



ADES DIT/FT/2B/03

Readdit

By Acosta Jan Jacob Domalanta (P2026271)

Tan Yong Rui (P2004147)

How Zu Kang Adam (P2026677)

TABLE OF CONTENTS

| | |
|--|-----------|
| ADES DIT/FT/2B/03 | 1 |
| TABLE OF CONTENTS | 2 |
| 1. Introduction | 4 |
| 1.1 Project Outline | 4 |
| 1.1.1 Project Idea | 4 |
| 2. Application Hosting | 5 |
| 2.1 Creating A Heroku Account | 5 |
| 2.2 Creating An Application | 6 |
| 2.3 Changes To Files To Take Note Of | 8 |
| 2.4 Creating and using ClearDB for cloud storage | 9 |
| 3. Web Client to Backend | 12 |
| 3.1 Web Client | 12 |
| 3.2 Backend | 12 |
| 4. Backend to MySQL Database | 13 |
| 4.1 Model, View, Controller (MVC Architecture) | 13 |
| 4.2 MySQL Database | 14 |
| 4.3 Employing Sequelize | 15 |
| 5. Self-Directed Learning | 16 |
| 5.1 ReactNative Mobile App | 16 |
| 5.1.1 Introduction | 16 |
| 5.1.2 What We Learnt From ReactNative | 16 |
| 5.1.3 Difficulties Faced While Using ReactNative | 16 |
| 5.2 Sequelize | 18 |
| 5.2.1 Introduction | 18 |
| 5.2.2 What we learnt | 18 |
| 5.2.3 Some difficulties | 20 |
| 5.3 Similar Search | 21 |
| 5.3.1 Introduction | 21 |
| 5.3.2 Levenshtein Distance | 21 |
| 5.3.3 Calculating Percentage of Similarity | 22 |
| 5.3.4 Further Improvements | 22 |
| 5.3.5 Implementations | 23 |
| 5.3.6 Limitations | 24 |
| 5.4 Sorting Algorithm | 26 |
| 5.4.1 Reddit's Sorting System | 26 |
| 5.4.1.1 Logarithmic Voting | 26 |

| | |
|---|-----------|
| 5.4.1.2 Linear Time | 26 |
| 5.4.2 Our Sorting Algorithm | 27 |
| 5.4.2.1 Linear Voting | 27 |
| 5.4.2.2 Quadratic Time | 28 |
| 5.5 Role-Based Access Control | 30 |
| 5.5.1 Introduction | 30 |
| 5.5.2 Token Creation | 30 |
| 5.5.3 Backend Restriction | 31 |
| 5.5.4 Frontend Restriction | 32 |
| 5.6 Cloudinary | 33 |
| 5.6.1 Introduction | 33 |
| 5.6.2 Takeaways From Cloudinary | 34 |
| 5.6.3 Difficulties While Using Cloudinary | 36 |
| 5.7 CSS Keyframes | 36 |
| 5.7.1 Introduction | 36 |
| 6. Teamwork | 36 |
| 6.1 GitHub | 36 |
| 6.1.1 Branches & Pull Requests | 37 |
| 6.1.2 Project Board (Scrum Board) | 38 |
| 6.1.3 Issues and Resolutions | 38 |
| 6.2 Diagrams | 38 |
| 6.2.1 ER Diagram | 40 |
| 7. Reddit Features / Manual | 40 |
| 7.1 Sign Up | 40 |
| 7.2 Login | 41 |
| 7.3 Subreddits | 41 |
| 7.3.1 Create Subreddits | 41 |
| 7.3.2 View Subreddits | 42 |
| 7.3.3 Edit Moderators | 42 |
| 7.3.4 Edit Flairs | 43 |
| 7.4 Posts | 43 |
| 7.4.1 View Posts | 43 |
| 7.4.2 Create Posts | 44 |
| 7.4.3 Comment on Posts | 44 |
| 7.4.4 Share Posts | 45 |
| 7.4.5 Report/Delete Posts | 45 |
| 7.4.6 Post Sorting | 48 |
| 7.4.7 Save Posts | 48 |
| 7.5 Search | 48 |
| 7.5.1 Basic Search | 48 |

| | |
|---|----|
| 7.5.2 Similar Search | 49 |
| 7.6 Rating System | 49 |
| 7.7 Profile | 50 |
| 7.7.1 View User Information | 50 |
| 7.7.2 View User Posts | 51 |
| 7.7.3 View User Comments | 51 |
| 7.7.4 View User Saved Posts | 52 |
| 7.7.5 Edit Account | 52 |
| 7.7.6 Access Admin Console (Admin Only) | 53 |
| 7.8 Admin Console | 53 |
| 7.8.1 View all Users | 53 |
| 7.8.2 Delete User | 54 |
| 7.8.3 Modify User | 54 |
| 7.8.4 Add User | 55 |
| 7.8.5 View Subreddits | 56 |
| 7.8.6 Modify Subreddit | 56 |
| 7.8.7 Delete Subreddit | 57 |
| 7.9 Mobile Application | 57 |
| 7.9.1 Homepage | 57 |
| 7.9.2 Search Function (Subreddits) | 57 |
| 7.9.3 Subreddits | 58 |
| 7.9.4 Post Information | 60 |

1. Introduction

1.1 Project Outline

Our team of three were tasked with the design and implementation of a 3-tier web application (Website, Backend, Database) with enough basic mechanisms to help solve the pains of our choice of problem statement.

1.1.1 Project Idea

The idea that we chose to implement into a web server was to create a clone of the popular social media platform, Reddit.

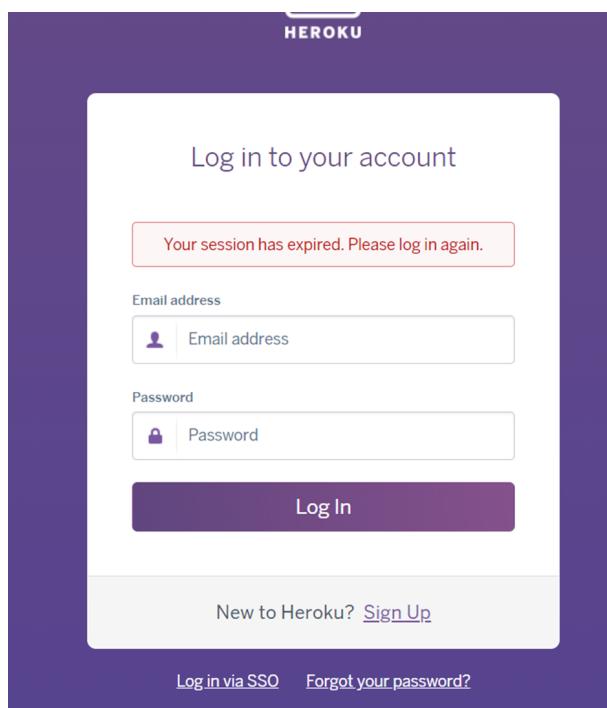
<http://reddit.com>

2. Application Hosting

2.1 Creating A Heroku Account

For hosting of our application for both frontend and backend, we have decided to use Heroku. In order to do that, you will first need an account.

Go to <https://id.heroku.com/login> and click on the **Sign Up** button shown below:



After clicking on Sign Up, fill in the Sign Up form with your details and click on **Create Free Account** shown below:

Country *

Singapore

Primary development language *

Node.js

I'm not a robot


reCAPTCHA
Privacy - Terms

CREATE FREE ACCOUNT

Signing up signifies that you have read and agree to the [Terms of Service](#) and our [Privacy Policy](#).
[Cookie Preferences](#).

An email will be sent to you to verify your email account. After verification, head over to the Heroku dashboard.

2.2 Creating An Application

Head to the dashboard and create a new app after choosing an app name that isn't taken yet.

App name

testing-app-ades

testing-app-ades is available

Choose a region

United States

Add to pipeline...

Create app

After creating an application, link the application to the github repository that has the code. You can search and select the branch that you want to deploy.

The screenshot shows the Heroku Dashboard for a new application named "testing-app-ades". The "Deploy" tab is selected. On the left, there's a section to "Add this app to a pipeline" with instructions to create a new pipeline or choose an existing one. On the right, there's a section to "Add this app to a stage in a pipeline to enable additional features" with information about pipelines connecting multiple apps. Below these are buttons for "Choose a pipeline", "Heroku Git (Use Heroku CLI)", "GitHub Connect to GitHub", and "Container Registry (Use Heroku CLI)". At the bottom, there's a "Connect to GitHub" section with a search bar for repositories and a "Search" button.

Ensure that your package.json, package-lock.json and server.js in the top directory:

The screenshot shows a GitHub repository for "frontend" with 26 branches and 0 tags. The commit history for the "frontend" branch is displayed, showing the following commits:

| File | Commit Message | Time |
|-------------------|--|-------------|
| node_modules | Committing to pull. | yesterday |
| public | added subreddit href to posts in profile.js | 2 hours ago |
| routes | Added: Design for profile.html | 3 days ago |
| .env | added function to facilitate sharing of post. also moved out all scri... | 10 days ago |
| Procfile | base files for backend with Procfile, ensure node_modules and lock fi... | 28 days ago |
| README.md | Initial commit | last month |
| package-lock.json | Committing to pull. | yesterday |
| package.json | Committing to pull. | yesterday |
| server.js | Added: Design for profile.html | 3 days ago |

Also ensure that you have a file named Procfile (no extensions) to tell Heroku the type of application you are running and what command you want heroku to run to start the application:

```
1 lines (1 sloc) | 22 Bytes  
1 web: nodemon server.js
```

In this case, I am telling Heroku I am running a web application, and run command “nodemon server.js” to start the server.

Do repeat step 2.2 again for either frontend or backend (depends on which one you did first).

2.3 Changes To Files To Take Note Of

Do remember to change your frontend calls to backend with the new Heroku backend application url.

```
//const baseurl = [ "http://localhost:3000" , "http://localhost:3001" ]  
const baseUrl = ["https://reddit-backend.herokuapp.com","https://reddit-sp.herokuapp.com"]  
  
let notifier = new AWN({icons:{enabled:false}})  
  
$(document).ready(function () {  
  
    var pathname = window.location.pathname;  
    var subReddit_path = pathname.split('/')[2];  
    var post_id = pathname.split('/')[3];  
    var retrieved_post_id;  
    $.ajax({  
        method: 'GET',  
        url: `${baseUrl[0]}/post/` + post_id,
```



There is no need to put the port number for the url as it is automatically defined by Heroku, but you still need to declare the port while starting the backend and frontend servers. To do this, you can use the value stored in process.env.PORT defined by Heroku:

```
if (process.env.PORT != null){  
  const port = process.env.PORT;  
  app.listen(port,function(){  
    console.log(`Server hosted on heroku!`);  
  });  
}
```

This port change applies to both frontend and backend.

2.4 Creating and using ClearDB for cloud storage

Note: This step requires a debit or credit card. Please add a payment method to your Heroku account before continuing with this step. We recommend creating a virtual debit card online if you don't have a card. Rest assured that ClearDB does not cost.

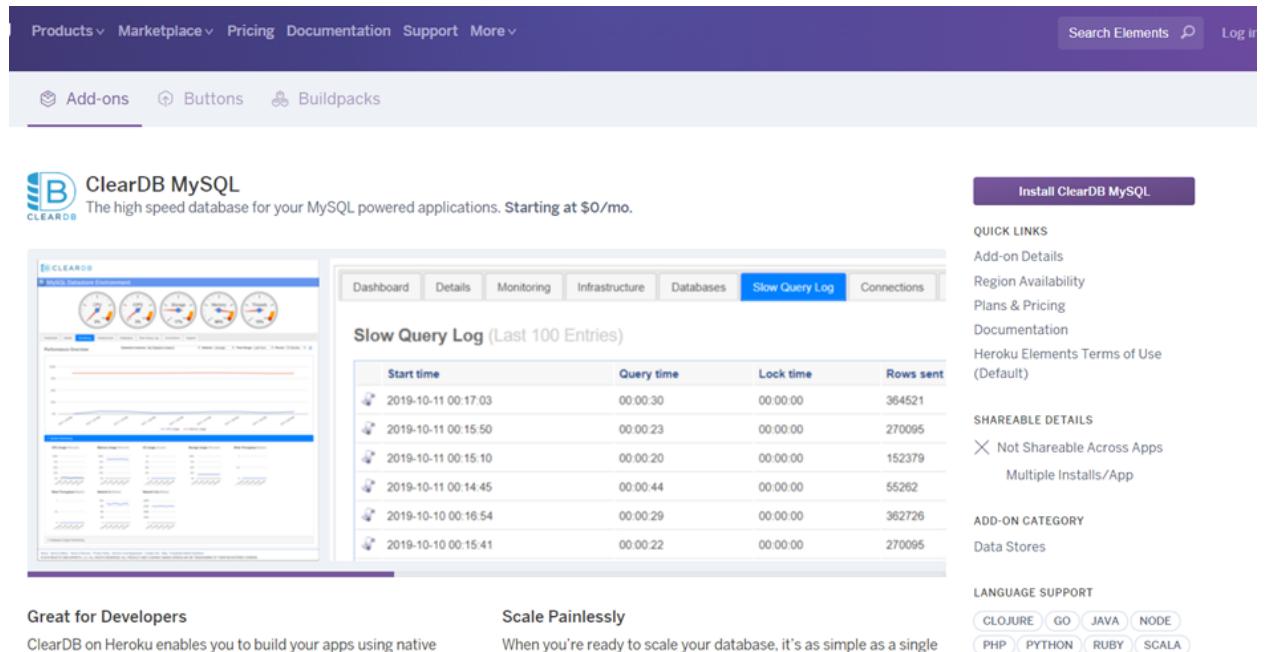
Go to your Heroku app dashboard and add new add-ons by clicking on **Find more add-ons**:

The screenshot shows the Heroku app dashboard for a specific application. At the top, there is a navigation bar with tabs: Overview (underlined), Resources, Deploy, Metrics, Activity, Access, and Settings. Below the navigation bar, there is a section titled "Dynos". A message states: "This app has no process types yet. Add a Procfile to your app in order to define its process types. [Learn more](#)". In the "Add-ons" section, there is a search bar with the placeholder "Quickly add add-ons from Elements". A message below the search bar says: "There are no add-ons for this app. You can add add-ons to this app and they will show here. [Learn more](#)". At the bottom of the dashboard, there is a table showing the "Estimated Monthly Cost" which is listed as "50.00".

Under data stores, scroll down and find ClearDB:

| | | | |
|---|--|---|---|
|  Apache Kafka on Heroku Reliable and powerful Apache Kafka as a service. |  RedisGreen Redis, Instrumented and Scaled |  Instaclustr Apache Cassa... Hosted and managed Apache Cassandra NoSQL databases |  SFTP To Go Managed SFTP/FTPS cloud storage as a service with Amazon S3 and HTTP file access |
|  Interplanetary Fission (IP... IPFS uploader for web applications |  HDrive Create and use Amazon S3, Azure Blob Storage and Google Bucket from Heroku App. |  Heroku Redis Reliable and powerful Redis as a service. |  ClearDB MySQL The high speed database for your MySQL powered applications. |
|  MSSQL | | | |

Click on **Install ClearDB MySQL**:



The screenshot shows the Heroku Elements marketplace. At the top, there's a navigation bar with links for Products, Marketplace, Pricing, Documentation, Support, More, Search Elements, and Log in. Below the navigation, there are tabs for Add-ons, Buttons, and Buildpacks, with 'Add-ons' being the active tab.

