

By Jake Hall and Ian Donovan

## Table of Contents

Overview…………………………………………………………………………………………………..1  
Purpose…………………………………………………………………………………………………....1

Summary…………………………………………………………………………………………………..1  
Software/Hardware/Networking Components…………………………………………………………2

Pre-existing Components………………………………………………………………………………..3  
Important design decisions that were made…………………………………………………………...3  
Diagrams…………………………………………………………………………………………………..4  
Major milestones that defined the order of implementation………………………………………….6  
Goals of the project……………………………………………………………………………………….6  
Unknowns and concerns that occurred throughout the course of the project……………………...7  
Things we wished we knew before starting the project………………………………………………7  
Final Words………………………………………………………………………………………………..8  
References………………………………………………………………………………………………...8

## 

## Overview

Buddy-Buddy is a Social Media Aggregator. That is, a website which allows users to access all their news feeds and profiles for every common social media platform in today’s day and age. Rather than polluting screen space in your browser, Buddy-Buddy consolidates it all into one place. After signing in using first-party OAuth buttons, a user can check their relevant posts across all platforms, as well as make posts to the social media platforms they have logged into.

## Purpose

The concept for Buddy-Buddy came about from the frustration of using multiple social media sites. Twitter, Facebook, LinkedIn, Instagram, and others all use their own site. This causes a user to need multiple apps, browser windows, or browser tabs to keep up to date with all their sites. We chose to tackle this problem and consolidate different social medias into a single, easy to access site.

## Summary

Buddy-buddy is broken into the following pages, leveraging a variety of technologies to accomplish this. Both sides take advantage of JavaScript and jQuery to modify the page, and the implementation for each side (Facebook, Twitter) was done through different implementations. Facebook’s side was done with their first-party SDK; conversely, Twitter had no SDK, and therefore needed to be implemented using a PHP backend that wrapped around a third-party library.

#### Home Page

The Home page acts as a “welcome” to buddy-buddy. From there, a user can select which social media they would like to start with. By clicking on any of the icons, the user is brought to the respective Login pages.

#### Login Pages

The Login page is a structurally simple page containing a prompt and a login button. On clicking it the following occurs:

* An OAuth request token is created
* Flow continues until an OAuth authorization window is created
* Request token is authorized then converted to an access token
* User is redirected and able to make API calls

#### Feed Pages

The Feed pages for both Facebook and Twitter are the focus of buddy-buddy. They control and display the content of your respective platform. It includes:

* Side bar for swapping to other contexts
* Left column populated with user’s news feed
* Middle column populated with user’s profile feed
* A section for creating statuses/tweets
* A section for notifications/other data
* Asynchronous scripts to update all feeds via API

#### About Page

The About page is a simple static page to be included with buddy-buddy. It includes:

* Credit to creators/contributors
* Link to git repository
* Links to resources
* Bug report button

#### Settings Page

A Settings page was a planned part of the UI intended to visualize any user preferences across the board. This would include a few things like refresh rates, the ability to logout of specific platforms, and more. Unfortunately, only the frontend side of it was completed, and the differences between Facebook and Twitter’s API made it a bigger task than intended.

## Software/Hardware/Networking Components

**Software Components:**

* Programming Languages
  + HTML
  + CSS
  + JavaScript
* Frameworks/Libraries
  + jQuery
  + React
* APIs
  + Facebook’s Software Development Kit (SDK)
  + Twitter’s Developer API
* IDEs of the Developer's Choice
  + Atom
  + Sublime Text 3
* Web Browsers for Testing
  + Google Chrome
  + Mozilla Firefox
  + Microsoft Edge

**Hardware Components:**

* Development Machines
  + Laptops
  + Desktops

**Network Components:**

* Website Hosting
* Local Web Servers
  + LAMP/WAMP Stack

### Pre-existing Components

Many of the components we used were pre-existing as our project is a website and not a physical item. The only non-pre-existing component utilized in the development of Buddy-Buddy is the website hosting we used for testing our developed code.

## Important Design Decisions

There were many decisions made during the creation of buddy-buddy, some are general decisions while others were specifically about one of the social medias we were implementing. Some general design decisions included the choice to create a homepage which links to individual pages for logging in that are specific to the social media that they represent. A decision was also made to separate the main ‘feed’ pages of the social medias into three columns with a side bar on the left. The general outline was decided that the left two columns would be used for the ‘feeds’ and the right column for sharing, notifications, and or messages.

