



Expressionary

Team 21 Design Document

Michael Quade, Ayush Patel,
Alexander Ferrard, Syed Mudabbir

Purpose

Social media and the internet have become places for expression. Twitter encourages a short and concise message, while websites like Instagram allow a user to display their life through visuals. Through these platforms, a new language has evolved. The use of inside references and acronyms has led to a culture that cannot be understood from the outside.

Expressionary is a website with the goal of creating its own unique form of expression. *Expressionary* presents users with real words found from the dictionary, such as immoral. The user can provide their own definition of the word. This can be literal, funny, or related to current events. One user might define immoral as, “The hoarding of cardboard”. Another user might refer to it by its literal definition. The users will shape the meta of the website, and form their own subculture on the internet. Definitions will be judged by fellow users using a simple +/- point system. Beyond the words, we are looking to expand into pictures, phrases, and possibly poetry.

Functional Requirements:

1. Account Creation & Management

- Users can create an account.
- Users can change their password.
- Users can view their profile.

2. Word pages

- There will be pages for words from the dictionary.
- Users will be able to post definitions to these words.
- Users can vote up or down on definitions to affect their point value.

3. Word Leaderboard

- There will be a page to view the word pages with the most activity, determined by the number of points given and taken to definitions on that page.
- With this users can take part in words that are popular right now.

4. User Leaderboard

- Shows the users with the most points.
- Good way to show the most successful users on the site.

5. Search Function

- Provide a search engine that can search for words or users.
- Users will be able to search and then go to the pages that they search for.

6. Random Page Function

- There will be an option to go to a random page.
- This will take the users to a random word page to participate in.

Non-Functional Requirements:

1. **Client Requirements**

- The webpage should be able to run smoothly on a laptop or computer without any bugs or crashes

2. **Extra Service**

- Sending the clients emails with updates regarding there posts
- Users to be able to personalize there notification settings

3. **Security Requirements**

- Encrypting the username and data while they pass through the server client medium
- Encrypt user personal information in the database to protect from hackers and unauthorized accesses

4. **Aesthetics**

- The website to be easy to use, user friendly and, if time allows, customizable to some extent.

Design Outline

High Level Overview

This Project uses a standard client-server model. The Client makes requests to our Node.js server, and the server sends responses back to the client via the web page. The figure below shows the high-level overview of the system.



1. Client

- Sends requests to server
- Receives full HTML pages from server.
- Receives responses from Node.js server

2. Node.js Server

- Receives client requests and checks for validation
- Processes responses and sends queries to the SQL database
- Generates HTML responses to send back to the client

3. SQL Database

- Maintains data of all user information.
- Stores information of words and their definitions.
- Updates point values and other information.

Design Issues

Functional Issues

1. Do users need to login to use the website?

Option 1: No login needed to browse site, but necessary to participate

Option 2: Login required

Decision:

We chose to use an optional login approach for the website. This way people can get a feel for the website before choosing if they would like to participate. The account system will be local to the site, and not connected with Facebook or Google accounts.

2. What back end framework should we use?

Option 1: Python

Option 2: NodeJS

Decision:

We decided to use Node.js for our backend. The main reason for this is we have some experience among us with Node.js. Also, Node (and more generally Javascript) offer a lot of great packages that will help us complete different functionalities for our design.

3. What database software should we use?

Option 1: MongoDB

Option 2: MySQL

Decision:

While we have experience with both softwares, we have decided to use MySQL. We have a bit more experience with MySQL and it has seen wide use by different servers. This will allow us to easily access and change the information in our database.

4. What front end framework should we use?

Option 1: Angular

Option 2: React

Option 3: Express

Decision:

This was a hard decision, as we all have limited experience with these frameworks. However, Express was made for Node.js, and so we feel it will be best to use this. Express allows for easy to use modules on web pages that the users can interact with and the Node.js server can process.

5. How should User permissions be handled? (admin mode/ user mode/veteran users/ new users)

Option 1: Create New Class for each Type

Option 2: Use One class with different permissions to access and post

Decision:

Using one class to encompass all users and then having a permissions “state” variable would likely be the easiest way to handle which users have what access rights. In this scenario, when the user comes onto any page in the site, the User Interface will only generate options that they have the permission to access.