The main content area features the 'ClearDB MySQL' add-on card. It includes the ClearDB logo, the name 'ClearDB MySQL', a brief description 'The high speed database for your MySQL powered applications. Starting at \$0/mo.', and a large 'Install ClearDB MySQL' button.

To the left of the card is a preview image of the Heroku Elements dashboard, showing various performance metrics and monitoring tools.

On the right side of the card, there are several sections: 'QUICK LINKS' (Add-on Details, Region Availability, Plans & Pricing, Documentation, Heroku Elements Terms of Use (Default)), 'SHAREABLE DETAILS' (Not Shareable Across Apps, Multiple Installs/App), 'ADD-ON CATEGORY' (Data Stores), and 'LANGUAGE SUPPORT' (GLOUCE, GO, JAVA, NODE, PHP, PYTHON, RUBY, SCALA).

Below the main card, there are two additional sections: 'Great for Developers' (describing ClearDB as enabling native app development) and 'Scale Painlessly' (describing the ease of scaling the database).

Type in the name of your Heroku App and click on the desired App, then click **Submit Order Form**:

The screenshot shows the ClearDB MySQL add-on page on the Heroku Elements Marketplace. It includes the ClearDB logo, a brief description, a link to the Marketplace, an 'Add-on plan' dropdown set to 'Ignite - Free', a search bar for provisioning to an app named 'testing', and a list of provisioning results showing one entry for 'testing-app-adcs'. A note about the Salesforce Master Subscription Agreement is present, followed by a 'Submit Order Form' button.

To get your ClearDB credentials, go to the Settings tab and scroll down to Config Vars, then click on **Reveal Config Vars** and copy the entire string:

The screenshot shows the 'Config Vars' section of the Heroku Settings tab. It displays a table with two rows: 'CLEARDB_DATABASE_URL' and 'heroku_1c89f72eef4896a'. The URL contains a yellow-highlighted database URL and an orange-highlighted database name. A 'Hide Config Vars' button is visible in the top right corner.

Example of copied string:

mysql://b7a6c1ee0950ab:3ee893d6@us-cdbreast-04.cleardb.com/heroku_1c89f72eef4896a?reconnect=true

The yellow highlight denotes the username, the green highlight denotes the password, the light blue highlight denotes the database URL/hostname and the orange highlight denotes the database name that **cannot** be changed. Ensure that your dump files that are going to be imported to ClearDB have their database names changed.

You can now replace your backend code's database connection details with ClearDB's hostname, username, password and database name:

```
// heroku credentials, comment as necessary. ensure data
// const database = "heroku_1c89f72eef4896a";
// const user = "b7a6c1ee0950ab";
// const password = "3ee893d6"
// const host = 'us-cdbreast-04.cleardb.com'
```

3. Web Client to Backend

3.1 Web Client

To start, we used HTML and CSS to create the application's structure and design. More importantly, we used a combination of jQuery's Ajax and Axios to send HTTP requests. As per our requests to the backend, we used the four cardinal request methods (i.e. GET, PUT, POST, DELETE) to perform retrieve, edit, create and delete operations in the backend.

After the backend responds to the request, we then use Javascript to dynamically display and design information.

Our JavaScript files are placed in a separate folder than in the html as a script tag as users with itchy fingers can edit the scripts in the script tags, which can alter and grant access to buttons and views that the average user is not supposed to see. There are also checks in the JavaScript files to redirect a user away from views they are not supposed to access.

Finally, we utilized the Bootstrap framework to expedite the process of making aesthetically pleasing designs

3.2 Backend

For the backend, we built it using Node.js and the Express framework. Which is what we were taught and what we are most familiar with.

Even though the frontend has its Javascript files to prevent users from accessing pages they are not supposed to (such as administrative tools), it is also good practise to secure the backend as well for functions reserved for users with higher levels of authority, such as moderators or administrators. We will touch on more about our Role Based Access Control in Section 5.5.

Regarding our backend's structure, we tried our best to follow through with the following guidelines, [Best Practices for REST API design](#) (John and Ryan, 2020). Specifically following the points 'Use Nouns instead of verbs in Endpoint Path', 'Use Logical Nesting on Endpoints' and 'Handling Errors Gracefully'.

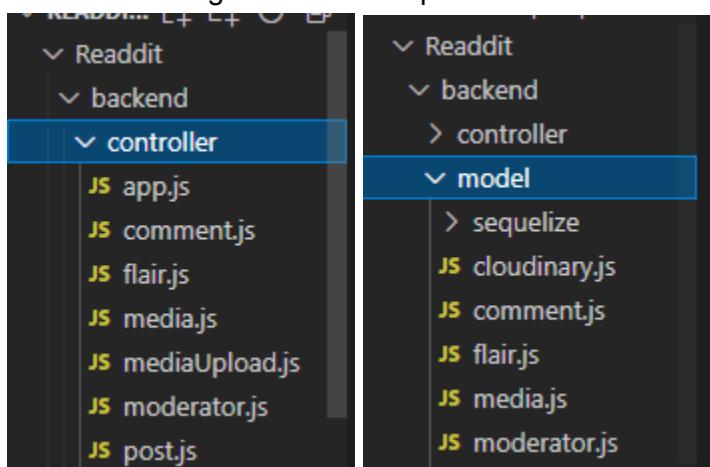
4. Backend to MySQL Database

4.1 Model, View, Controller (MVC Architecture)

For Readdit, we followed the MVC architecture.

As a short recap, MVC is a system that separates **an application into three main logical components**: the model, the view, and the controller. The Model handles all logic related to data (database and SQL statements). The View corresponds to the UI components that the user will see and interact with. And the Controller is responsible for handling requests received from the View and sending responses back to the view after applying business logic to data retrieved from the Model.

Below is an image of how we separated our View and Controller directories.



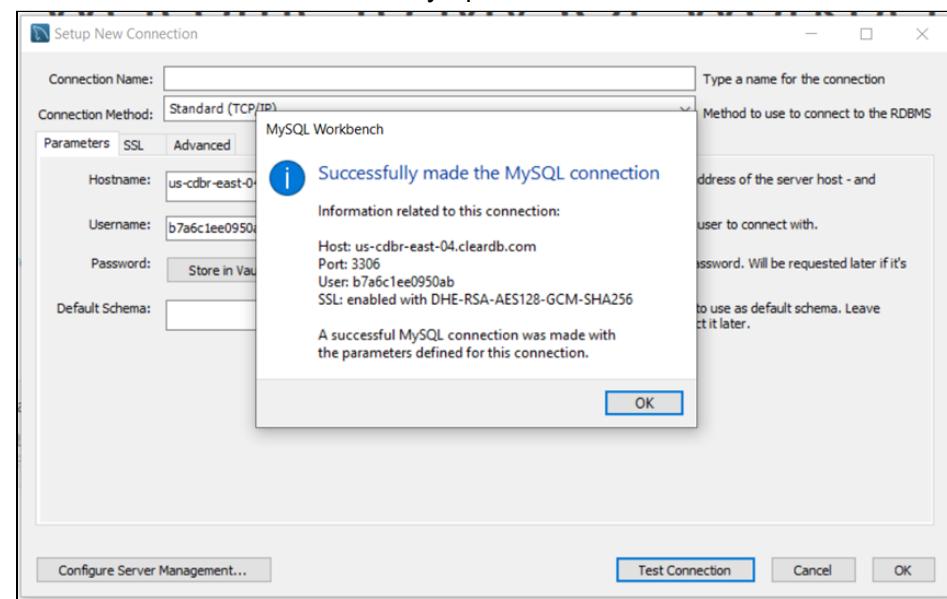
4.2 MySQL Database

For our database of choice, we chose to employ ClearDB MySQL Database. ClearDB is a free and online database that is hosted on a cloud and endorsed by Heroku.

Therefore, saving us the rigmarole of having to host and provision our own database on one of our personal servers.

We connected our created ClearDB database to our Heroku App to generate ClearDB's hostname, username, password and database name.

By entering the ClearDB Credentials into MySQL Workbench, we were able to access the data in the ClearDB Database via Mysql Workbench.

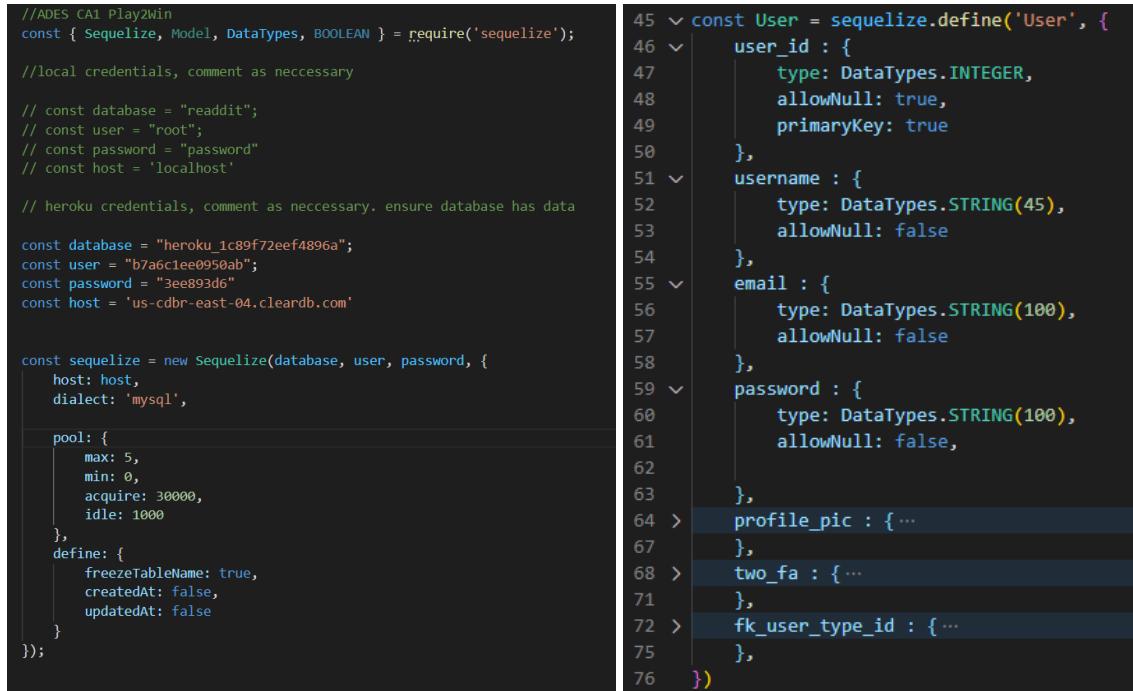


The screenshot shows the MySQL Workbench interface for a database named 'Reddit-ClearDB'. The 'Navigator' pane on the left lists tables such as 'comment', 'comment_vote', 'content_type', 'flair', 'fifa', 'fifa_votes', 'moderator', 'post', 'post_vote', 'report', 'saved', 'user', 'user_type', and 'subreddit'. The 'Tables' section is expanded, showing sub-tables like 'comment_vote' and 'content_type'. The 'Schemas' section shows the current schema is 'heroku_1c89f72eeef4896a'. The 'post' table is currently selected. A query editor at the top shows the command: 'SELECT * FROM heroku_1c89f72eeef4896a.posts'. Below the query is a 'Result Grid' displaying 10 rows of data. The columns in the grid are: post_id, fk_subreddit_id, fk_user_id, title, content, fk_flair_id, pinned, and created_at. The data includes posts like 'Diluc discussion', 'Ayaka Showcase (No buffs!)', 'My top two anime scenes', 'Castlevania anyone?', 'Discord user TYRPRO is trending worldwide!', 'evarhastet', 'Goku Chibi Review', 'A lot of game characters', 'Bilboes Playboy Diluc Whitman buys new Man...', 'Luck of the Draw', 'The wife of a taxi driver who survived an explo...', and 'Chopard L.U.C. Tech Strike One Limited Edition'. The 'Output' pane at the bottom shows the execution log with two entries: '23:17:03 SELECT * FROM heroku_1c89f72eeef4896a.user LIMIT 0, 1000' and '23:17:11 SELECT * FROM heroku_1c89f72eeef4896a.post LIMIT 0, 1000'. The log indicates 4 rows returned.

4.3 Employing Sequelize

Instead of the usual MySQL queries, we chose to implement Sequelize for the sending of queries from the backend to the database.

Sequelize is a promise-based Node.js Object-Relational Mapping tool. This is the [link](#) to their extensive documentation that we followed to use Sequelize. But in a nutshell, the core of Sequelize revolves around **models**. From what we understand, models are an abstraction of the tables inside the database, and we run methods/functions stemming from these models to interact with the database (similar to the JavaBeans that we learnt last semester).



```
//ADES CA1 Play2Win
const { Sequelize, Model, DataTypes, BOOLEAN } = require('sequelize');

//local credentials, comment as neccessary

// const database = "readdit";
// const user = "root";
// const password = "password"
// const host = 'localhost'

// heroku credentials, comment as neccessary. ensure database has data

const database = "heroku_1c89f72eef4896a";
const user = "b7a6c1ee0950ab";
const password = "3ee893d6"
const host = 'us-cdbr-east-04.cleardb.com'

const sequelize = new Sequelize(database, user, password, {
  host: host,
  dialect: 'mysql',

  pool: {
    max: 5,
    min: 0,
    acquire: 30000,
    idle: 1000
  },
  define: {
    freezeTableName: true,
    createdAt: false,
    updatedAt: false
  }
});

45  const User = sequelize.define('User', {
46    user_id : {
47      type: DataTypes.INTEGER,
48      allowNull: true,
49      primaryKey: true
50    },
51    username : {
52      type: DataTypes.STRING(45),
53      allowNull: false
54    },
55    email : {
56      type: DataTypes.STRING(100),
57      allowNull: false
58    },
59    password : {
60      type: DataTypes.STRING(100),
61      allowNull: false,
62    },
63    },
64    profile_pic : { ... },
65    },
66    two_fa : { ... },
67    },
68    fk_user_type_id : { ... },
69    },
70    },
71    },
72    },
73    },
74    },
75    },
76  })
```

Image: Initializing Database Connection

Image: Our User model

The image on the left shows us initializing the sequelize object with our database credentials. While the image on the left is an example of us defining the User **model**.

While we do agree that Sequelize provides many benefits like speeding up development and securing the code (i.e. Sequelize protects against SQL injections as it uses parameterized queries and filters the data for us), using an ORM has introduced some shortcomings which we will talk about more in section 5.2.

5. Self-Directed Learning

5.1 ReactNative Mobile App

5.1.1 Introduction

ReactNative allows you to code in JavaScript and run your application on either iOS or Android (MacOS required for iOS). Designing the application is very akin to HTML and CSS, which will appeal to programmers who have experience in web development.

While developing an application for Readdir, we used back the same endpoints that were used in the web version of Readdir, which is great for code efficiency.

5.1.2 What We Learnt From ReactNative

Through ReactNative, we have learnt how to create an application with multiple screens that are able to load their content from external sources (CloudDB, Cloudinary).

