Simun

An anonymised image sharing random network.

Version 1.0

Elias Kassell Raymond

25th May 2019

Introduction

What is Simun?

Simun is a unique **web application** for sharing images and gifs. It is completely **depersonalised** and **anonymous**. It is styled for both **mobile** and **desktop**.

The concept of Simun is based on simple principles. A **snippet** is a shared image or gif wrapped up with a title. When a user shares a bit of content it is converted into two snippets that are forwarded to two users at random. The users who receive these snippets can view them on the website. If they like them, they can **forward** the snippet (with an additional optional **comment**) on to two other users, also at random; alternatively if they dislike the snippet they can either **trash** the snippet or **report** it for inappropriate content.

Users send snippets from an **outbox** and receive them in their **inbox**. Users can also view the top ten snippets on a **global** ranking page. There is also a **home** page which informs users of how to use the website.

The current web address for Simun is **simon.co.uk**. It is hosted on an ubuntu 18.04 virtual machine hosted by Microsoft Azure.

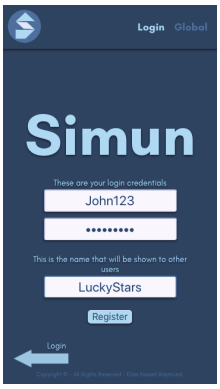
Why Make Simun?

Typical websites track users intensely and try to monetise them, whereas Simun does not. It is designed to completely anonymise users. In fact users **aliases** (visible name attached to snippets) are not required to be unique across the system.

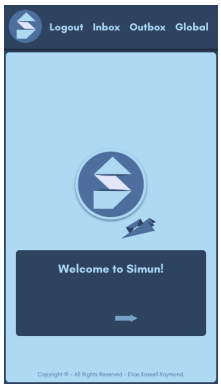
The method for sharing content is different to current social media websites. Simun is different because it allows the sharing of content, but you cannot specify to whom it is delivered; users who know each other in real life have no method of contacting each other directly. In this way it encourages **content diversity** by forcing users to interact with those they may do so regularly