Without the UI elements to send higher-rights-access requests back to the server, lower-rights users will not be even be given the option until their permissions are updated, and the UI is refreshed. This way we can avoid performing a check with every request, since if the user had insufficient rights, they would not have had the option to send that request in the first place.

Non-Functional Issues**1. What type of encryption should we use for sensitive data?**

Option 1: RSA encryption

Option 2: AES encryption

Option 3: Our custom encryption

Decision:

For now we are thinking of using our own encryption so it can’t easily be decoded unless someone knows the source code. This would make our program more versatile. However this is a security issue and might change depending on the security requirements of that present time.

2. Set up an android application for the website for easier access?

Option 1: Yes

Option 2: No

Decision:

For now we are keeping it simple with just a webpage. An android application seems like a good idea, however, given the time and workload we have decided to not do it right now. It might be added on in a later update, however that is not for sure.

3. Have multiple servers running the website at the same time so that we can handle enough concurrent users at the same time?

Option 1: Have multiple servers running

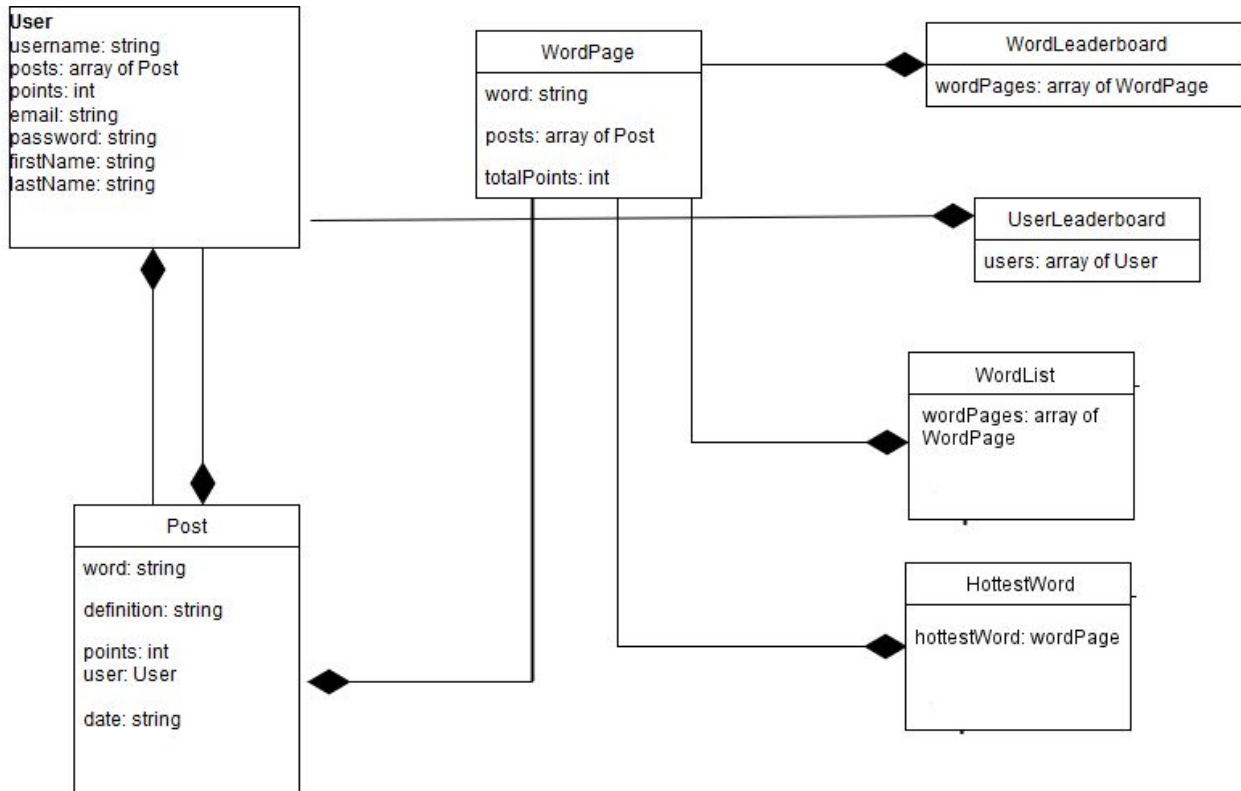
Option 2: Keep one server for now with room for expansion

Decision:

Since our website is just starting, we are not sure how the traffic is going to look like. For now we are keeping only one server running on a local device. We will keep monitoring the traffic of the website to ensure server running speed. If the traffic gets higher than expected then we will consider buying more servers to run the website concurrently

Design Details

Class Diagram



1. User

- The user information is stored in this class. The username, email, password, first and last names are determined at registration. Posts will include the definitions posted by the user, and their points determined by the points those points receive. In the future there could be links to other users that the user follows or are friends with will also be in this class. A profile picture could be added in the future as well.