We also learnt how to detect for specific key inputs, such as the enter button, which is used to submit a user's search request.

5.1.3 Difficulties Faced While Using ReactNative

Firstly, there is the issue of the application not waiting for the backend server to respond with data before trying to render it, which keeps throwing errors left and right. After hours of research, we found that we can tell ReactNative to render data conditionally:

```
<Text style={styles.postTitle}>
  {isLoading ? <ActivityIndicator size="large"/> : item.title}
</Text>
```

Let's focus on the line of code in between the `<Text></Text>` tags. This line of code tells the application to check if `isLoading` is true or false. If it is, show an `ActivityIndicator` (loading spinner). If not, display the post's title. This checking of `isLoading` is widely used across the application as the majority of the content is loaded from online resources.

We also tried implementing the same image, GIF and video support, but only images were supported natively. GIF support natively was broken on the current

ReactNative version I was working on, hence we had to find another module to handle GIFs.

Fortunately, we found FastImage, which was able to support both images and GIFs, and was almost exactly used the same way as the native Image component in ReactNative.

However, we were not able to find any modules with working video playback support, with the large majority only working on Expo or iOS devices. This is one major drawback of ReactNative, where modules that promise functionality may not work as intended. Meanwhile, we will try our best to bring video playback support in CA2.

5.2 Sequelize

5.2.1 Introduction

After being recommended (by our lecturer) to attempt using an ORM for our application. We endeavoured to use Sequelize as our backend's daily driver for sending queries to the database.

To begin, although I would like to say that we learnt a lot from using ORMs - we learnt a lot about the ORM format and the usefulness of the model system as a form of database abstraction - we didn't get here without some difficulties.

5.2.2 What we learnt

First let's begin with the obvious, Sequelize is helpful as it speeds up development dramatically. We have to write less code comparatively and clears up the time needed to think up of an efficient and adequate SQL query.

As an example, here are two images. The one on top uses the base MySQL library while the one below uses Sequelize. Both images are functions attempting to retrieve all categories **but** the bottom one requires much less syntax (e.g. getting connection) and doesn't require us to come up with a SQL query.

```
getAllCategories: function (callback) {
  var conn=db.getConnection();
  conn.connect(function (err) {
    if(err) {
      console.log(err);
      return callback(err,null)
    }
    else {
      console.log("Connection Successful");
      var sqlQuery = `SELECT * FROM sp_games.category`;
      conn.query(sqlQuery,function (err,result) {
        conn.end();
        if(err) {
          console.log(err);
          return callback(err,null);
        }
        else {
          return callback(null,result);
        }
      })
    }
  })
}
```

Image: Base MySQL function

```
284     getAllCategories: function (callback) {
285       Category.findAll().then(function (result) {
286         callback(null,result)
287       }).catch(function (err) {
288         callback(err,null)
289       })
290     }
291   }
```

Image: Sequelize Function

Next, Sequelize is much easier to update and maintain.

Have a look at this example:

```
284   284     getPost: function (post_id, callback) {
285   285       Post.findOne({
286   286         attributes: ['post_id', 'title', 'content', 'pinned', 'created_at'],
287   287         where: { post_id: post_id },
288   288
289   289         include: [
290   290           {
291   291             model: User,
292   292             attributes: ['username']
293   293           },
294   294           {
295   295             model: Subreddit,
296   296             attributes: ['subreddit_name', "subreddit_id"]
297   297           },
298   298           {
299   299             model: Flair,
300   300             attributes: ['flair_name', "flair_colour"]
301   301           },
302   302           {
303   303             model: Post_Vote,
304   304             attributes: ['vote_type']
305   305           },
306   306         ],
307   307       }
308   308     }
309   309   }
```

This is an image of a function that retrieves a post from our database.

Initially, we didn't have a rating system so we didn't have a Post_Vote table.

After implementing the rating system all we had to add was line 302 to line 305 in order for us to retrieve the votes associated with the post. In comparison, imagine attempting this via SQL query.

We would have to add an inner join and table1.col = table2.col to every single function. Not the most difficult, but very time consuming compared to the Sequelize method.

Lastly, another small benefit that Sequelize has is protection against SQL injection. Sequelize has an inbuilt parameter filter and also uses parameterized queries. I say that this is a small benefit as we're already used to using parameterized query and we also created our own middleware function to check users' input in our previous module.

```
    savePost: function (post_id, user_id, callback) {
      Saved.create({
        fk_post_id: post_id,
        fk_user_id: user_id
      }).then(function (result) {
        return callback(result, null);
      }).catch(function (err) {
        return callback(null, err);
      })
    },
  ],
}
```

5.2.3 Some difficulties

One of the drawbacks that we encountered was the initial loss in productivity due to the time taken to get over Sequelize's initial learning curve. Having to memorise the library's functions and syntax, then having to teach it to other group members took a lot of time. In our case, it took 2 - 3 weeks (out of 6 weeks) before we were confident in our ability to write Sequelize code. Although I'm sure this would help in the long run (for learning other ORMs), I feel like this process really hampered our ability to start writing code at the beginning.

Another drawback that I encountered was the loss of understanding of the SQL language after playing with Sequelize. Sequelize has the ability to 'super' simplify the query making process as we fully bypass the time and effort it takes to craft a custom made SQL query. While the simplification of query crafting is useful in expediting the process of development, I fear that using this tool has made us lose some essential time to practice writing SQL queries while we're still studying.

An analogy for this would be using a calculator before mastering mental arithmetic. Likewise, I think we should have more practice on writing SQL queries or finish our Data Engineering module before we're allowed to be using 'calculators' which seem to be using 'joins', 'sub-selects' and multiple-inserts.

```
Executing (default): SELECT `Saved`.`saved_id`, `Saved`.`fk_post_id`, `Post`.`post_id` AS `Post.post_id`, `Post`.`title` AS `Post.title`, `Post`.`content` AS `Post.content`, `Post`.`pinned` AS `Post.pinned`, `Post`.`created_at` AS `Post.created_at`, `Post->User`.`user_id` AS `Post.User.user_id`, `Post->User`.`username` AS `Post.User.username`, `Post->Subreddit`.`subreddit_id` AS `Post.Subreddit.subreddit_id`, `Post->Subreddit`.`subreddit_name` AS `Post.Subreddit.subreddit_name`, `Post->Post_Votes`.`post_vote_id` AS `Post.Post_Votes.post_vote_id`, `Post->Post_Votes`.`vote_type` AS `Post.Post_Votes.vote_type` FROM `Saved` AS `Saved` LEFT OUTER JOIN `Post` AS `Post` ON `Saved`.`fk_post_id` = `Post`.`post_id` LEFT OUTER JOIN `User` AS `Post->User` ON `Post`.`fk_user_id` = `Post->User`.`user_id` LEFT OUTER JOIN `Subreddit` AS `Post->Subreddit` ON `Post`.`fk_subreddit_id` = `Post->Subreddit`.`subreddit_id` LEFT OUTER JOIN `Post_Vote` AS `Post->Post_Votes` ON `Post`.`post_id` = `Post->Post_Votes`.`fk_post_id` WHERE `Saved`.`fk_user_id` = '1';
```

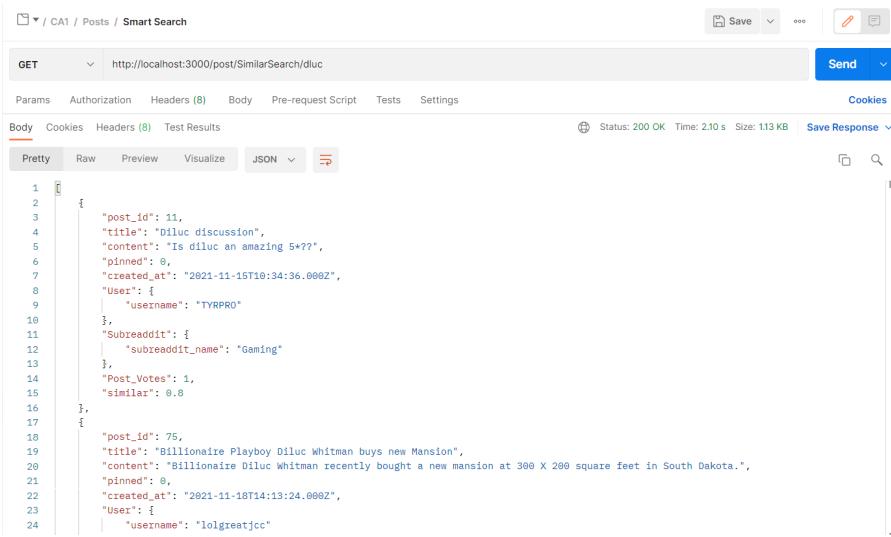
Image : Needlessly complicated query that we didn't expect

5.3 Similar Search

5.3.1 Introduction

One of our implemented features is an algorithm that searches our database for all posts with titles similar to our input string.

Example:



The screenshot shows a Postman interface with a GET request to `http://localhost:3000/post/SimilarSearch/dluc`. The response status is 200 OK, and the JSON data is as follows:

```
1  [
2    {
3      "post_id": 11,
4      "title": "Diluc discussion",
5      "content": "Is diluc an amazing 5*??",
6      "pinned": 0,
7      "created_at": "2021-11-15T10:34:36.000Z",
8      "User": {
9        "username": "TYRPRO"
10      },
11     "Subreddit": {
12       "subreddit_name": "Gaming"
13     },
14     "Post_Votes": 1,
15     "similar": 0.8
16   },
17   {
18     "post_id": 75,
19     "title": "Billionaire Playboy Diluc Whitman buys new Mansion",
20     "content": "Billionaire Diluc Whitman recently bought a new mansion at 300 X 200 square feet in South Dakota.",
21     "pinned": 0,
22     "created_at": "2021-11-18T14:13:24.000Z",
23     "User": {
24       "username": "lolgreatjcc"
25     }
26 }
```

The basis of this algorithm is actually a math algorithm widely known as Levenshtein Distance or Edit Distance.

5.3.2 Levenshtein Distance

Levenshtein Distance was founded by the Russian scientist, Vladimir Levenshtein to calculate the similarities between two strings by calculating the number of ‘moves’ it takes to transform one string to the other.

Hence, the smaller the Levenshtein Distance, the more similar the strings are.

Example:

To transform ‘MAN’ into ‘WOMAN’, you would require two moves.
Hence the Levenshtein Distance between the strings will be 2.

To transform ‘MAN’ to ‘MANDATORY’, you would require six moves.
Hence the Levenshtein Distance between the Strings will be 6.

5.3.3 Calculating Percentage of Similarity

Now that we have a basic understanding of the Levenshtein Formula, let's get into how we applied it to our search algorithm. This is the formula we used:

```
var num = (longerLength - editDistance(longer,  
shorter)) / parseFloat(longerLength);
```

This formula subtracts the Levenshtein Distance from the length of the Longer String and then calculates that number as a number of the Longer String's Length.

This will return a number from 0 - 1, 0 meaning that it is not similar at all (Since the Levenshtein Distance is = to the number of letters in the word) while 1 meaning that the strings are 100% identical (Since Levenshtein Distance is 0)

Example:

'DLUC' and "DILUC" will return a value of 0.8, meaning that it is 80% similar since the Levenshtein Distance of the strings is 1. Since only 1 character is removed from 'DILUC', which is 5 characters long. It means that % of the characters in 'DILUC' is untouched, which is also 80%.

5.3.4 Further Improvements

However, just this percentage alone is not enough, since it assumes that the compared strings are both single words. In that case, where a sentence or phrase is input, this percentage will be inaccurate.

Thus we resolved this issue by splitting the sentence into words using `.split(" ")`, and then running the percentage function for each word, and finally the full sentence. If the score of the string is above 0.4 (At least 40% similar), we will add it to a results array.

Both strings will be split and we compare every occurrence of the split string for as many results to be added to the results array as possible.

Finally, we output the average of all scores in the results array as the final score of the string. Since the string will be compared to every post title in our database,

if the string's final score is above 0.4 (At least 40% similar), we add the corresponding record to our return endpoint to send to the frontend.

Example:

Comparing “Dog” to “I am a doggy dog”

“I am a doggy dog” → [‘I’, ‘am’, ‘a’, ‘doggy’, ‘dog’], where “Dog” will be compared with every element in this array and finally compared with “I am a doggy dog” itself, totalling 6 rounds.

Only the scores of “doggy” (0.6) and “dog” (1) will be added to the results array. Thus the final score for the similarity between “Dog” and “I am a doggy dog” will be 0.8, allowing it to be added to the list of similar results.

5.3.5 Implementations

Although we have a list full of records similar to our input string, our job is still not over.

Since we’re implementing this similar search along with a basic search, we will need to remove all occurrences from both searches from the similar search so that the similar search will only display results the basic search fails to retrieve.

Loops to remove duplicate results from similars array

```
for (var i = 0; i < similars.length; i++) {  
    var duplicate = false;  
    for (var count = 0; count < posts.length; count++) {  
        if (similars[i].post_id == posts[count].post_id) {  
            duplicate = true;  
        }  
    }  
}
```

Next, we will also need to order the results in the similar post such that the most similar results will appear first, while the least similar results are at the bottom.

```

function arrangeSimilars(arr) {
    for (var i = 0; i < arr.length; i++) {
        if (i == arr.length - 1) {
            break;
        }
        if (arr[i].similar < arr[i + 1].similar) {
            var tmp = arr[i];
            arr[i] = arr[i + 1];
            arr[i + 1] = tmp;

            i = -1;
        }
    }
    console.log(arr);
    return arr;
}

```

This function passes in an array and rearranges them based on their similarity attribute in descending order.

```

success: function (data, status, xhr) {
    var similar = arrangeSimilars(data);
}

```

The function is called at the top of the success function in the ajax call.

With those two sorted out, we now have a working frontend similar search.

The screenshot shows a search interface with a purple header bar containing a search input field, a magnifying glass icon, and a 'Profile' button. Below the header, there are tabs for 'Subreddits' and 'Posts', with 'Posts' being the active tab. The main content area is titled 'Displaying Search Results for luc'. It lists three posts from different subreddits:

- 1 r/Gaming • Posted by u/TYRPRO • 6 days ago
Diluc discussion
- 0 r/News • Posted by u/lolgreatjcc • 3 days ago
Billionaire Playboy Diluc Whitman buys new Mansion
- 0 r/News • Posted by u/lolgreatjcc • 3 days ago
Luck of the Draw

Below these results is a section titled 'Similar results' containing one post:

- 6920 r/Tech • Posted by u/TYRPRO • 3 days ago
Chopard L.U.C. Tech Strike One Limited Edition

5.3.6 Limitations

However, this similar search is not without its limitations. Since the algorithm relies heavily on the Levenshtein Distance formula, the limitations of the Levenshtein Distance formula apply to the algorithm as well.

Example:

Our formula will view ‘BILION’ to be more similar to ‘EDITION’ than ‘BILLIONAIRE’.

Although ‘BILION’ is mathematically more similar to ‘EDITION’ than ‘BILLIONAIRE’, humans don’t perceive this similarity the same way. Thus this is one limitation of our Algorithm.