Mobile Snapshots

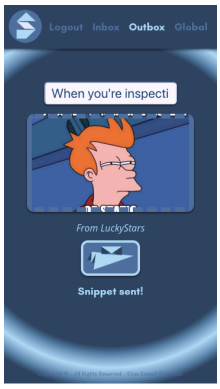
Login



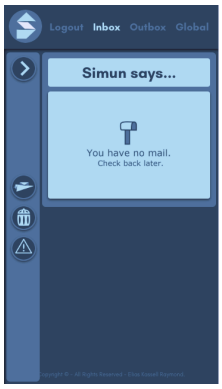
Register



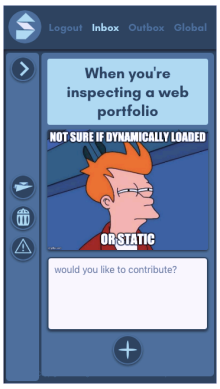
Home Page



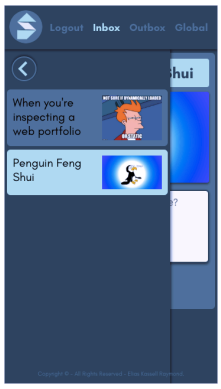
Outbox



Inbox - Empty



Inbox

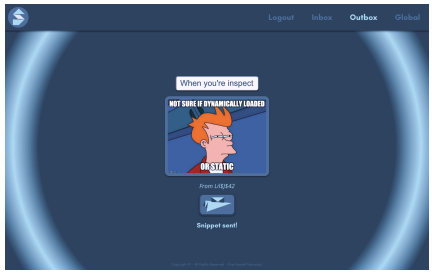


Inbox - Selector

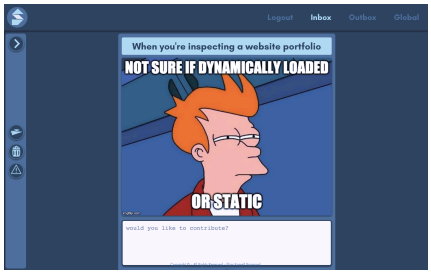
Global

Desktop Snapshots

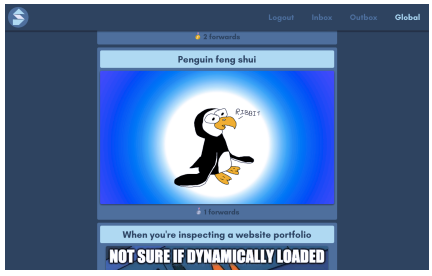
Login



Outbox



Inbox



Global

Technologies Used

The web application is based on **Node.js** and uses the **Express.js** framework, using an integrated **SQLite** database. The application delivers page templates using **Pug** (Jade). CSS for the templates is preprocessed using **Node-sass**. **Bcrypt** is used for password hashing, while sessions are maintained using **Express-Jwt's** JSON Web Token functionality. **Cookie-parser** is used to send and retrieve cookies, with their tokens encrypted with **OpenSSL** generated 512 bit RSA keys. **Mocha** and **Chai** were used for unit testing. **Npm** (Node Package Maid) was used for package management. Other smaller javascript modules were used and can be seen in the **package.json** file. **Imgur** is used to store files.

Inkscape was used for SVG creation and **Gimp** was used for image manipulation. The majority of icon designs were made by following Material Design's Product Icon Design Methodology.

The project is hosted publicly on **GitHub**. The keys stored here are different to those used on the publicly hosted site.

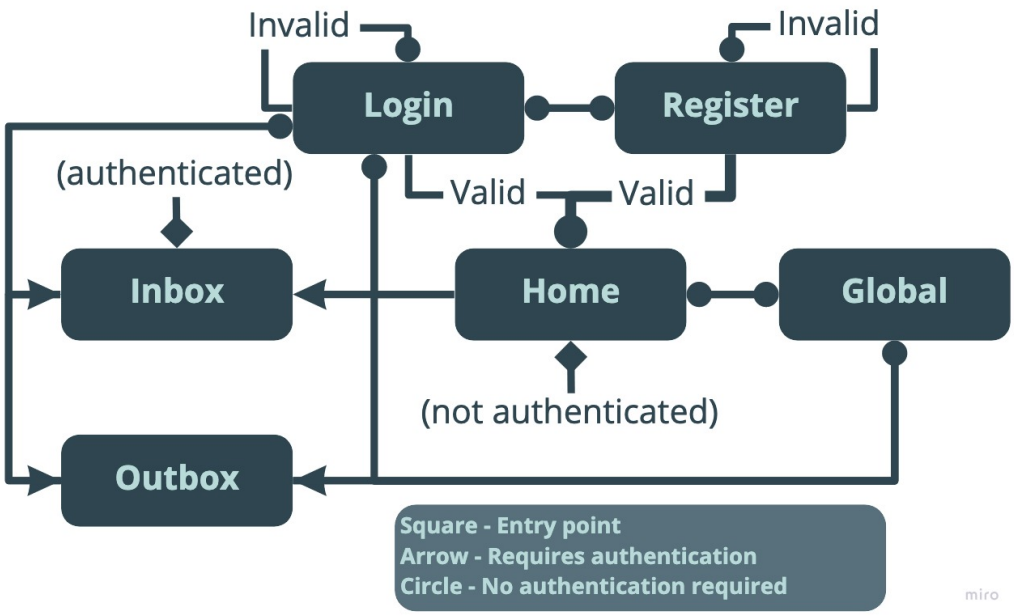
Front End Development

UX and Affordance

The website has content for users who haven't yet logged in, with the bonus functionality of being able to send and receive snippets when logged in. The majority of pages are available to unauthenticated users, as just the inbox and outbox pages require authentication.

The **user flow** (p5) shows that a user is never more than two clicks away from the desired page. In order to make pages transitions clear the available ones are placed in the **header bar**, with the name of the current page highlighted.

Feedback is given to the user when hovering over buttons to provide feedback as to the users current pointer position; a light border appears around them.



User Flow.

UI Style

In order to ensure congruent design across pages, a colour picker was used to decide colours; **coolors.co**. The colour palette contains various shades of monotonic blue (p6). A dark theme was chosen in order to reduce strain on users eyes, just in case the site was addictive enough to cause extreme usage. This resulted in the lighter blue being used as a highlight colour, with the two darker blues being used for background divider colouring.

Color Palette.

The fonts used on the website are the same as in this report. **Glacial Indifference** by Hanken Design Co was used for titles and headers, while **Open Sans** was used for text in the body.

Icons are composed purely of the colours presented by the colour palette. Either sharp edges or ellipses are allowed. Buttons should be styled with circular background contrasting with the layer below's colour. This is not required for text in the header.

Buttons in the Inbox.

Animations have a transition time of 0.2 seconds to smooth visual changes, while not delaying the user's ability to interact with the website.

