Tony Ly
Mark Santa Presca
Alvin Yan

Homework 4: Specifications

Development Overview

Developing AmiiboDex's prototype was challenging and took a significant amount of research and development to properly work. For the tools we used, we primarily wrote in vanilla JavaScript to provide the functionality necessary for our application to work, while we focused on using a backend service provider to implement our real time database. In particular, we used Firebase as our backend which provided us a simple to use, but challenging to implement API that allowed us to connect our user's data on their amiibos to our application.

There were many processes that took place during the development of the web application's prototype, especially when it came from a spectrum of experiences from the developers. Since the web integration with Firebase was a new concept to most of the developers, five hours of setting up the environment and reading the documentation of the Firebase platform was needed before delving into the implementation process. Later on, the flow of the web pages must be kept with consistency, especially when it comes to terms of the user session being active. By doing so, JavaScript was incorporated into the HTML files to monitor if the user is in session via Firebase functions given. This process took about three hours to maintain the proper web pages for the users to walk through. The CRUD functionalities were developed in JavaScript, where some of them were incorporated into the HTML files, while some were called upon from their own individual JavaScript files. This implementation process took the majority of the development process, with the cost being a total of fourteen hours. Later on, debugging and code cleaning took place after the main implementation of the application. There were many user testings that took place, checking to see if all functionalities were in place. In addition, the code was cleaned and commented to aid any future development of the application. This process took around six hours to complete.

Authentication Overview

One of the primary and pivotal stages upon the prototype development was to be educated about the Firebase platform. The Firebase platform provides a user database functionality for developers to work with to obtain user authentication. Furthermore, Firebase allows user authentication by various social medias, such as email, Google, Facebook, and Twitter. Nonetheless, email and Google were enabled for the web application, in part of the requirements. The Firebase platform provides various tutorials on setting up and providing integration between the web application and the platform database.

The user authentication is monitored in every webpage of the application, maintaining a consistent user session flow for each webpage. Before the authentication monitoring was implemented, many researching and reading of the Firebase platform was done to get a better understanding. This includes topics on checking up the current user, signing in and up, and logging off the current session. The monitoring was implemented by a JavaScript function provided by Firebase, indicated whether if there was an active user session in progress. If a user session wasn't in progress, the user would be redirected to the proper webpage, preventing any inconsistencies.

CRUD Overview

The Firebase platform provides many features and tools to work from to develop the CRUD features of the web application. With the user authentication provided by Firebase, creating, reading, updating, and deleting objects from the user accounts is possible. Furthermore, the database structure revolves around JSON, a database structure used specifically for JavaScript.

Each CRUD functionality was implemented to their respective webpages. Such as, the create functionality would be implemented into the 'add' page, whereas the update functionality would be implemented into the 'edit' page. Doing so, the functions were implemented in JavaScript to arrange the stored database. In each CRUD function, the JavaScript commands the assets in the Firebase storage to pull and push any changes in a formulated JSON manner. This proposed a challenge to us as, while we were still learning the basics of Firebase and what the platform has to offer.

Performance and Optimization Overview

The performance of the web application was taken to consideration, especially when it comes to terms with the user experience. Many variables from the database can take effect on the application performance, specifically the image assets. Even though there is no control over in both the server and the network throttle programmatically-wise, the time distribution of loading assets can play a crucial role for the user experience. Furthermore, the performance of the application can be affected by the CRUD features, varying the amount of assets being pulled from the Firebase database. Such as, the create and add functionality can add more assets, increasing the loading time for the users, whereas, the delete functionality can lessen the amount of assets, resulting in a decreased loading time. As a result, the application's performance can be affected by countless variables affected by both the users and the external technologies, like servers and networks.

To overcome this technological challenge, the concept of user response was taken into consideration to approach the optimization of the web application. Since users prefers a reasonable response rate, regardless of media, we decided to approach the loading issue by the "bit-by-bit" solution. That is, to load the image assets few at a time, rather than all at once.

By doing so, the user is given a constant response of each uploaded image, rather than waiting for all the images to be uploaded all at once. This solution was done with an algorithm in JavaScript to create a small amount of field assets to load every certain amount of time. As a result, this boasts the possible optimization of the application to improve user experiences with slower internet connections. The drawback of our “bit-by-bit” solution would users with fast internet speeds that would have loaded all sets of data quickly, ultimately our “bit-by-bit” solution would be slower for those users. Our solution would most benefit the low to medium end of users by creating a better experience than loading all the database information at once.

Future Overview and Setbacks

Even though many objectives were accomplished in the prototyping of the AmiiboDex web application, there are many things that needs to be fixed and looked into. Many performance challenges are still at large, requiring more time to develop and fully complete the web application’s intentions. One of the reasons for the majority of the challenges brought forth is due to the technological nature of the JavaScript language. The JavaScript language runs on an asynchronous manner, resulting many undesirable results. From an example during the development process, the for-loop and the next process both runs simultaneously. This can inflict an issue due to the variables needed to being set up by the for-loop and being used prematurely for the next process. Due to this difficulty, the JavaScript language has brought multiple challenges that need to be viewed, but its speed is what we desire in a fast application. We will also be looking to clean our code a bit more so unnecessary JavaScript is not required to be downloaded by a user for certain pages.