Link Search for Post / Subredit Profile

Subreddits Posts

Displaying Search Results for bilion

No results found

Similar results

- ↑ r/Tech • Posted by u/TYRPRO • 3 days ago
6920 Chopard L.U.C. Tech Strike One Limited Edition
- ↑ r/News • Posted by u/TYRPRO • 3 days ago
6920 Billionaire Mogul Thomas Cross found dead
- ↑ r/News • Posted by u/folgreatjcc • 3 days ago
6920 Billionaire Playboy Diluc Whitman buys new Mansion

Thus results containing ‘EDITION’ are put on top of results containing ‘BILLIONAIRE’.

5.4 Sorting Algorithm

5.4.1 Reddit's Sorting System

Let me preface this by showing you the research we performed on Reddit's (our competitor) 'Hot' algorithm.

$$f(\text{seconds}, \text{sign}, n) = \log_{10} n + \frac{\text{sign} * \text{seconds}}{45000}$$

Where **n** is the total rating of the post (i.e. Upvote - Downvote) and **seconds** is the time in seconds since the first post. For simplicity's sake let us assume **sign** is equal to positive one.

First let us break down the formula into two parts, the part of the function before the plus and the part of the function after the plus.

5.4.1.1 Logarithmic Voting

As you can see from the function above Reddit's 'Hot' algorithm evaluates the total rating of the post logarithmically.

$$\log_{10} n$$

In essence, this means that the value/worth of votes reduces as the value of n grows. In layman terms this means that the first 10 upvotes have the same weight as the next 100 and the 100 upvotes hold the same weight as the next 1000.

Now, why is this bad? In our opinion, we think that votes should be created equal and shouldn't be worth less as more votes get added. Simply because we want to make users feel as if their votes actually matter and actually decide the sorting of posts.

5.4.1.2 Linear Time

Next, is the seconds divided by 45 000 or 12.5 hours. In essence, this just means that a post needs to have 10 times as many votes as another post which is 12.5 hours earlier. But this is besides the point, what I want to focus your attention on how it is calculated.

$$\frac{\text{sign} * \text{seconds}}{45000}$$

As seen from the image above, the confidence rating of a post is calculated linearly with respect to time. This means that the time it takes for the post in 'Hot' to drop is quite slow comparatively (i.e. the posts high up in the 'Hot' section will most likely be the same 8-12 hours later) . Of course, we could just change the value of 45 000 but we chose to go for a different approach.

5.4.2 Our Sorting Algorithm

So how did we solve the problems we had with Reddit's algorithm? Based on our research and knowledge of Mathematics we have come up with our own sorting algorithm which attempts to solve our issues with Reddit's algorithm.

$$f(v, \text{time}) = \frac{v}{(\text{time})^2}$$

Where v is equal to the total rating of the post (i.e. Upvote - Downvotes) and time is equal to the time in hours since the post's submission.

5.4.2.1 Linear Voting

As seen **above**, when calculating the rating of the post we take into account the total rating of the post linearly. This means that compared to Reddit's algorithm (which takes votes logarithmically), we instead take them into account linearly. This in essence solves our issue with their algorithm since every vote will be created equally since the first vote will have the same worth as the thousandth.

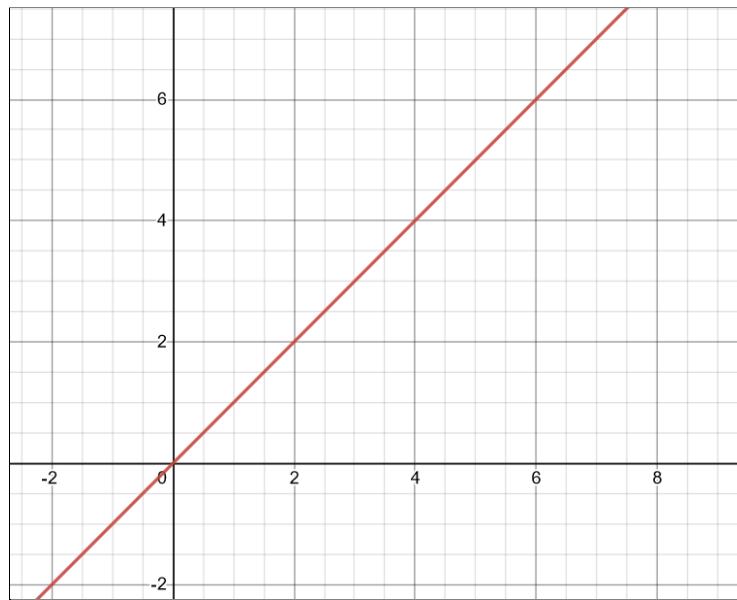


Image: $y = x / (1^2)$

The above graph shows how the post's confidence rating grows (in our formula) as the number of votes grows given that the variable of time doesn't change (i.e. one hour since post submission).

5.4.2.2 Quadratic Time

Regarding our issue with the speed by which posts in Reddit's 'Hot' section are rotated, we solved it by calculating the rating through quadratic time.

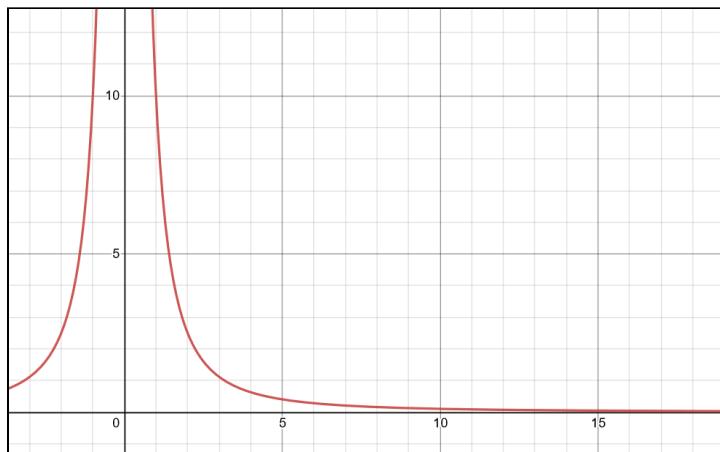


Image : $y = 10 / (x)^2$

As you can see from the above graph as time goes on (given the same rating), the post's confidence rating drops quickly but later the gradient levels off.

We see this as the best possible solution since the post that users will see rotates quickly giving way for newer posts but still greatly values user's votes as after most posts have aged, higher voted posts will still generally have a higher confidence rating and thus be put higher up.

5.5 Role-Based Access Control

5.5.1 Introduction

We also implemented Role-Based Access Control to differentiate two types of users, Admin and User roles.

| | user_type_id | user_type |
|---|--------------|-----------|
| ▶ | 2 | Admin |
| ● | 1 | User |
| * | NULL | NULL |

5.5.2 Token Creation

We make use of JWT tokens to help verify and authenticate our users. This token is generated on every login.

```
login: function (email, password, callback) {
    User.findAll({
        attributes: ['user_id', 'username', 'fk_user_type_id'],
        where: {
            [Op.and]: [
                { email: email },
                { password: password }
            ]
        }
    })
}
```

The login function gets the user info of the specified user using their email and password.

```
var token = jwt.sign(
    // (1) Payload
    {
        user_id: result[0].user_id,
        type: result[0].fk_user_type_id
    },
    // (2) Secret Key
    config.key,
    // (3) LifETIME of token
    {
        //expires in 24 hrs
        expiresIn: 86400
    }
);
```

If successful, a new token will be created by signing the values of the user id and user role, encoding it with a secret key and then setting its lifetime.

```
var secret='s12xyz00'; //your own secret key
module.exports.key = secret;
...
```

This secret key is defined in our config.js, and it will be called again whenever we want to extract information from the token.

5.5.3 Backend Restriction

We restricted the access of users from using admin only endpoints using our own middleware function, verify.js

Verify.js contains three functions for different uses, but the function we use for verifying admins is checkAdmin

```

checkAdmin : function (req, res, next){
  if (typeof req.headers.authorization !== "undefined") {
    // Retrieve the authorization header and parse out the
    // JWT using the split function

    let token = req.headers.authorization.split(" ")[1];
    //console.log('Check for received token from frontend : \n');
    //console.log(token);

    jwt.verify(token, config, (err, data) => {
      console.log("data extracted from token \n", data);
      if (err) {
        console.log(err);
        return res.status(403).send({ "message": "Unauthorized access", errCode: 1 });
      } else {
        if (data.type != 2){
          console.log("Not an admin!");
          return res.status(403).send({ "message": "Unauthorized access", errCode: 2 });
        }
        else{
          req.body.token_fk_user_type_id = data.type;
          next();
        }
      }
    });
  } else {
    res.status(401).send({ "message": "Unauthorized access" });
  }
},

```

The token we created contains the user's role id. Thus this function extracts the role id from the endpoint's Authorization Header and checks its value.

This is how we ensure that these endpoints are only used by those with the valid role id.

```

//delete user
app.delete('/users/:userid', verify.checkAdmin ,printDebugInfo, function (req, res) {
  var userid = req.params.userid;

  user.delete(userid, function (err, result) {
    if (!err) {
      res.status(204).send({"Result" : result});
    } else {
      res.status(500).send({"message":"Error deleting user." });
    }
  });
});

```

Delete user is an admin only endpoint, thus we secure it using verify.checkAdmin

5.5.4 Frontend Restriction

In the frontend, we restrict user's access to certain pages by checking their role id from the website's local storage.

When users log in, we store the created jwt token in the local storage.

| Key | Value |
|------------------|--|
| userInfo | {"user_id":1,"username":"Klyntar5","fk_user_type_id":2} |
| pb_exceptions | {"null":{"lastUpdatedAt":"2021-11-22T10:01:42.861Z","exceptions":[]}} |
| pb_user_activity | {} |
| token | eyJhbGciOiJIUzI1NilsInR5cCl6IkpxVCJ9eyJ1c2VyX2lkjoxLCJ0eXBIIjoyLCip... |
| pb_forms | {"null":{"lastUpdatedAt":"2021-11-22T10:01:42.861Z","forms":[]}} |
| pb_notifications | {} |

Whenever the user accesses an admin only page, we send a call to the backend to verify the user's role using `verify.checkAdmin`

```
$(document).ready(function () {
    try {
        var userData = localStorage.getItem('userInfo');
        var token = localStorage.getItem("token")
        // userData = userData.slice(1,-1);

        var userJsonData = JSON.parse(userData);

        $.ajax({
            headers: { 'authorization': 'Bearer ' + token },
            url: `${baseUrl[0]}/verify`,
            type: 'GET',
            contentType: "application/json; charset=utf-8",
            dataType: 'json',
            success: function (data, textStatus, xhr) {
                console.log(data);
            },
            error: function (xhr, textStatus, errorThrown) {
                window.location.assign(` ${baseUrl[1]}/home.html`);
            }
        });
    } catch (error) {
        window.location.assign(` ${baseUrl[1]}/login.html`);
    }
})
```

If the user is not an admin, he will be redirected back to the home page. Thus this makes sure that only admins can access these pages.

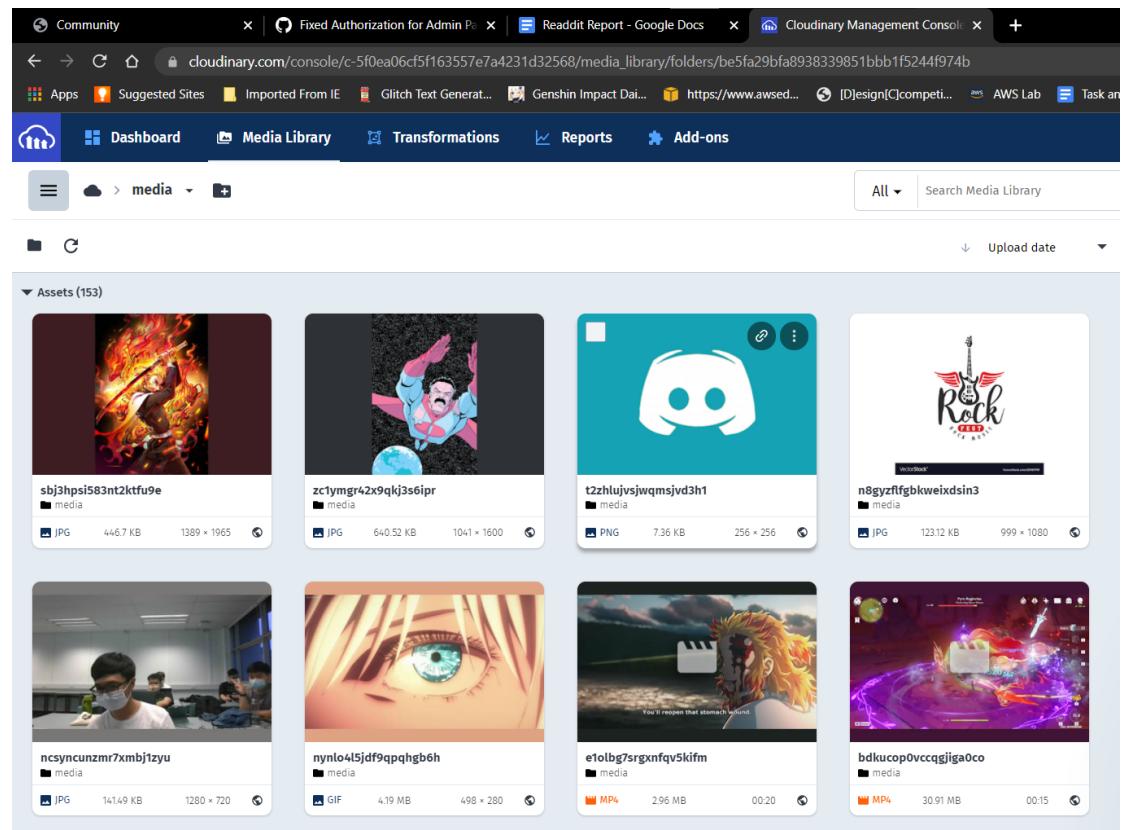
5.6 Cloudinary

5.6.1 Introduction

Cloudinary is an online media storage platform that is free for everyone to use with the free tier. We will be storing all of our media (Images, GIFs, Videos) in Cloudinary.

5.6.2 Takeaways From Cloudinary

Through using Cloudinary, I have learnt how to upload images, GIFs and videos to an online cloud platform. All of our application's media (save for the header logo) are stored in Cloudinary and fetched when needed)



I also learnt how to use new modules to process the media uploads, as well as to check for a file's type and size. This is important when verifying file uploads to ensure that the application doesn't try to display a PDF in a video or image HTML tag, or to ensure that the Cloudinary upload doesn't fail due to the file being too large.

I also learnt how to transform images using Cloudinary to return rounded images, which was useful for profile images.

