

CS 5412 Final Report

INTRODUCING TOXXIC!!!

Mo Islam (mti7), Jerry Guo (jzg7)
Fabrizio Casanova (fec24)

Introduction

In today's age of digital development and increased reliance on communication, we have seen the creation, growth, and effect of a new wave of media - "online social media platforms". From building online communities to simply sending messages to acquaintances on the other side of the world, society has observed a continuous trend of the effect of social media and its platforms on billions of individuals everyday. Numerous corporations and tech-giants have created shareable, efficient, and usable platforms to increase their user base and profit from this growing industry. That being said, just like society has rules on how one should behave in public, social media utilizes a similar system to keep the discussion and communication between users civil. This makes sense as you can't really build a profit off something that causes problems for others. However, with these restrictions, also came a question - What if we created a site that not only allowed these restrictions, but actually encouraged them? That was the birth of **TOXXIC!!!**

Background & Motivation

We decided to build this project since we all had an interest in social media and web development. We know that traditional social media forums have rules and try to keep discussions/posts as civil as possible. However, there are cases where people often make rude posts and that causes a lot of chaos between a number of members. We wanted to use the cloud and its resources to build out a logical and efficient method to determine who these drivers of discourse are. Then, the users of the forum can collectively decide to evict such drivers. This is our version of collective cancellation of poor individuals. We wanted to do this since we felt it would be fun to build out a forum that encourages people to cancel others. The functionality isn't really built out in popular social media technologies, so it could be something unique and interesting to try.

Cloud Components Utilized

- Cosmos DB: Stores information on website users, posts, and comments. Multiple tables used.
- Azure App Service: Hosts website and runs background scripts.
- Azure DevOps: Automated build pipeline and hosts git repository.

Non-Cloud Components Utilized

- HTML - Used to build out the structure of the User Interface for the platform. Connected the CosmosDB, Azure data, and Flask Login system retrieved from the cloud with webpage tools (buttons, links, text input boxes etc.)
- Bootstrap - Backbone for all the stylistic elements used in the web page and subpage displays. Package applied for every subpage to make usability more fluid and display better
- Python - The backend component language. Used to initialize the Azure cloud resources we needed and connect to the cloud platform so we can see our developments and updates made in real time. All the algorithms and underlying functions for the platform were built using Python. Also, used data processing libraries when testing for the scalability of the page and measuring the efficiency of the page.

Distribution of Responsibilities

- Jerry (Backend)
 - Login system
 - Posting
 - Cancellation
 - Commenting
 - Upvote & Downvote organization algorithm
- Mo (User Interface)
 - Database System Initialization & Setup
 - User Interface
 - Main Page & Subpage Connections & Content
 - Data Display & Tracking
- Fabrizio (Testing & Integration)
 - User Interface
 - Deployment
 - Algorithms Testing
 - Website Initial Setup
 - Scalability & Efficiency Testing

Design

Cosmos DB Tables:

Users: stores users

id	password	cancelled
String name of user	String password	Bool representing whether user was cancelled

Posts: stores posts

id	title	text	upvotes	downvotes	totalvotes
Int id of post	String title of post	String text of post	Int number of upvotes	Int number of downvotes	Int representing upvotes - downvotes

Posts continued

poster	calledout	datetime	voters	archived
String name of user who made post	String name of user post is calling out	String ISO representation of time post was made	String list of names of users who have voted	Bool representing whether post is archived

Comments: stores comments

id	postid	text	poster	datetime
Int id of comment	Int id of post comment is under	String text of comment	String user that posted the comment	String ISO time comment was posted

Cancelled: stores cancelled users and the post that cancelled them

id	postid	title	group	datetime
String name of cancelled user	Int id of post that cancelled user	String title of post that cancelled user	Int number that groups users by time they were cancelled	String ISO representation of time user was cancelled

Webpages:

- Home - Contains hyperlinks to login, register, and cancelled users and posts page. When logged in, it contains hyperlinks to the page to make new posts and links to all active posts. Will display the username at bottom left of screen when logged in.
- Register - Contains text boxes to input username and password, and submission button. Submission creates a new user if the username is not taken.
- Login - Contains text boxes to input username and password, and submission button. If the username and password are valid, the user is logged in.
- Post - Contains form to create a new post. Contains text boxes for post title, post text, and who is being called out.
- Post Page - Displays information about a post. Shows post title, text, number of upvotes/downvotes, and comments. Can only be accessed by logged in users. Contains buttons to upvote/downvote if the user has not voted yet. Contains text box and submission button to write comments on the post.
- Cancelled - Displays cancelled users/posts that cancelled them.
- Uncancelled - Shown to cancelled users, cancelled users cannot access site

Cancellation Program:

At set intervals (every 2 minutes in demo for ease) we find the top upvoted post/posts and “cancel” the user that those posts call out. Those users are kicked from the website and cannot access it anymore. The top posts are archived and no longer displayed. If there are any posts that have survived two rounds of cancellation, they are removed and the users being called out are safe.