2. **Post**

- A post contains the word that it is attached to. It carries the string definitions that is posted. It tracks the points given to it, as well as the user who posted it and the date.

3. **WordPage**

- Used for the physical page, but also culminates the total activity of a word. Contains the word, the posts attached to it, and the total points involved in these posts.

4. **UserLeaderboard**

- Contains an array of users sorted by their points.

5. **WordLeaderboard**

- Contains an array of words sorted by their points.
- It's important to note that this is not a point total based on the standing points of the posts involved, but rather the total times a vote has been cast on a post of a word. So the negative votes will also increase the point total for this leaderboard.

6. **WordList**

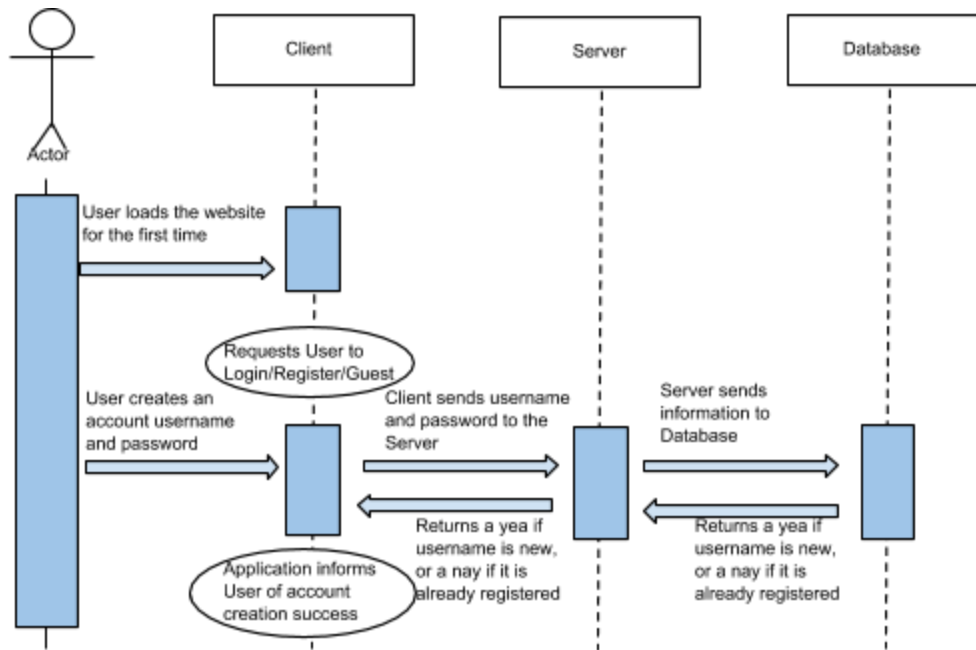
- Contains a list of the entire library of words we have.