Uploaded image in Cloudinary:

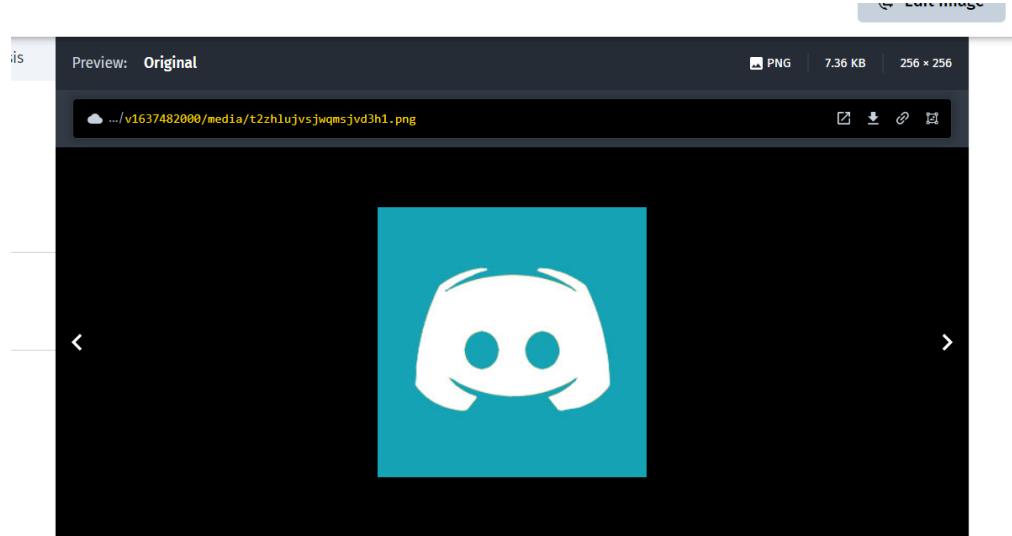
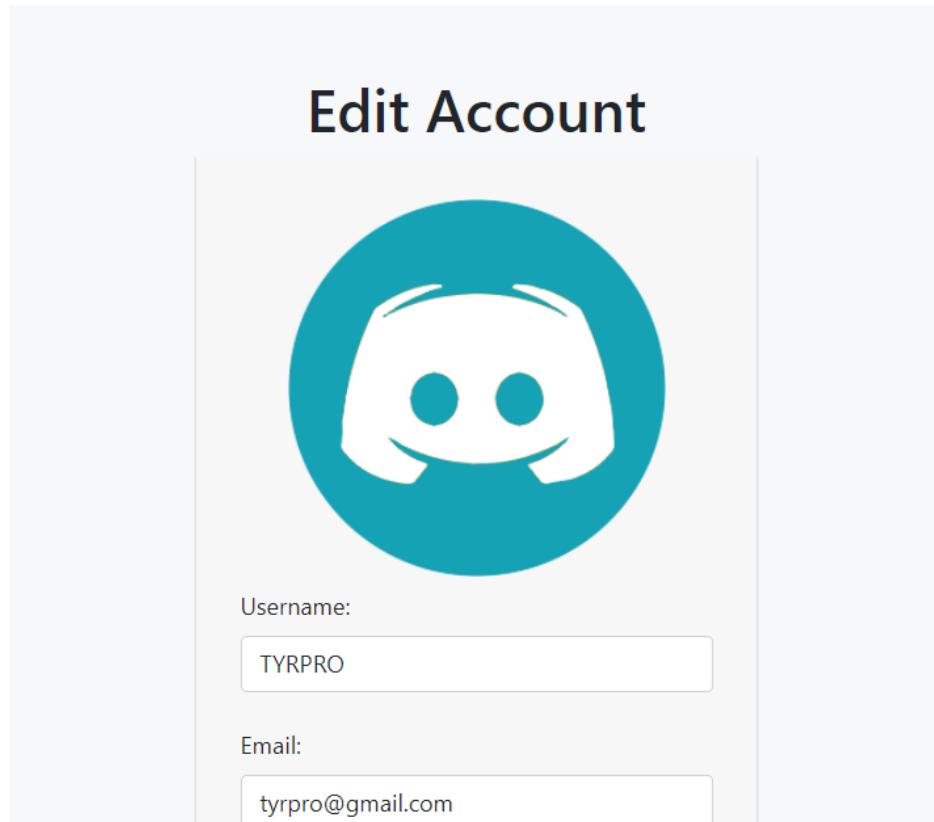


Image after Cloudinary transformation:



You could argue that the profile image could be formatted to achieve the same transformation via CSS, but having Cloudinary do it means that all the frontend client has to do is load the image, which saves some processing time.

5.6.3 Difficulties While Using Cloudinary

I needed to accept file uploads from the user, but files cannot be passed from the frontend to the backend using JSON data type. After doing some research, I settled on WebFormData, which allows me to pass user uploads to the backend server.

Even after passing the WebFormData to the backend, I was having trouble extracting the data from said WebFormData. Doing further research led me to find a npm module named multer, which was used to parse the WebFormData in the backend.

From these two incidents, I now know that WebFormData must be used in conjunction with multer for me to successfully implement file upload support.

A user might also accidentally submit a file type that is not a media type, such as PDFs and documents. They might also upload files that are too large for Cloudinary to support with the free tier. I ran into continuous issues of uploads failing as I did not know that Cloudinary had file size limitations.

After doing some research, I found that Cloudinary has a 10MB and 100MB max file size for images and videos respectively. In order to check for the file size, I had to use the fs (file system) import, which was foreign to me, to check for the file's size.

After some trial and error, checks for file size and file types were in place, and users were notified if the file size was too large or the file type was unsupported.

5.7 CSS Keyframes

5.7.1 Introduction

Due to our image uploading feature in our posting feature, one of the drawbacks we encountered is that the uploading of posts took a considerable amount of time.

Therefore we had to come up with a way to show users that their posts **were being uploaded**. And then a way to tell them once their post **has been uploaded**. Of course, we can add a simple text, which says “post being uploaded” and “post successfully posted” but that felt inelegant and visually boring. Therefore, I took this opportunity to learn how to animate using CSS and learn about keyframes. We will talk more about this in the presentation.

6. Teamwork

6.1 GitHub

Our project planning tool of choice was GitHub. GitHub enabled us to organize our work by allowing us to view the current progress of the project is, what everyone is currently working on, and what still needs to be done. We felt that our focus on GitHub and project planning increased our work efficiency significantly as we were always able to quickly comprehend the current status of the project. Below are some of GitHub's project planning tools that we used. [Link to our GitHub repository](#).

6.1.1 Branches & Pull Requests

Firstly, we decided to use GitHub's branches (and pull request) system for any new features or issues that we were developing on. Using different branches instead of pushing to the central branch allows us to 'isolate' code and thus quickly figure out where an issue/bug stemmed from.

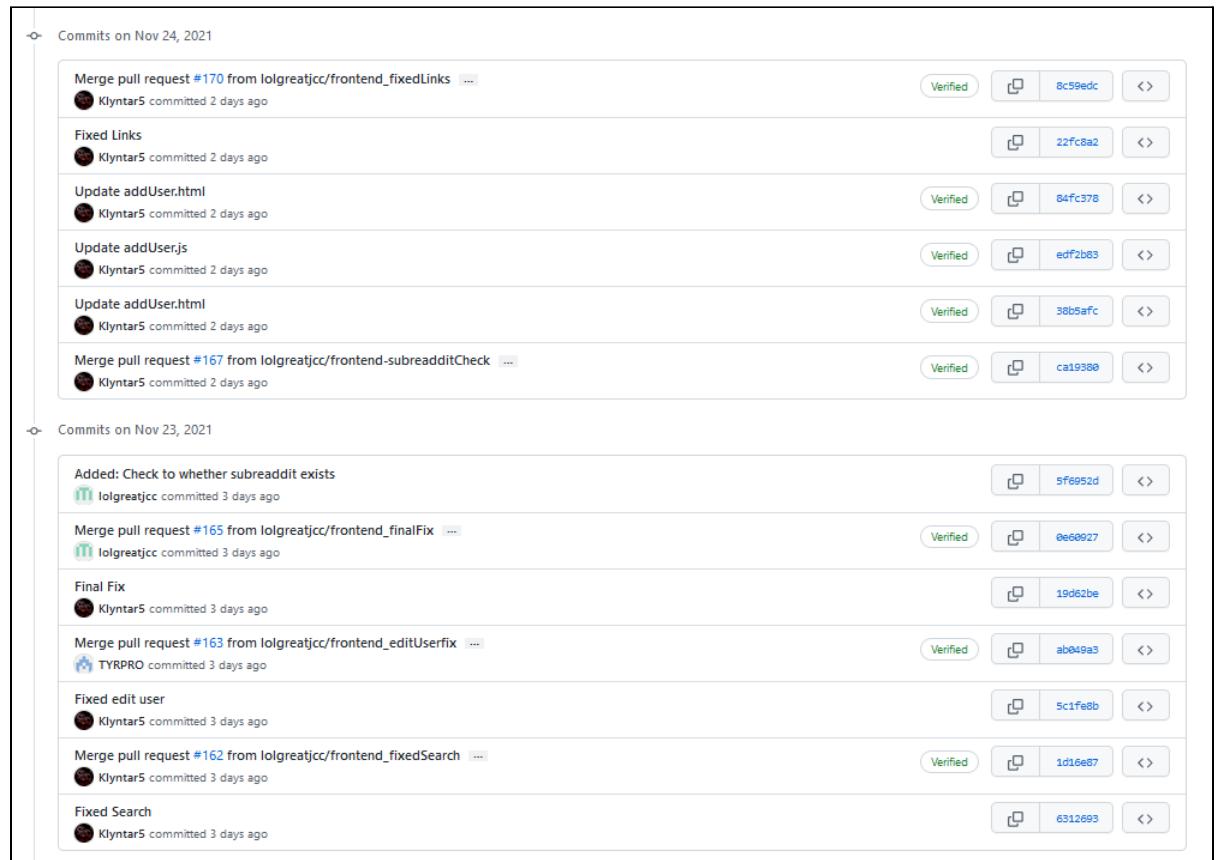


Image: Pull Request Commits

6.1.2 Project Board (Scrum Board)

Next, we decided to utilize GitHub's project boards as our Scrum board. Having a Scrum board was indispensable as it quickly allowed us to find out who to ask regarding the machinations of a specific feature. Therefore, accelerating our feature integration process. Furthermore, having a Scrum board helped us visualize the progress of the project. This allows us to quickly decide whether we need to speed up to reach the project deadline or slow down to produce higher quality work. [Link to our Scrum Board](#).

6.1.3 Issues and Resolutions

We also heavily utilized GitHub's issues page. Using GitHub Issues increases the efficiency at which we fix bugs since all our application's bugs are able to be seen from one page. Another thing that helped us is its heavy integration with GitHub. GitHub Issues allows us to refer to pull requests and other users. We even incorporated a rule which states that once an issue is fixed have a short description of how the issue was solved and **refer to the pull request which implements the fix.**

| | | |
|---|--|---|
| #139 by TYRPRO was closed 5 days ago | | |
| <input type="checkbox"/> <input checked="" type="radio"/> Missing admin authorization checks #138 by TYRPRO was closed 5 days ago | | 2 |
| <input type="checkbox"/> <input checked="" type="radio"/> Some of our endpoints retrieve user_id from the localStorage value and not the auth token. #135 by lolgreatjcc was closed 5 days ago | | 2 |
| <input type="checkbox"/> <input checked="" type="radio"/> Clicking on carousel or media on homepage leads to post #133 by TYRPRO was closed 6 days ago | | 1 |
| <input type="checkbox"/> <input checked="" type="radio"/> The rating system and saving system doesn't work on the Home page. bug #128 by lolgreatjcc was closed 6 days ago | | 1 |
| <input type="checkbox"/> <input checked="" type="radio"/> The rating system and Save feature hasn't been implemented to post.html #118 by lolgreatjcc was closed 6 days ago | | 1 |
| <input type="checkbox"/> <input checked="" type="radio"/> Subreddit.js shows every post to be saved even when not all are saved. bug #117 by lolgreatjcc was closed 7 days ago | | 1 |
| <input type="checkbox"/> <input checked="" type="radio"/> Non-admins and non-moderators are able to access pages they aren't supposed to have access to. bug #116 by lolgreatjcc was closed 6 days ago | | 2 |
| <input type="checkbox"/> <input checked="" type="radio"/> Subreddit search tab shows similar posts for similar results instead of similar subreddits bug #108 by TYRPRO was closed 7 days ago | | 1 |
| <input type="checkbox"/> <input checked="" type="radio"/> Post still uploads even when file size is too big bug #102 by Klyntar5 was closed 7 days ago | | 1 |
| <input type="checkbox"/> <input checked="" type="radio"/> Add a loading bar for post creation. enhancement #101 by Klyntar5 was closed 7 days ago | | 1 |
| <input type="checkbox"/> <input checked="" type="radio"/> When the window is small, the media file usually goes over its container. bug #100 by lolgreatjcc was closed 7 days ago | | 1 |
| <input type="checkbox"/> <input checked="" type="radio"/> Some API calls in the frontend are still using the temporary user_id. bug #97 by lolgreatjcc was closed 7 days ago | | 1 |
| <input type="checkbox"/> <input checked="" type="radio"/> Admins are able to delete themselves in admin tools. Will be fixed by Adam once core is complete. bug #96 by lolgreatjcc was closed 7 days ago | | 5 |

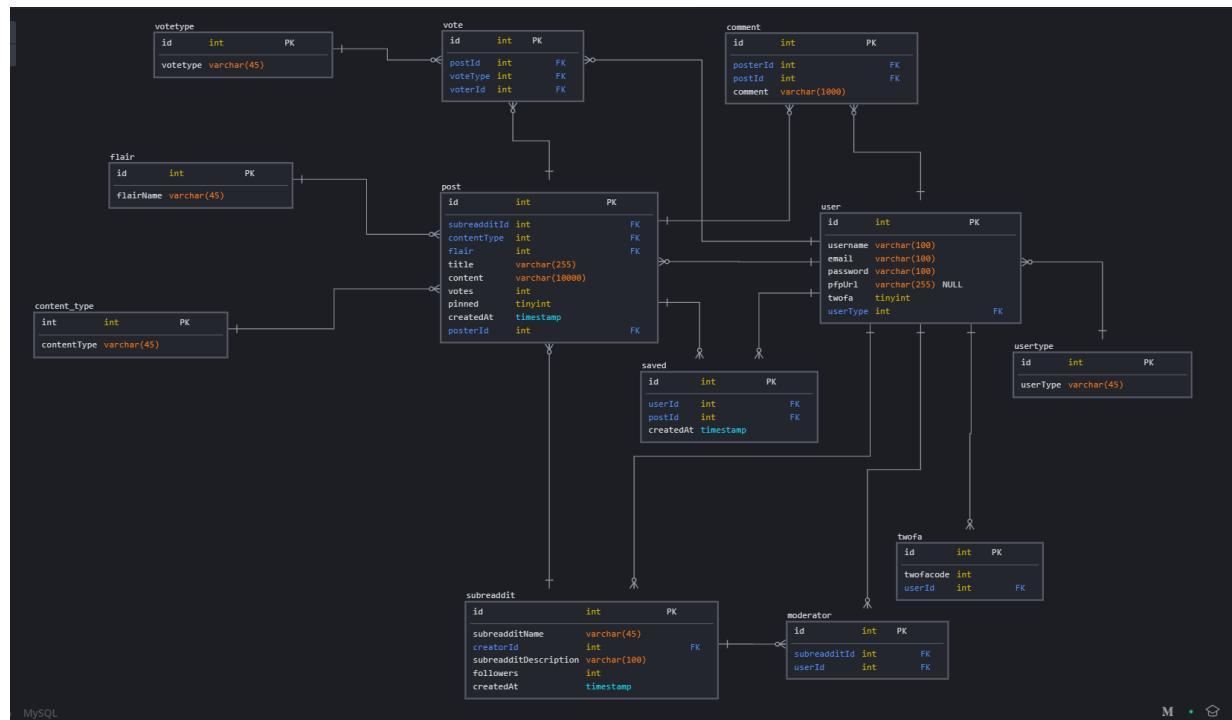
The assign feature was also very useful as at all times we could tell who was working on a specific feature so that two persons wouldn't implement the same fix.