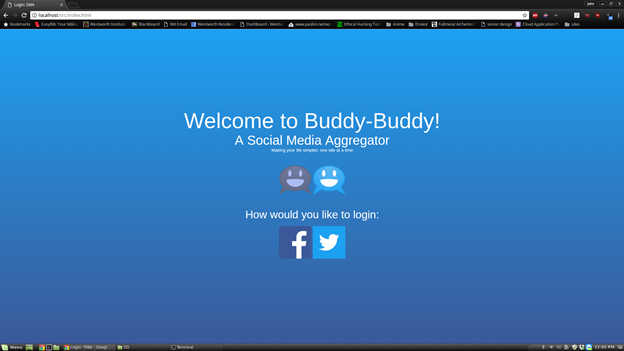
#### Facebook

For the Facebook page, there were some specific decisions that were made due to the limitations of the Facebook SDK. This included the decision to filter non-public posts, as well as split the news feed into a user’s posts on the left column and posts that they we tagged in, in the middle column. It was decided for the right column, to only implement sharing a post due to limitations of the Facebook SDK.

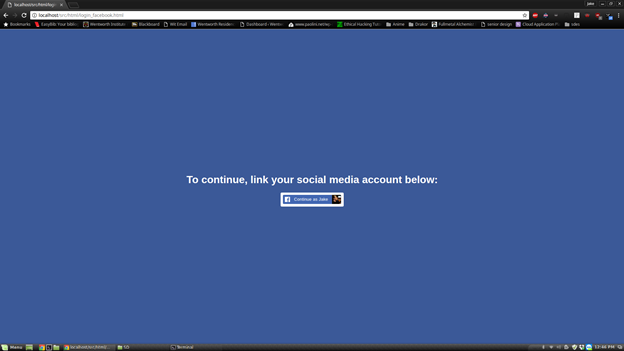
#### Twitter

On Twitter, the biggest design decision was a limitation imposed by the Twitter API. From what I could determine, there’s no actual support for notifications via API calls. The closest that Twitter provided to developers was direct user mentions, which is only half of notifications. Not only that, but it lacked a viable way to see if it was “read,” meaning that it ran the risk of being more annoying than useful. For this reason, we chose to cut it from the product.