7. **HottestWord**

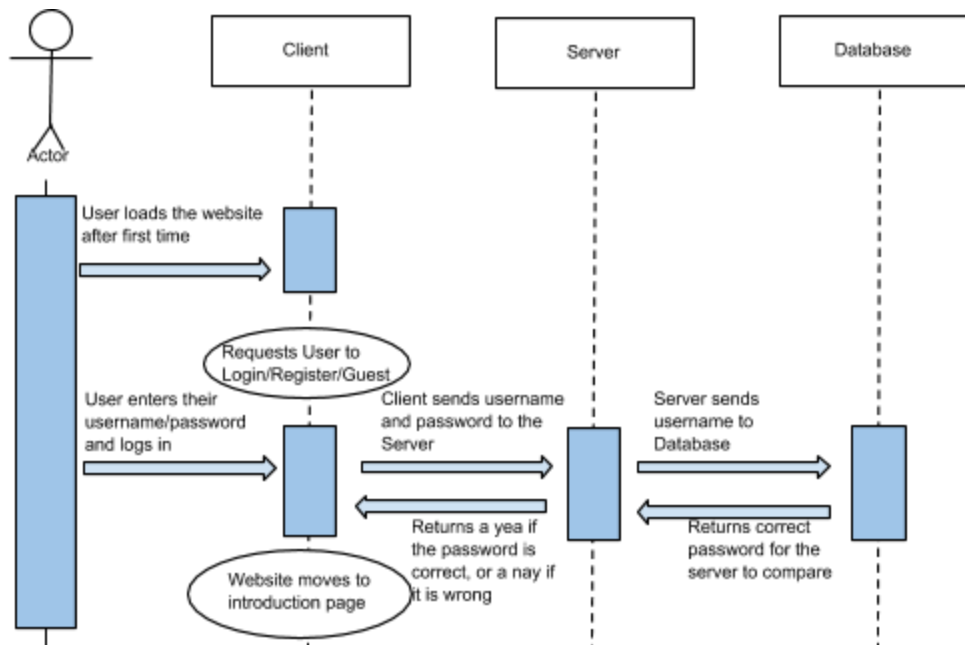
- Presents a link to the hottest word page, determined by top activity within a certain period of time.

Sequence Diagrams

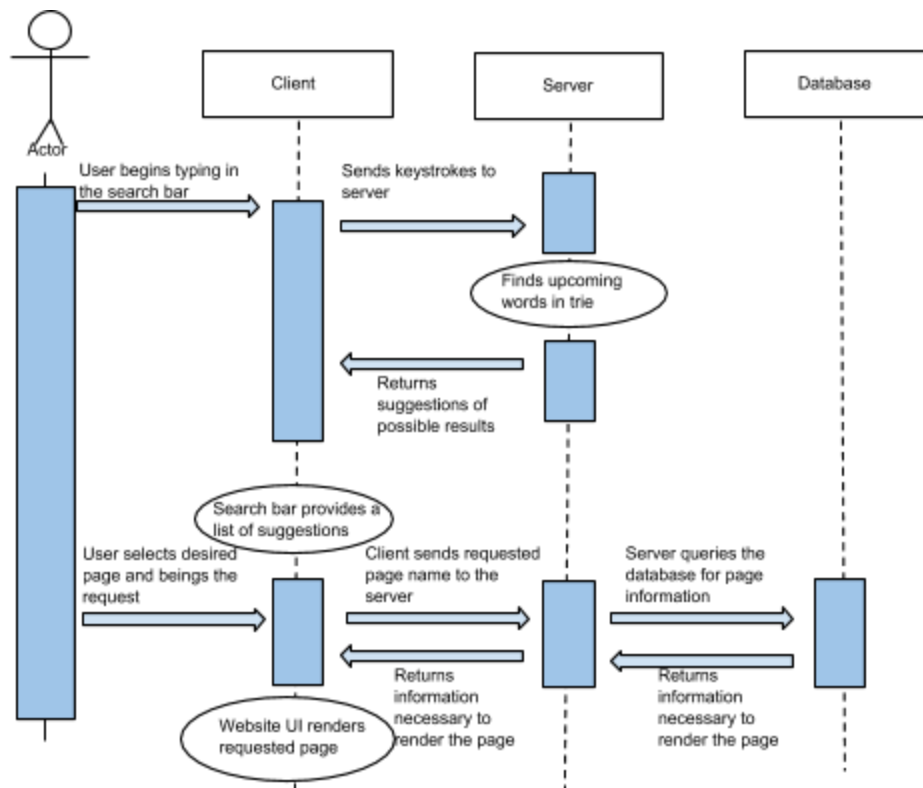
Events that occur when the user opens the website for the first time



