Elias Kassell Raymond. Licensed under Gnu GPLv3

    - Carefully designed PNGs will be used on the stats page
- **SVG: A**
    - Lots of SVGs have been made (using **Inkscape**) as they provide a crisper feel to the website. For example this custom send button:

- **Server: B**
    - There is a working server implementation using **Express** with url validation, port numbers and redirections
    - The server does not currently handle UTF-8, content negotiation for old browsers, web sockets etc
- **Database: B**

    - There is an integrated **SQLite** database can be interacted with in most desirable ways. Currently images are not stored directly in the database but are hosted on Imgur/other external sites. This dramatically reduces the load on the server
    - More complexity will be added to it as required
- **Dynamic Pages: B**
    - Each page is delivered by the server, and the content on the receive page is requested by the client from the server
    - More pages will require more code

Design Palette

Black	Dark	Primary	Light	White
#000000	#2f4550	#586f7c	#b8dbd9	#f4f4f9

- Titles use the **Glacial Indifference** font
- General text uses the **Open-Sans** font

Feedback

Feedback on current implementation would be greatly appreciated :)