6.2 Diagrams

Firstly, we'd like to start by stating that we know of the existence of other useful diagrams (e.g. UI Flow and Sequence Diagrams). But considering that our team is small and none of the other aspects was too complex, we decided that crafting other diagrams wasn't warranted.

6.2.1 ER Diagram

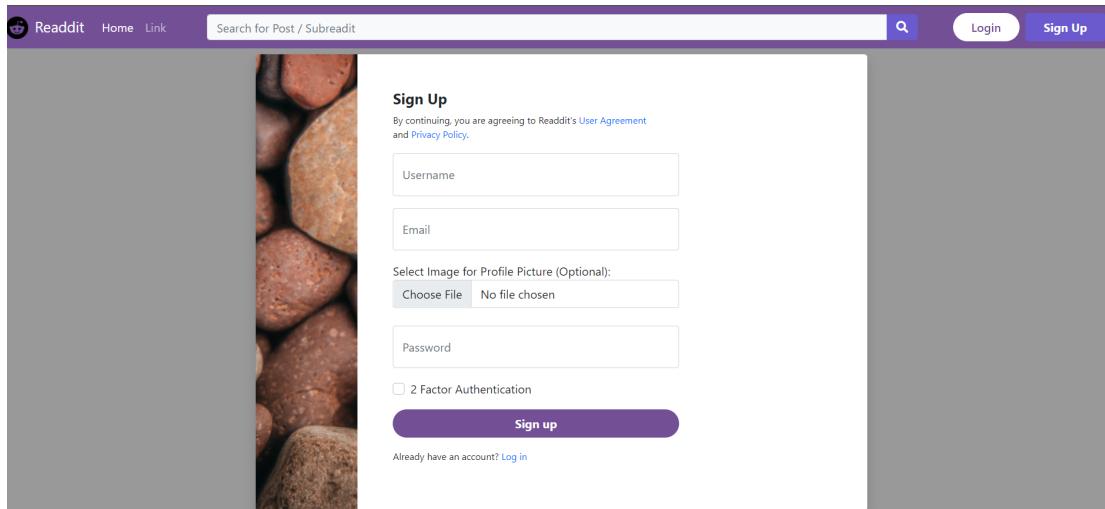
One of the aspects that did feel complex was our database. With our many tables and relationships, we felt compelled to create an ER diagram. Below is an image of our first ER diagram.



Creating an ER diagram made database creation less stressful since from the start of development we were all on the same page on what the database should look like. Furthermore, the ER diagram is helpful as we are using the Sequelize ORM which meant that we had to make an abstraction of the database tables in Sequelize. Having the ER diagram to refer to expedited the model creation process.

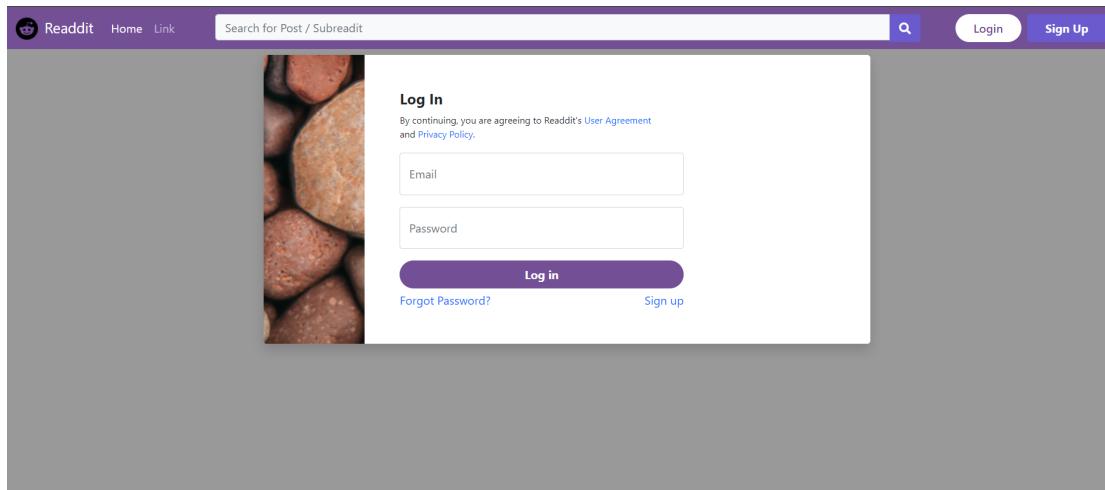
7. Reddit Features / Manual

7.1 Sign Up



Users can sign up for a reddit account here, where they are required to input a username, password, email and an optional profile picture.

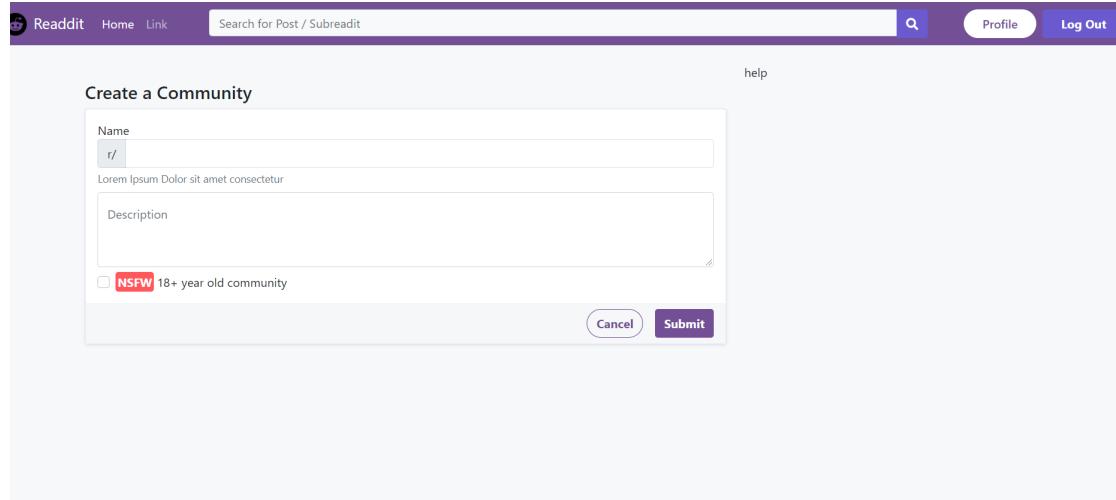
7.2 Login



Users who have signed up can log into their accounts using their created email and password.

7.3 Subreddits

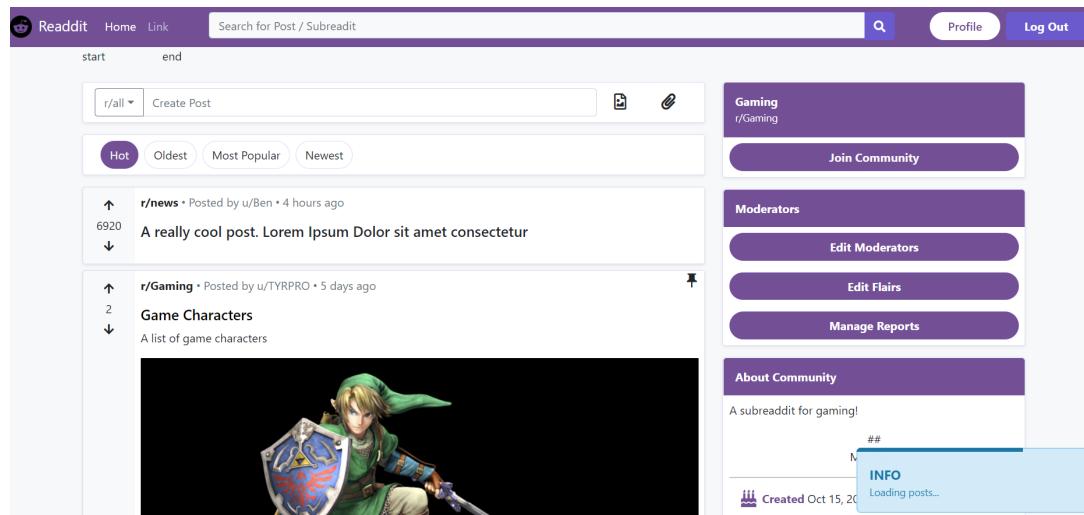
7.3.1 Create Subreddits



The screenshot shows the 'Create a Community' page on Reddit. At the top, there's a purple header bar with the Reddit logo, 'Reddit', 'Home', 'Link', a search bar containing 'Search for Post / Subreddit', and 'Profile' and 'Log Out' buttons. Below the header, the main form has a title 'Create a Community'. It includes fields for 'Name' (with 'r/' entered), 'Description' (containing placeholder text 'Lorem Ipsum Dolor sit amet consectetur'), and a checkbox for 'NSFW 18+ year old community'. At the bottom right of the form are 'Cancel' and 'Submit' buttons.

Users can create subreddits for any category they want. Each subreddit community requires a subreddit name and a subreddit description - describing what the subreddit is about.

7.3.2 View Subreddits



The screenshot shows the 'r/Gaming' subreddit page on Reddit. The top navigation bar is identical to the one in the previous screenshot. The main content area shows the 'Hot' tab selected, displaying a post from 'r/news' by user 'u/Ben' posted 4 hours ago. The post content is 'A really cool post. Lorem Ipsum Dolor sit amet consectetur'. Below this, another post from 'r/Gaming' by user 'u/TYRPRO' is shown, titled 'Game Characters' with the description 'A list of game characters'. To the right of the posts, there's a sidebar with sections for 'Gaming' (including a 'Join Community' button), 'Moderators' (with 'Edit Moderators', 'Edit Flairs', and 'Manage Reports' buttons), and 'About Community' (describing it as a subreddit for gaming). At the bottom right of the sidebar, there's an 'INFO' button with the text 'Loading posts...'.

Users can view created subreddits and all posts associated with that subreddit. There is also a section on the right which shows the subreddit's name, description and when the subreddit was created.

7.3.3 Edit Moderators

The screenshot shows the 'Add a Moderator' section of a subReddit's moderation tools. At the top, there is a search bar labeled 'Search for Post / Subredit'. Below it, two buttons are visible: 'start' and 'end'. A purple header bar contains the text 'Add a Moderator'. Below this, there is a search input field with the placeholder 'Search for username' and a magnifying glass icon. Another purple header bar below it is labeled 'Moderators'. A message 'There are no moderators' is displayed in the main content area.

SubReddit owners can assign other users as Moderators.

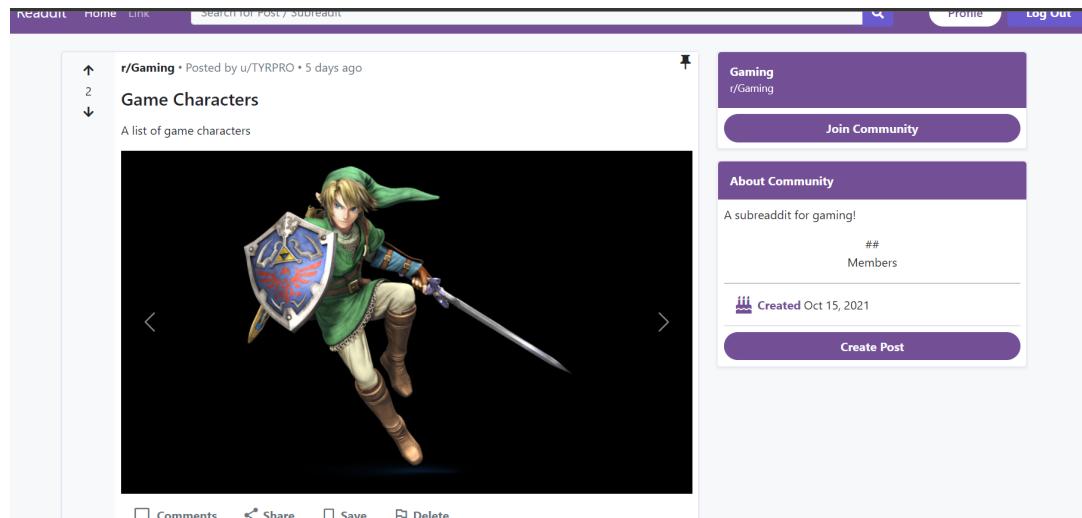
7.3.4 Edit Flairs

The screenshot shows the 'Add a Flair' section of a subReddit's moderation tools. At the top, there is a search bar labeled 'Search for Post / Subredit'. Below it, two buttons are visible: 'start' and 'end'. A purple header bar contains the text 'Add a Flair'. Below this, there are two input fields: one for 'Enter Flair Name' and another for 'Enter Flair Colour in Hex'. To the right of these fields is a blue button labeled 'Add Flair'. Another purple header bar below it is labeled 'Existing Flairs'. A message 'There are no flairs' is displayed in the main content area. In the bottom right corner, there is a light blue info box with the word 'INFO' and the text 'Getting flairs...'.

Moderators can create flairs for users to use in the subReddit.