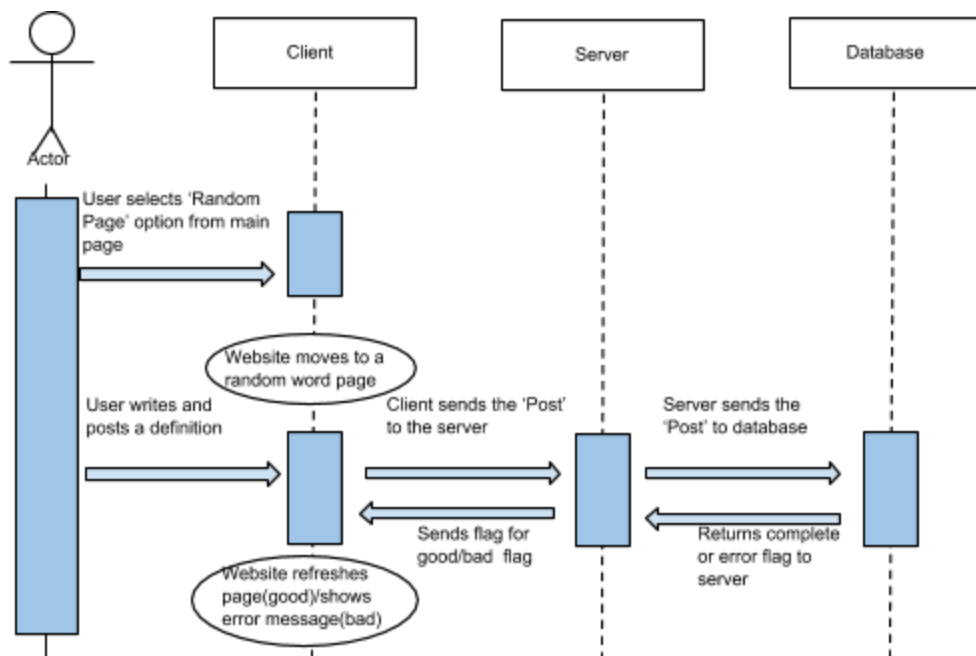
Events that occur when the user opens the website after creating an account



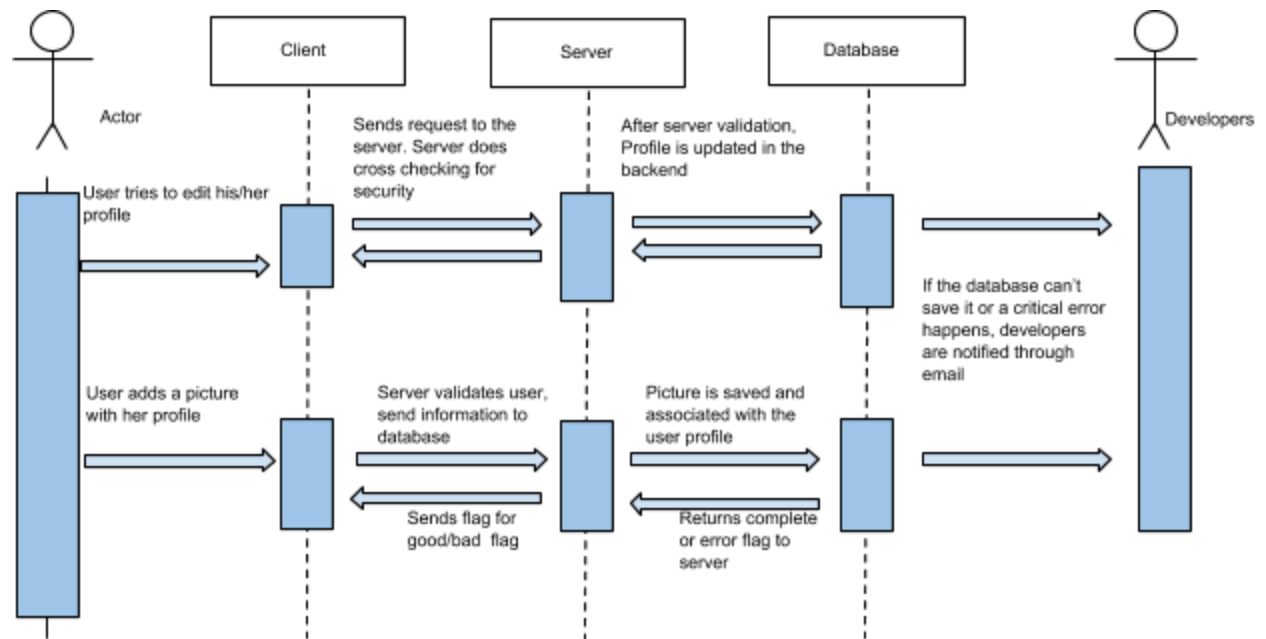
Events that occur when the user uses the search function