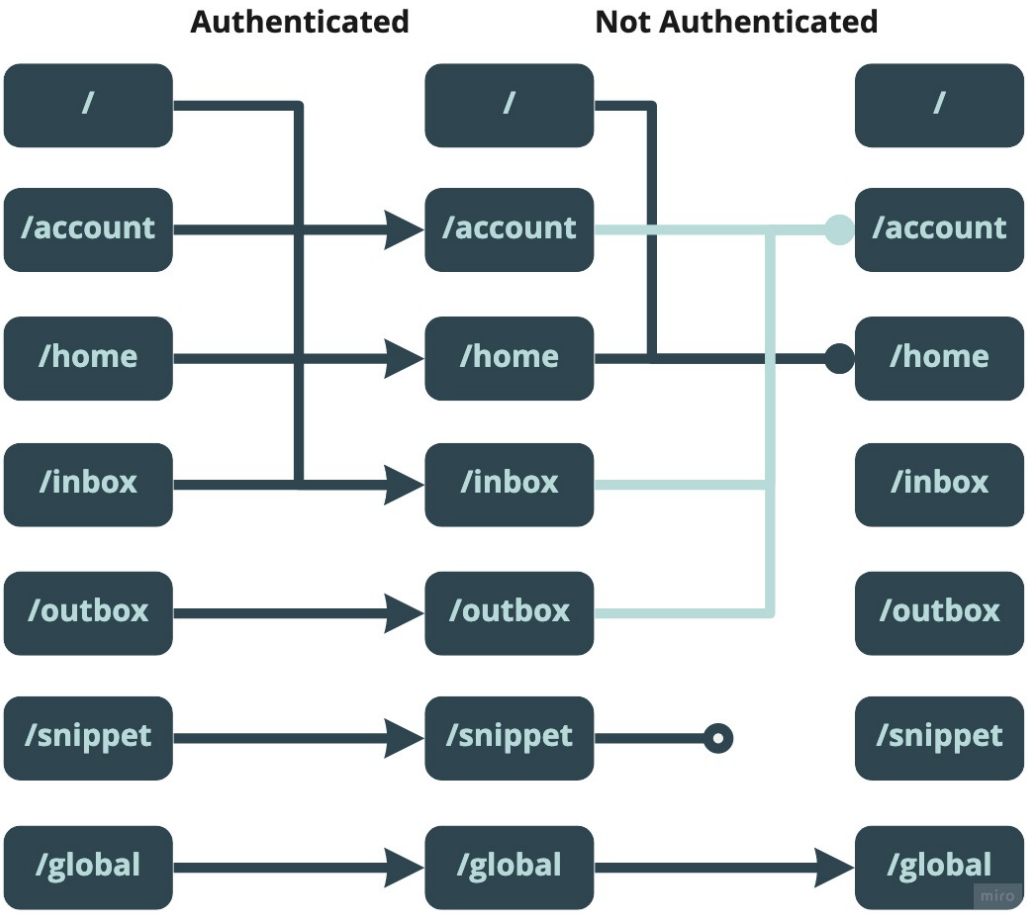
Back End Development

Server Routing

A router was used to redirect requests towards the correct subdomain. A similar split of subdomains was used as pages described in the user flow (p5). The server redirects (p7) can be seen below. Subdomains which may not be obvious are **/account** and **/Snippet**. /account deals with logging in, logging out and registering, while /snippet deals with retrieving snippet and snippet contents.

Authentication

When the user logs in they are given a cookie with a token relating to their session. For every request made requiring authentication, the server first verifies the token, then decodes it if required. The website uses **HTTPS** rather than **HTTP** encrypting all communication between client and server, important for sending the initial password when registering an account.



Server Redirects.

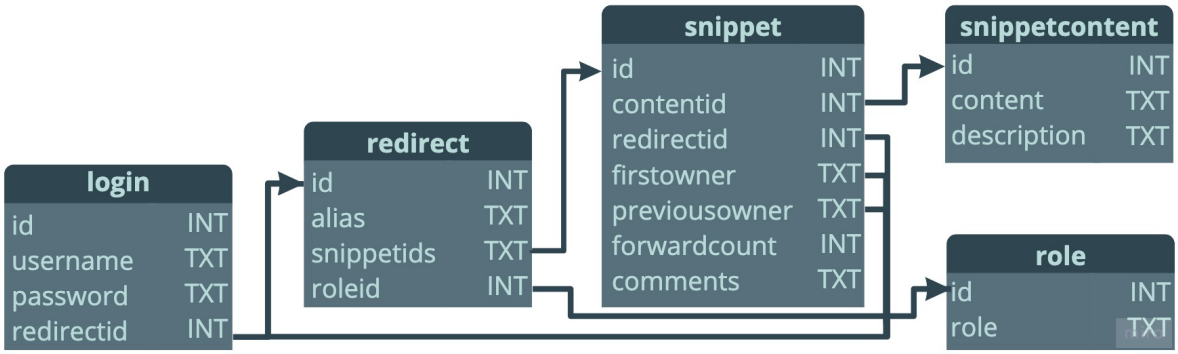
Usernames and aliases must be 3-15 characters, while passwords must be 8-32 characters containing both upper and lower case numbers. Passwords are **salted** 10 rounds and **hashed** before being stored in the database.