Final Product

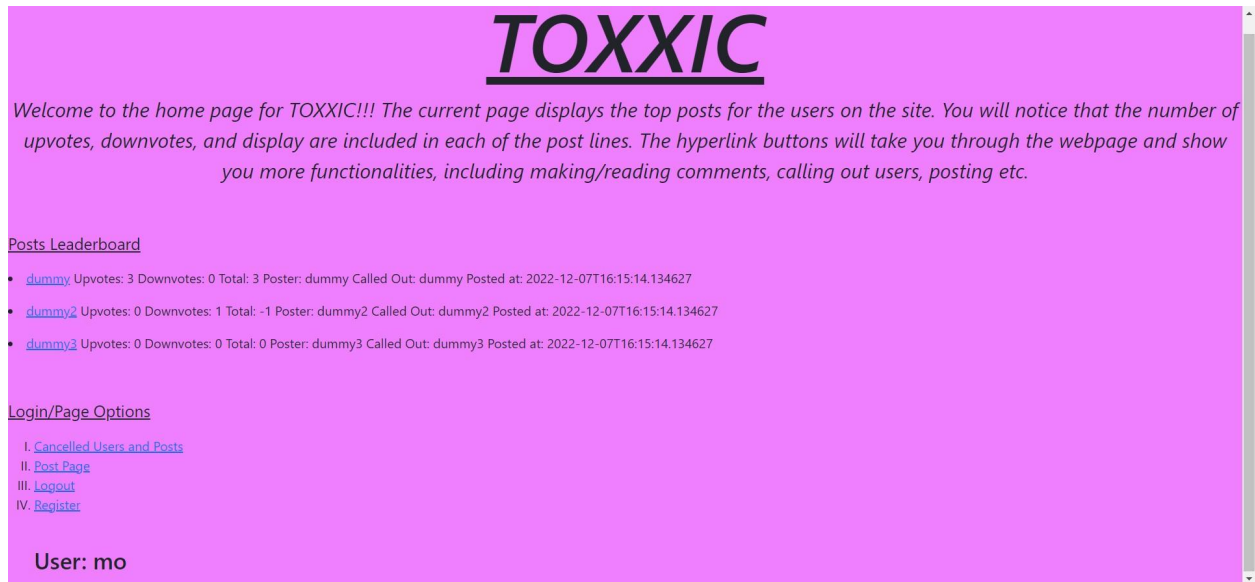


Figure 1. Home Page (Logged In)



Figure 2. Login Page



Figure 3. Register Page

TOXXIC

Create a post! Include a title, the text content, and a user to call out

Figure 4. Create Post

TOXXIC

This page displays the users that have been cancelled by the upvotes given the current active users. You can access their posts through the user hyperlink buttons and view each posts's contents.

Cancelled Users Dashboard

- [dummy](#) Upvotes: 3 Downvotes: 0 Total: 3 Poster: dummy Called Out: dummy Posted at:2022-12-07T17:27:31.709622 Group: 1
- [dummy2](#) Upvotes: 0 Downvotes: 1 Total: -1 Poster: dummy2 Called Out: dummy2 Posted at:2022-12-07T17:27:31.709622 Group: 2
- [dummy3](#) Upvotes: 0 Downvotes: 0 Total: 0 Poster: dummy3 Called Out: dummy3 Posted at:2022-12-07T17:27:31.709622 Group: 2

[Home Page](#)

Figure 5. Cancelled Users Page

TOXXIC

This page displays the post information made by a certain poster. The page displays information such as the poster, comment, called out user etc. You have the ability to make comments on their post and upvote or downvote their post.

dummy

dummy

Poster: dummy

Calling Out: dummy

☒ Upvote ☐ Downvote

Comment

[Home Page](#)