User selects random page option from main page



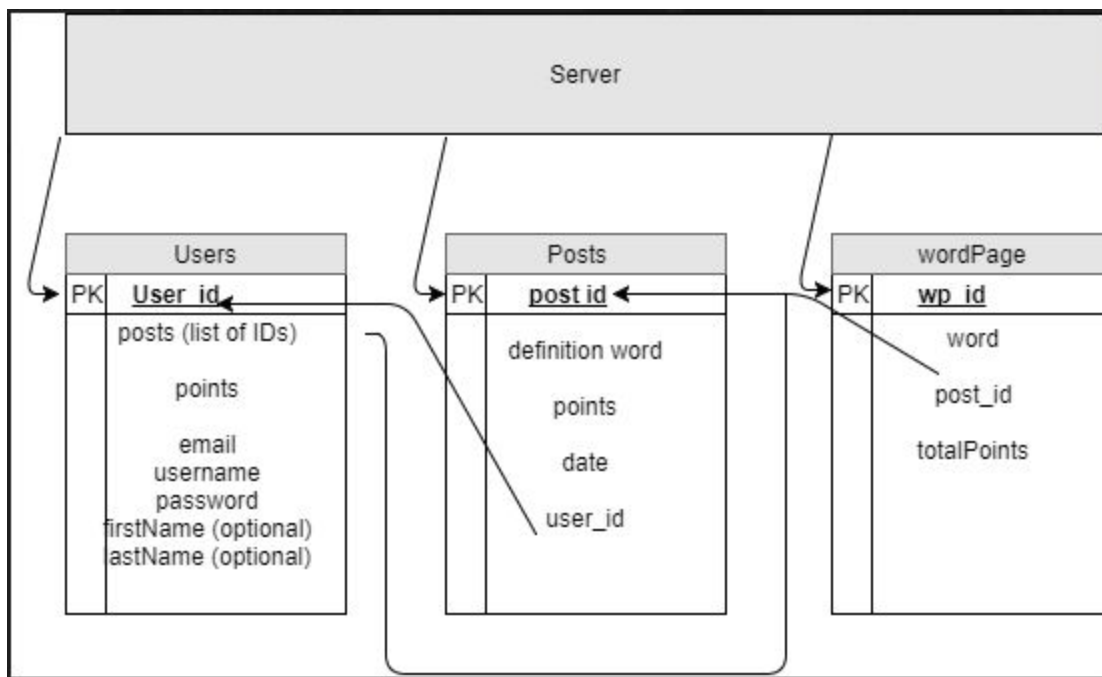
When the user tries to post a definition



Database Diagram

We operate under a relational database which contains tables to store user data, an archive of posts and it's associated values, and word page data. All of these tables will be used to add entries into the the tables following them in the diagram and all will be used in order to generate the UI when a user navigates to a page.

The primary keys come from the server when new entries are created, and are used as identifiers in following tables to gather all of the necessary data when the UI is being generated.



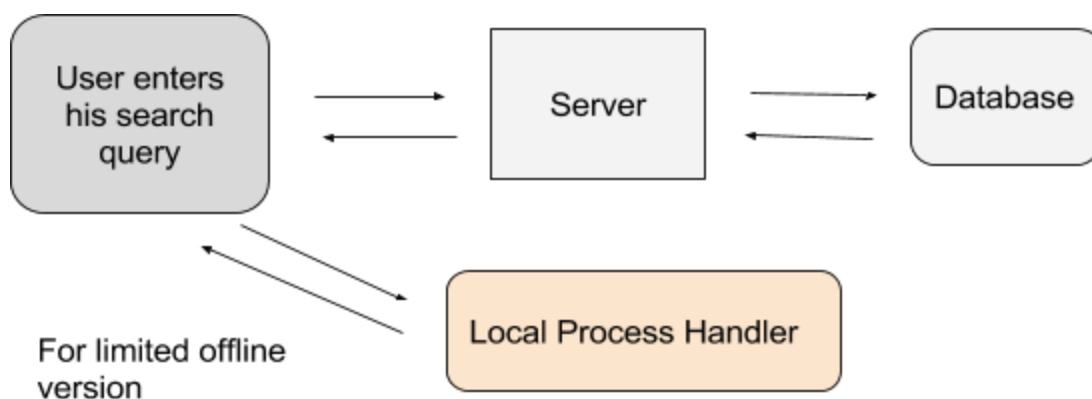
UI Mockups:

The Search Bar :

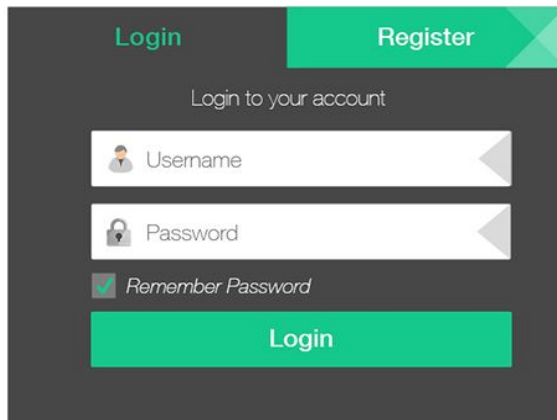


The search bar will give users the option to search for either users or words. This will be beneficial for the users and the developers on narrowing in on what they are searching for. The server will have a local process handler, allowing the search feature to work, though limitedly, even when the user is offline. The flowchart below elaborates the concept

Working of the search bar:



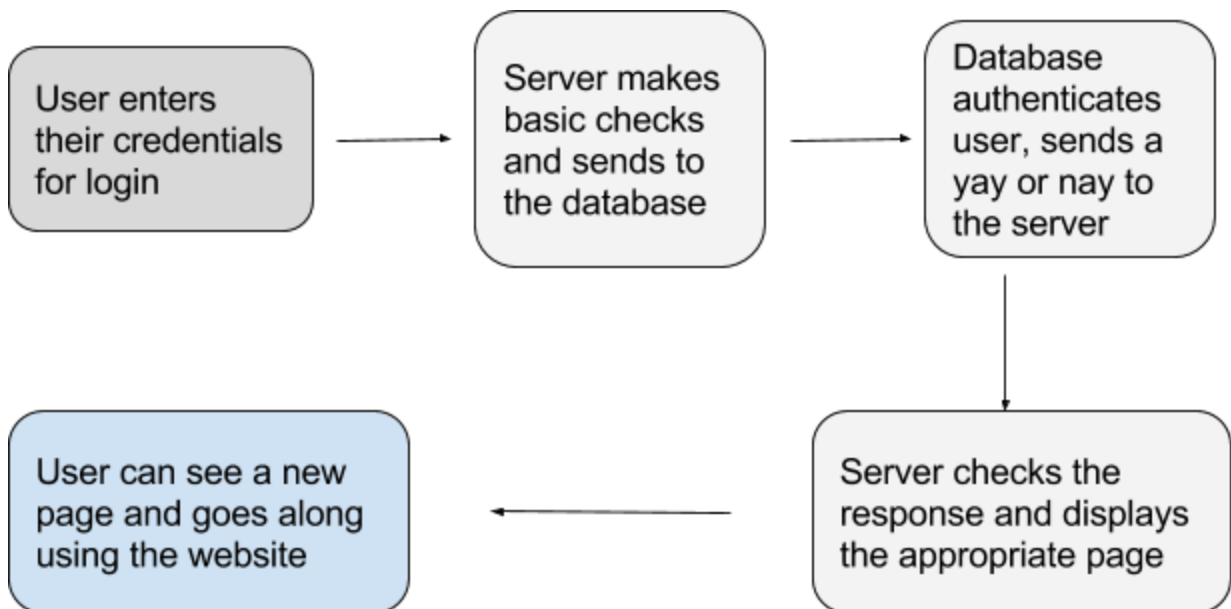
Login for the user:



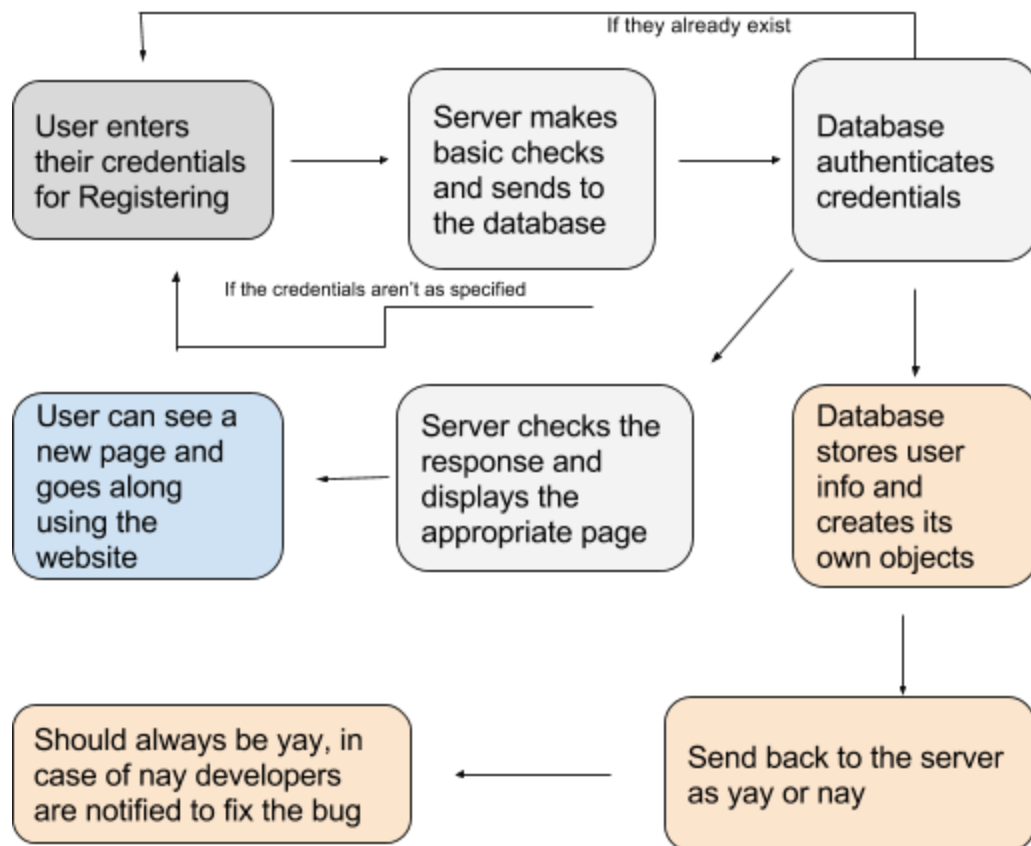
The image shows a login form with a dark grey background. At the top, there are two tabs: 'Login' (highlighted in green) and 'Register'. Below the tabs, the text 'Login to your account' is displayed. The form contains three input fields: 'Username' with a user icon, 'Password' with a lock icon, and a 'Remember Password' checkbox. A green 'Login' button is positioned at the bottom of the form.

The fields will look similar to this where the user can login using their credentials or register with a new account if they are new to the web page.

For login



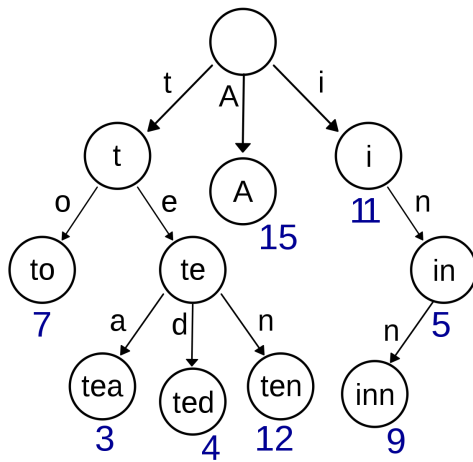
For registering



Explanation:

The website will feature login and registering features. The UI that we have in mind will look something similar to the image above. The process for login and registering a user will be quite different. This is elaborated nicely with the flow chart above. Registering a user can take more checks and more processes while login would be much simpler

Concept of word list



While most of the data for the backend will be stored in the database, the server can house a trie of all words and usernames for quick access time, low memory footprint as compared to a normal list, and the ability to offer suggestions and autocompletion in the search bar.