Image Management

Imgur is used to store images in order to minimise the computational and networking requirements of the hosting server. This means whatever content is allowed on imager is also allowed on Simun, including **gifs**; the url of the content on Imgur is stored in the database. In order to further reduce load on the server, the images are uploaded directly from the client to Imgur, then the url of the uploaded image is sent to the Simun server.

Database

SQLite version 3.0 was used to interface with the database. There is an additional database set up just for tests, allowing for testing the creation, removal and interaction with data. All database tools are tested. The structure of the database can be seen below.

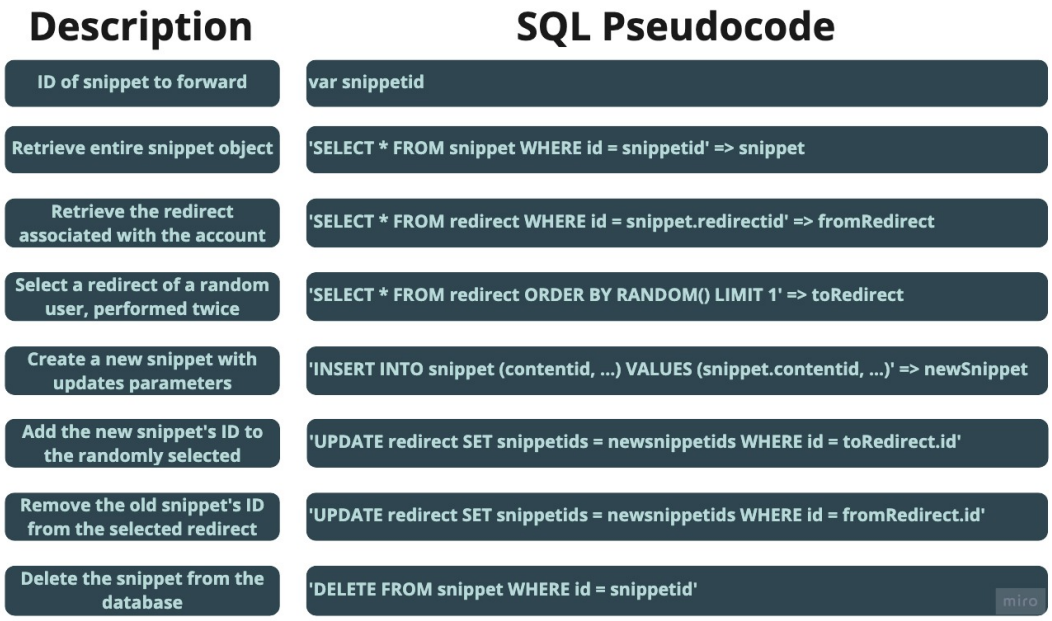


Database Structure.

The **login** and the **redirect** are separate entities in order to keep authentication separate and harder to associate with a user. The login is only ever used when authenticating, while data associated with the redirect is safe to store (encrypted) in a cookie in order to reduce the number of calls required.

The database persists between the server shutting down and restarting by being stored in a .db file.

The logic required to do with **snippet logic** is intensive. The flow for forwarding pseudocode is given below (p9).



Forward Snippet Logic and SQL.

Further Work

Design and Interaction

The information stored in the database would allow for some interesting updates to the user experience. Some examples include providing a **reward** when a newly created snippet ends up in your own inbox, including the comments attributed with the most forwarded snippet on the global page, or allowing users to **vote** on particular comments to move them further to the top (similar to as on reddit).

Optimisations

To make the web application more light, the efficiency could be improved by reducing the number of calls to the database.

Currently the website is not scaleable, only allowing about 100 active users; in order to make it more scaleable different technologies would have to be used, such as using **Go** for the server and **MongoDB** for the database, containerised with **Docker**.