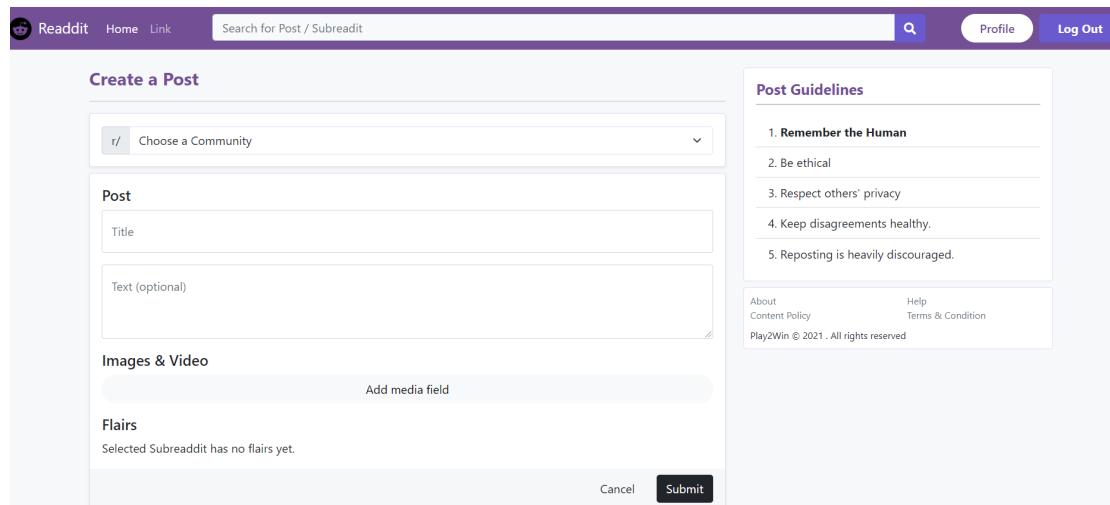
7.4 Posts

7.4.1 View Posts



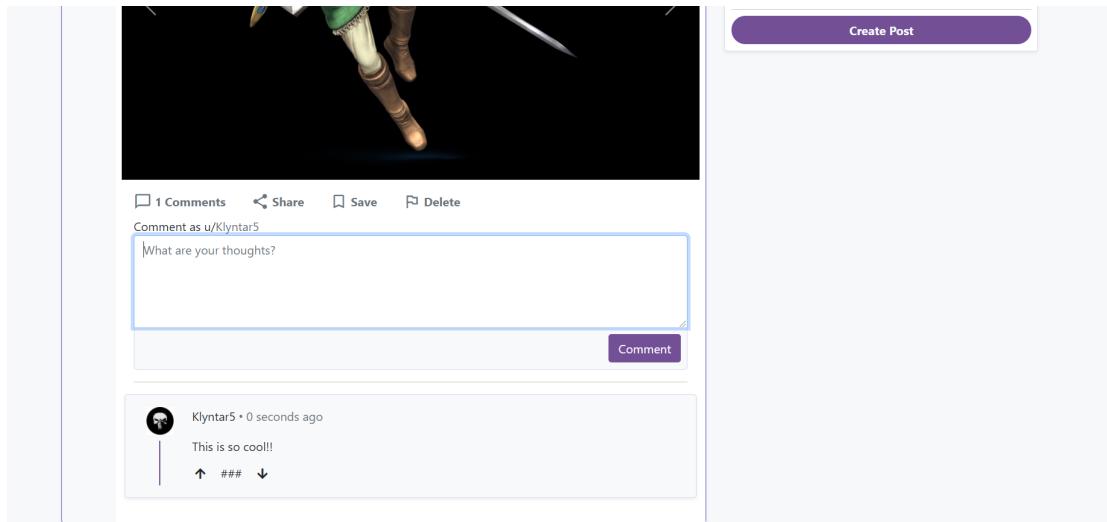
Users can view individual posts. Each post entry displays the post's title. Optionally each post may also display a description and media. The entry will also show the post's author and how long ago the post was created.

7.4.2 Create Posts



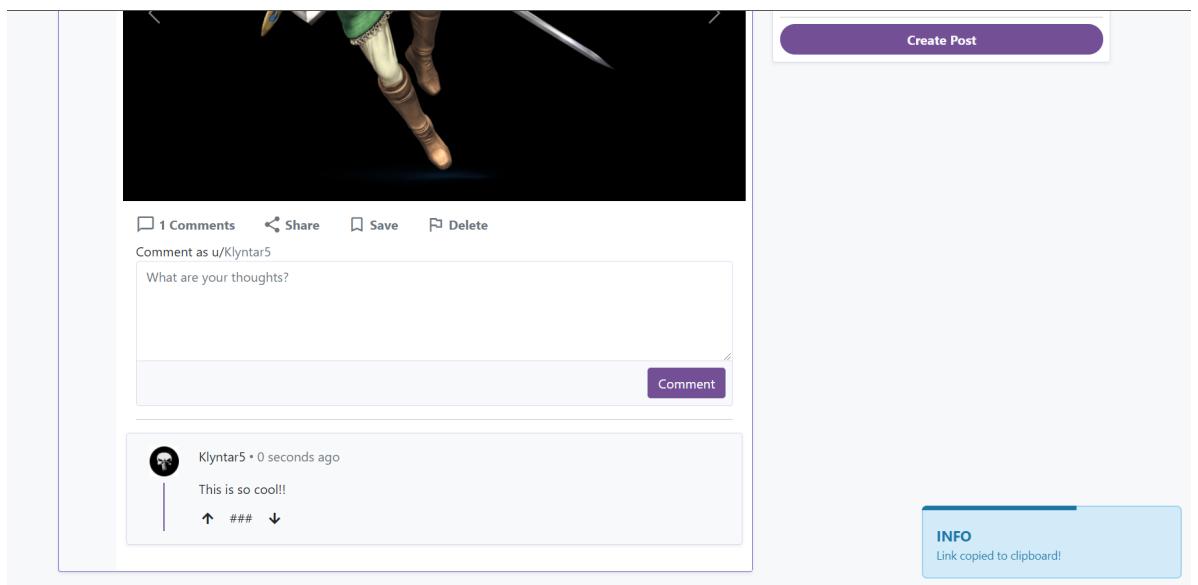
Users are able to create posts on the subreddit of their choice. The post can contain GIFs, Videos, Images or a combination of any of those three. An optional description can also be added, but each post must be titled.

7.4.3 Comment on Posts



Users are able to comment on posts. Comments can be seen by everybody. And each comment entry shows who authored the comment and how long ago the post was made.

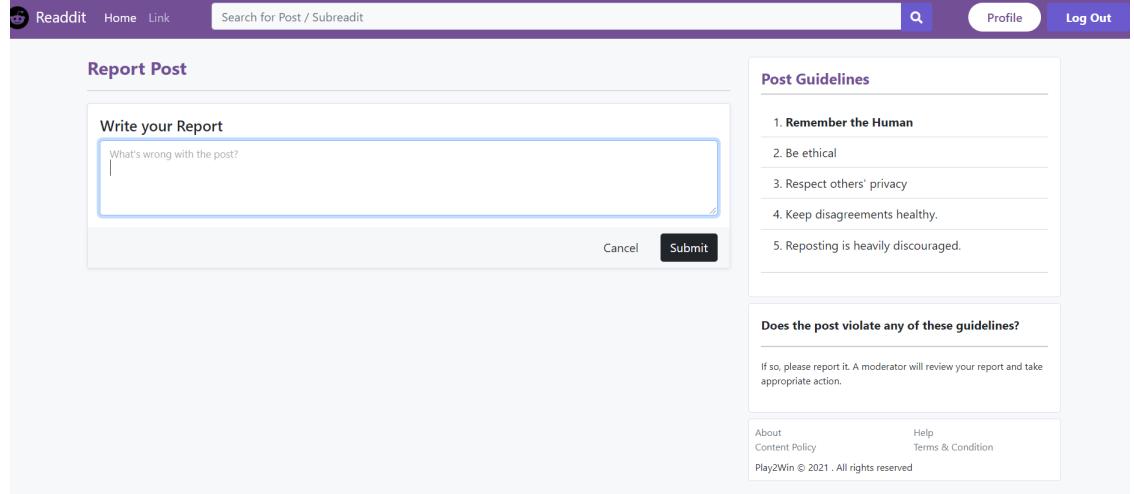
7.4.4 Share Posts



Users can share post by copying the post link when clicking the share button.

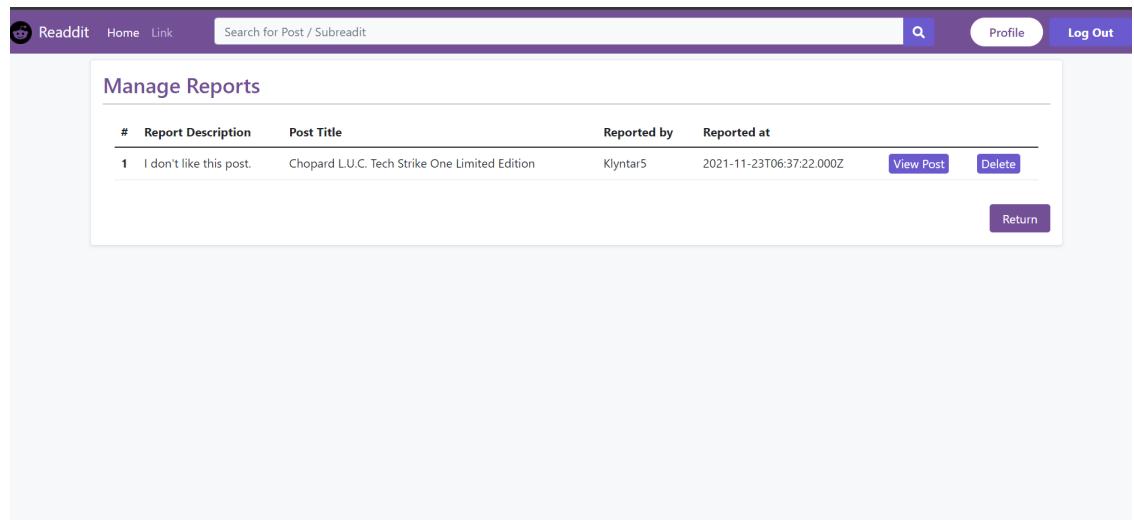
7.4.5 Report/Delete Posts

Report button



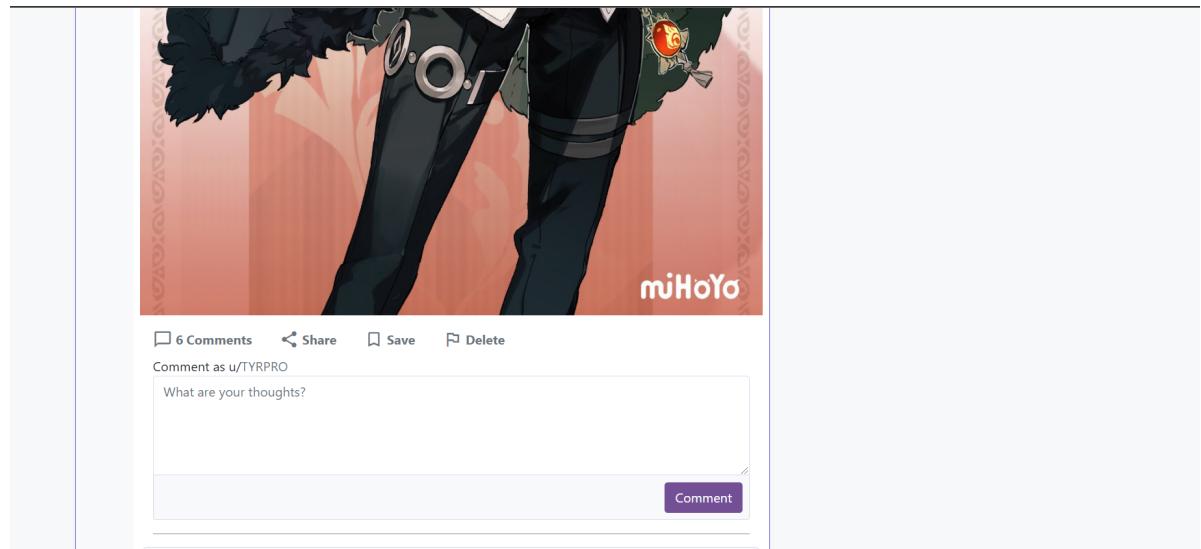
The screenshot shows the 'Report Post' modal window on a Reddit page. At the top left is the Reddit logo and navigation links: 'Reddit', 'Home', and 'Link'. A search bar contains 'Search for Post / Subreddit'. On the right are 'Profile' and 'Log Out' buttons. The main area has a title 'Report Post' and a large text input field labeled 'Write your Report' with the placeholder 'What's wrong with the post?'. Below the input are 'Cancel' and 'Submit' buttons. To the right is a sidebar titled 'Post Guidelines' with five numbered rules: 1. Remember the Human, 2. Be ethical, 3. Respect others' privacy, 4. Keep disagreements healthy, and 5. Reposting is heavily discouraged. At the bottom of the sidebar is a section titled 'Does the post violate any of these guidelines?' with a note: 'If so, please report it. A moderator will review your report and take appropriate action.' At the very bottom of the page are links: 'About', 'Content Policy', 'Help', 'Terms & Condition', and 'Pay2Win © 2021 . All rights reserved'.

Users can report posts that they deem inappropriate.

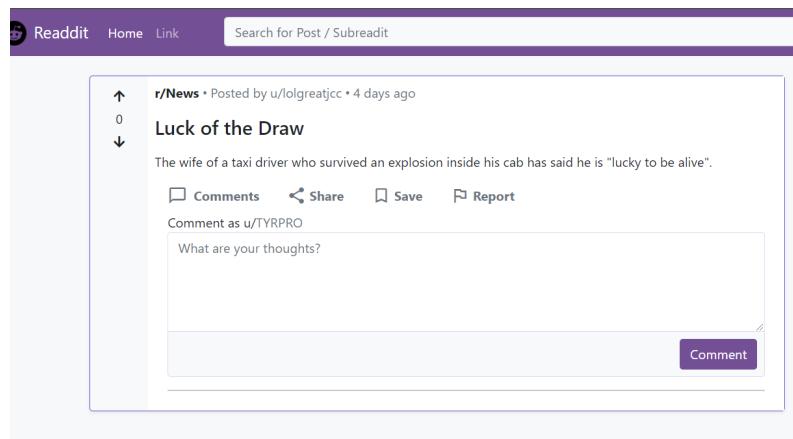


The screenshot shows the 'Manage Reports' page on a Reddit page. The top navigation includes the Reddit logo, 'Home', 'Link', a search bar, 'Profile', and 'Log Out'. The main content area is titled 'Manage Reports' and displays a table of reported posts. The columns are: '#', 'Report Description', 'Post Title', 'Reported by', and 'Reported at'. One row is shown: '# 1 I don't like this post.', 'Post Title: Chopard L.U.C. Tech Strike One Limited Edition', 'Reported by: Klyntar5', 'Reported at: 2021-11-23T06:37:22.000Z', and buttons for 'View Post' and 'Delete'. A 'Return' button is at the bottom right.

Moderators of that subreddit can view reports and either delete the post or delete the report.

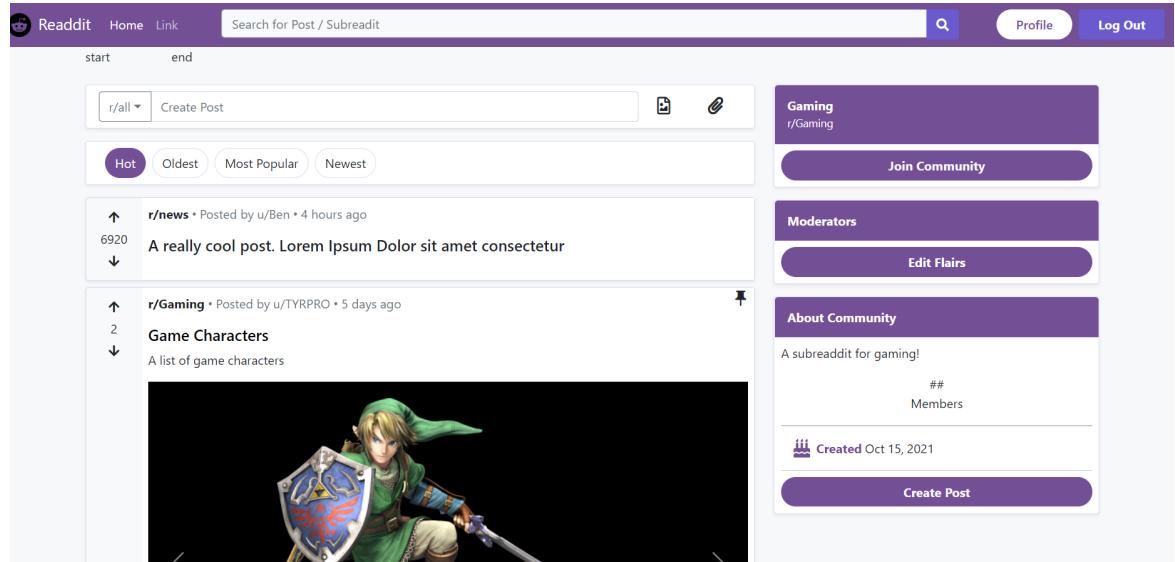


When viewing a post, moderators will be presented with a delete button



While normal users are presented with a report button.