Figure 6. Post Info Page

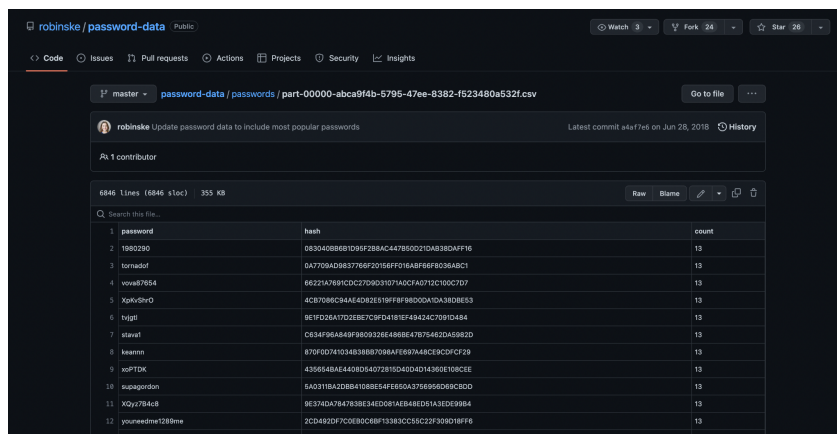
Performance Evaluations

We performed both manual and automated testing in order to test the functionality of the website. For manual testing, we tested that non-cancelled users had access to the the entire website: that is, they should be able to post, make comments, and upvote posts in order to choose who they believe should be cancelled. From using the site manually, we were each able to make posts, comments, and upvote posts from multiple accounts, so our site allows non-cancelled users full access to its functionality. Additionally, those without a registered, non-cancelled accounts, or those with a cancelled account, should not have access to site functionality. We verify this by checking whether the user is logged in and whether their account has been cancelled before they are able to view the site normally. From manual testing, this check works as expected, meaning that no one without a non-cancelled account has normal access to the site.

For automated testing, we tested the capabilities of our database by running a script registering a large number of users sequentially. We utilized this dataset containing passwords and their hashes:

<https://github.com/robinske/password-data/blob/master/passwords/part-00000-abca9f4b-5795-47ee-8382-f523480a532f.csv>.

From there, we used a script which queried the website registration page, using the ‘password’ and ‘hash’ columns from the database with the ‘password’ column being used as the username and the ‘hash’ column as the password, attempting to registering a new user with the information from this database.



password	hash	count
19802290	083040888f095f288ac447800210ab38caff16	13
turnadol	0a720a08637766f2016f016a8f66f8036abc1	13
vuael7854	86321a7698cd2708031071a0cfa0712c100c7d7	13
xpkvdlvD	4c87086c844e4d82e519f8f8b800da1da3808e93	13
rdgit	9e1f026a17028e7c9fd4181ef49424c70910484	13
staraft	0834f96a849f88032e4888e578754820a5982d	13
keason	670f0d74103483888708af8807481c9c2cf29	13
uoftDK	436654844408054072819d0d0d14360e108c3e	13
supagordon	1a0318a20b81088e5af650a3766969608c8d0	13
XQyz79Acl8	9e3f8da7847838e34cd081a8ba8ed5a3ede9984	13
yourmedme128lme	2cd482df7c080c8bf13383cc5c22f308018ff6	13
qpc038m	c70f0c0e345405830f4e372e10f071310080800	13

Figure 7. Screenshot of the dataset

During execution of the script, we looked at performance speed by keeping note of any slowdowns that may have occurred. This was done by two of us posting comments and making

posts while one other person ran the script. We also queried the database by attempting to log in using the username/password combinations created during the execution of the script.

During execution of the script, we experienced no slowdowns when using the website manually. This showed us that the performance for users posting or making comments would not be affected by users creating accounts. After executing the script, we manually attempted to log in to 20 separate accounts that were created by the script. We were able to successfully log in to all 20 accounts, which demonstrates that our database was able to maintain the user accounts we queried to be stored.

Next Steps

If we had more time to develop our project, we would focus on miscellaneous aspects that would enhance the user experience. One possible enhancement would be to increase customizability for users, such as by allowing the upload of profile pictures, profile descriptions, and allowing users to customize the look of their own homepage. In order to implement profile pictures, we would need a new database containing profile pictures and have them linked to the user which uploaded them. In order to allow for scalability, we could utilize sharding in the picture database as described in lecture. For homepage customization, the main challenge would be to make sure that the user's chosen template can be incorporated into the current design and properly integrated. One small step towards this would be to let the user choose between a set of possible templates chosen by the developers that are known to work properly.

Conclusion

In conclusion, we were able to complete almost all of our intended goals for this project. We all collectively agree that this was one of the most difficult, but rewarding projects that we have worked on while attending Cornell. We were given complete freedom from the beginning on describing what we wanted to build and what components that we wanted to utilize for the project. The final product did not have the best display we originally planned for, but still had all the intended functionality. We focused immensely on the backend and making sure the algorithms would work properly and were tested thoroughly. Throughout the semester, we gained invaluable knowledge on the application, theory, and limitless possibilities of cloud. There were definitely points of our development cycle where we disagreed and had to make compromises and adjustments to our original goals, but at the end of the day, we were able to create something pretty interesting by leveraging our programming skills and new knowledge on cloud computing. We hope you enjoy TOXXIC as much as we continue to!