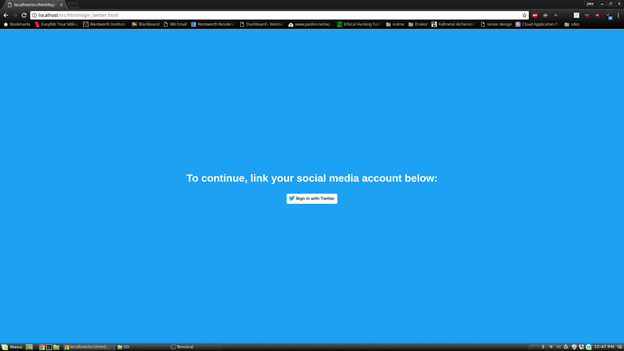
## Diagrams



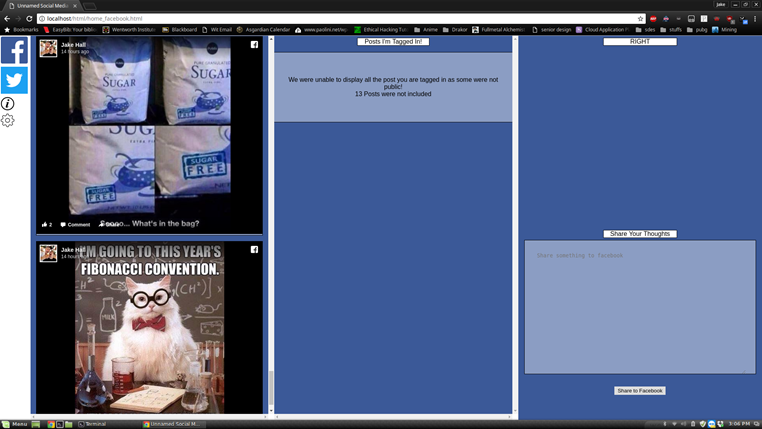
*Buddy-Buddy’s Home Page*



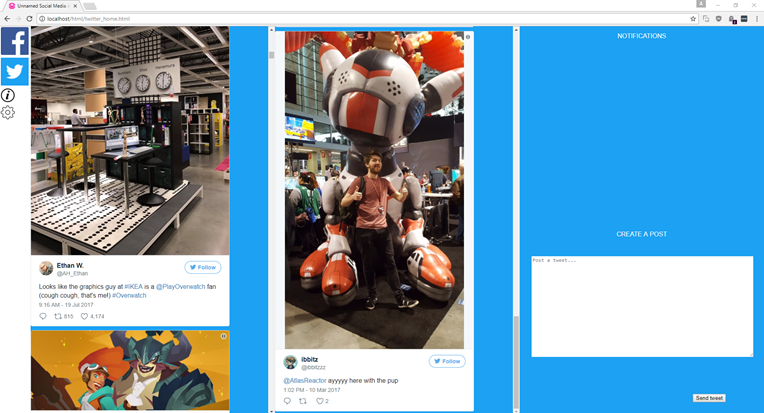
*The Facebook Login Page*



*The Twitter Login Page*



*The Facebook News Feed*



*The Twitter Feed*

## Milestones That Defined the Order of Implementation

There are a few major milestones that we set for the development of Buddy-Buddy to help assess our progress in its development:

1. Login functionality. The ability to login using your Facebook and Twitter credentials.
2. Frontend Development. A completed main page, and login/home pages for both social medias.
3. The implementation of a user’s feed into the homepage designed in milestone two.
4. The implementation of sharing functionality.

## Goals of the Project

The criteria for this project to have been considered a success were very straightforward. Buddy-Buddy will have been considered a success when it is a working website that allows users to sign in to, access, and user at least two social media platforms through our single site. Buddy - Buddy should have the ability to switch between the linked social media site, as well as make a post or posts to the connected sites. The user will also be able to disconnect/reconnect their profile from the sites as needed.

Of these goals specified in the original design document, all but one was met. Unfortunately, due to time restrictions, we were unable to put in a simple way for a user to disconnect their profile from the site. This can be done manually by clearing the cache or logging out of the social media site.

## Unknowns and Concerns That Affected Us

Like any project being started from scratch, Buddy-Buddy ran into its fair share of issues, most of which focused around the development kits from Facebook and Twitter.

#### Facebook

For the Facebook developments, an unforeseen issue arose when much of the anticipated functionality became deprecated. Unfortunately, many actions such as pulling all of a user’s posts, or sharing text ran into hitches. Workarounds were found such as filtering for only public posts or sharing something as a quote instead of text; However, a lot of time and effort was put into this that could have been better spent further developing the project.

#### Twitter

On the other hand, Twitter’s API required a huge amount of effort to implement authorization, of which there was no simple way of implementing without third-party libraries. Additionally, after deciding upon using Twitter’s first-party imbedded post system, we discovered that it posed a significant usability issue: these posts required OAuth flows for any social interaction (retweeting, liking, etc). While these interactions are easily available via API - as seen in the tweet creation UI - the time constraints to implement custom embeds and API calls for each would have been massive by comparison.

## Things We Wished We Knew Before Starting

#### Jake

Coming out of this project there are a few things that I wish that I had known going in. One of the biggest things I could have benefited from knowing ahead of time was the limitations of the Facebook SDK. If I had known, I would have chosen a friendlier social media platform or decided on a different project. Another thing I would have benefited from was to have learned website development before starting this project instead of learning it in tandem with development.

#### Ian

While Prior to creating buddy-buddy, I had no formal knowledge of web development. I had done a small amount of plain HTML/CSS, but didn’t know much outside of this. This caused multiple moments of confusion and uncertainty when determining how to decide on technologies. I needed extensive help on web hosts, setting up server stacks, best backend languages, and how to dynamically populate pages. Looking back, I would’ve benefitted from using a framework like React over raw JavaScript and jQuery. Setting up custom components would have been a massive help in getting it presentable.

Final Words

Overall, I believe that this project is successful, as we have completed the majority of functionality we set out to accomplish. Functionally, buddy-buddy can accomplish core social media functionality and present it in a logical manner. We believe that the potential for a good product is there, but the combination of time constraints, our technology stack, and limitations of the platform APIs hurt us. We were not fully able to realize the app, and in many cases had to sacrifice some of our structure/cleanliness to get things to work. Neither of us had prior web development experience, and between part-time work and other projects, we might not have had as much time to reiterate and refactor. Regardless, this was a positive learning experience; we came out with a stronger understanding of RESTful APIs, fundamental web development, and a better perspective.

Were we to revisit the project, we would have better standards in place, and potentially use a framework built for dynamic content, like React. Additionally, we would aim to remove ourselves from the someone worse features in our program - like having persistent logins through database interactions as opposed to using session variables. Lastly, we would have also benefitted from abstracting the functionality, so that both Facebook and Twitter sides would be using the same core code/method names.

References

1.Facebook SDK: https://developers.facebook.com/docs/javascript

2.Twitter Developer Portal: https://dev.twitter.com

3.TwitterOAuth: https://twitteroauth.com