7.4.6 Post Sorting



The screenshot shows the Reddit interface for the r/Gaming subreddit. At the top, there's a search bar and a navigation bar with 'Profile' and 'Log Out'. Below the search bar, there are two buttons: 'start' and 'end'. A dropdown menu shows 'r/all' and 'Create Post'. Below these are sorting options: 'Hot' (which is selected), 'Oldest', 'Most Popular', and 'Newest'. The main content area displays two posts:

- A post from r/news by u/Ben, posted 4 hours ago, with 6920 upvotes. The title is "A really cool post. Lorem Ipsum Dolor sit amet consectetur".
- A post from r/Gaming by u/TYRPRO, posted 5 days ago, with 2 upvotes. The title is "Game Characters", and the description is "A list of game characters". Below the title is a large image of Link from The Legend of Zelda.

On the right side of the screen, there's a sidebar for the r/Gaming community. It includes sections for 'Moderators' (with a 'Edit Flairs' button), 'About Community' (described as "A subreddit for gaming!", with member statistics like "# Members"), and a 'Create Post' button.

Subreddits posts can be ordered by:

1. Hot. Sorts posts according to our sorting algorithm which takes into account the time since submission and votes.
2. Oldest. Sorts post by oldest posts first
3. Most Popular. Sorts post by number of votes
4. Newest. Sorts post by newest posts first

7.4.7 Save Posts



The screenshot shows a user's profile page with a saved post from r/Gaming. The post is titled "Thinking of buying a PS5" and was posted by u/Klyntar5 4 days ago. The post content is "Should I? I already have a pretty good PC and a PS4, but I heard that the PS5 is amazing". Below the post are standard Reddit controls: 'Comments', 'Share', 'Save' (which is highlighted in purple), and 'Delete'. Below this, another post is shown: "Diluc discussion" by u/TYRPRO 7 days ago, with the comment "Is diluc an amazing 5*??". This post also has standard Reddit controls.

Users can save posts they like to view it again later in the profile page.

7.5 Search

7.5.1 Basic Search

The screenshot shows the Readdit search interface. At the top, there is a purple header bar with the Readdit logo, 'Readdit', 'Home', 'Link', a search bar containing 'Search for Post / Subreddit', a magnifying glass icon, 'Profile', and 'Log Out'. Below the header, there are two tabs: 'Subreddits' and 'Posts', with 'Posts' being the active tab. A message 'Displaying Search Results for diluc' is shown above a list of search results. The results are presented in two sections: 'Similar results' and 'More results'. The first section contains one item: 'r/Gaming • Posted by u/TYRPRO • 7 days ago Diluc discussion'. The second section contains one item: 'r/News • Posted by u/lolgreatjcc • 4 days ago Billionaire Playboy Diluc Whitman buys new Mansion'. At the bottom right, there is a light blue box labeled 'INFO' with the text 'Searching for subreddits and posts...'. The background of the main content area has a subtle gradient.

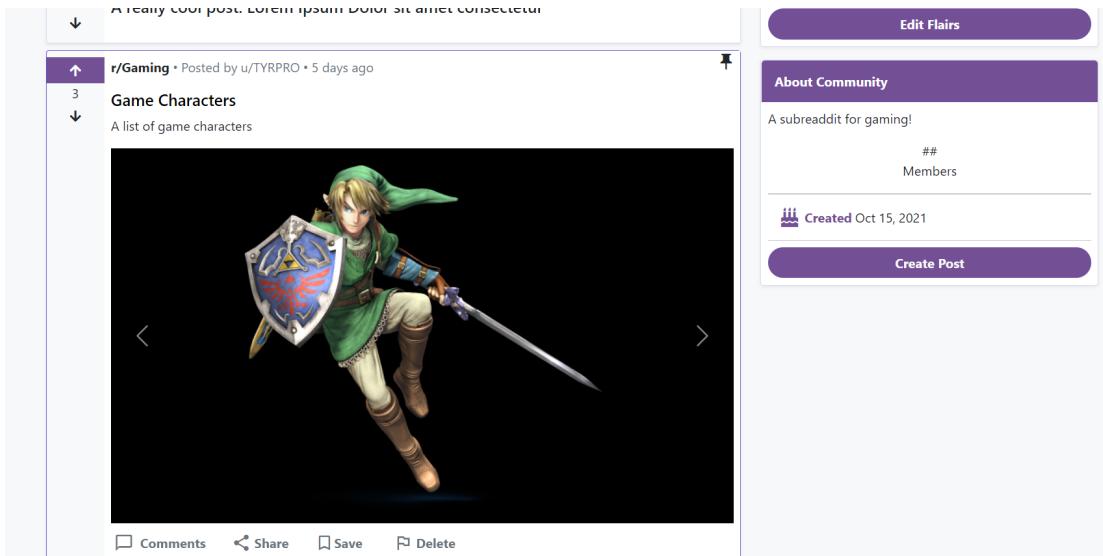
Users can search for posts or subreddits by entering a keyword.

7.5.2 Similar Search

The screenshot shows the Readdit search interface, similar to the previous one but with different results. The search term 'diluc' is entered in the search bar. The results are displayed under the heading 'Displaying Search Results for diluc'. A message 'No results found' is shown. Below this, there is a section titled 'Similar results' which lists three items: 'r/Gaming • Posted by u/TYRPRO • 7 days ago Diluc discussion', 'r/News • Posted by u/lolgreatjcc • 4 days ago Billionaire Playboy Diluc Whitman buys new Mansion', and 'r/News • Posted by u/lolgreatjcc • 4 days ago Luck of the Draw'. At the bottom right, there is a light blue box labeled 'INFO' with the text 'Searching for subreddits and posts...'. The background of the main content area has a subtle gradient.

Similar Search displays results similar to the keyword that the basic search does not display. This is done using our own function that calculates the similarity between two strings.

7.6 Rating System



Reddit's Rating system consists of upvotes and downvotes. The total number of votes for a post is the number of downvotes subtracted from the number of upvotes. As said previously this is used for our sorting algorithm.

7.7 Profile

7.7.1 View User Information

A screenshot of a user profile page on Reddit. The profile belongs to u/TYRPRO. It shows a blue profile picture with a white character icon. The user has posted 11 items, including 5 posts and 6 comments. The posts are: "Diluc discussion" (r/Gaming), "Ayaka Showcase (No buffs!)" (r/Gaming), "My top two anime scenes" (r/Anime), "Castlevania anyone?" (r/Anime), and "Discord user TYRPRO is trending worldwide!" (r/Gaming). On the right side of the profile page, there are buttons for "Edit Account", "New Post", and "Admin Console". At the bottom, there are links for "About", "Content Policy", "Help", "Terms & Condition", and "Play2Win © 2021 . All rights reserved".

Users can view information about their account in their User Profile.
(i.e. Username, Profile Picture and Account Creation Date)

7.7.2 View User Posts

The screenshot shows the Reddit user profile for 'u/TYRPRO'. The main content area displays five posts from subreddits r/Gaming, r/Anime, and r/Play2Win. The posts are as follows:

- 1 Diluc discussion (r/Gaming)
- 3 Ayaka Showcase (No buffs!) (r/Gaming)
- 3 My top two anime scenes (r/Anime)
- 1 Castlevania anyone? (r/Anime)
- 3 Discord user TYRPRO is trending worldwide! (r/Gaming)

The right sidebar contains the user's profile picture, name (u/TYRPRO), a 'Edit Account' button, a 'New Post' button, and an 'Admin Console' button. It also includes links for 'About', 'Content Policy', 'Help', 'Terms & Condition', and the copyright notice 'Play2Win © 2021, All rights reserved'.

Users can also view posts they've created in User Profile.

7.7.3 View User Comments

The screenshot shows the Reddit user profile for 'u/TYRPRO'. The main content area displays five comments made by the user on various posts. The comments are as follows:

- "I think he's not bad" (commented on Diluc discussion)
- "But I also think that he is getting outclassed" (commented on Diluc discussion)
- "I also really need to make more comments about him" (commented on Diluc discussion)
- "I want to test my app" (commented on Diluc discussion)
- "Please don't delete thank you" (commented on Diluc discussion)
- ... (commented on Ayaka Showcase (No buffs!))
- "She's so cool!" (commented on Ayaka Showcase (No buffs!))
- ... (commented on My top two anime scenes)

The right sidebar contains the user's profile picture, name (u/TYRPRO), a 'Edit Account' button, a 'New Post' button, and an 'Admin Console' button. It also includes links for 'About', 'Content Policy', 'Help', 'Terms & Condition', and the copyright notice 'Play2Win © 2021, All rights reserved'.

Users can view comments they've made on posts in User Profile

7.7.4 View User Saved Posts

Reddit Home Link Search for Post / Subredit Profile Log Out

Posts Comments Saved

↑ r/News • Posted by u/Klyntar5 • 2 days ago
0 This ain't for the best.
↓

Comments Share Unsave Report

u/TYRPRO Edit Account

Cake Day 1 November 2021

New Post Admin Console

About Content Policy Help Terms & Condition Play2Win © 2021. All rights reserved

Users can also view posts they've saved in User Profile

7.7.5 Edit Account

Edit Account

Username: TYRPRO

Email: tyrpro@gmail.com

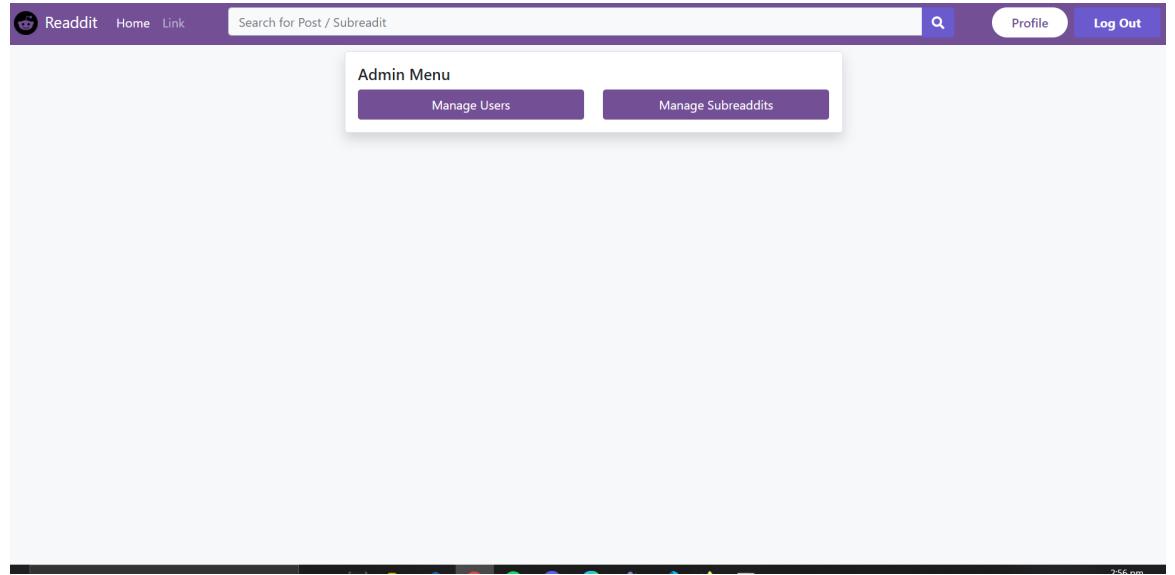
Select New Image for Profile Picture (Optional):

Choose File No file chosen

2 Factor Authentication

Users can also edit their account information from the User Profile.

7.7.6 Access Admin Console (Admin Only)



If the User is an Admin, they will be able to access the Admin Console via their User Profile.

7.8 Admin Console

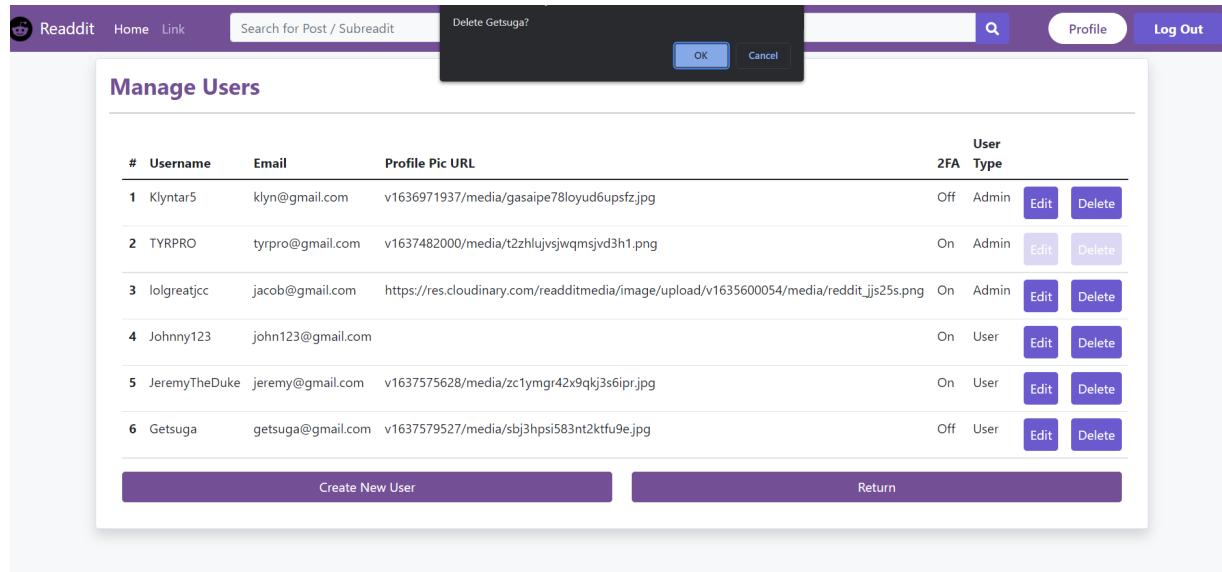
7.8.1 View all Users

| # | Username | Email | Profile Pic URL | 2FA | User Type | |
|---|---------------|-------------------|--|-----|-----------|---|
| 1 | Klyntar5 | klyn@gmail.com | v1636971937/media/gasaipe78loyud6upsfz.jpg | Off | Admin | <button>Edit</button> <button>Delete</button> |
| 2 | TYRPRO | tyrpro@gmail.com | v1637482000/media/t2zhluvjwqmsjvd3h1.png | On | Admin | <button>Edit</button> <button>Delete</button> |
| 3 | lolgreatjcc | jacob@gmail.com | https://res.cloudinary.com/readditmedia/image/upload/v1635600054/media/reddit_jjs25s.png | On | Admin | <button>Edit</button> <button>Delete</button> |
| 4 | Johnny123 | john123@gmail.com | | On | User | <button>Edit</button> <button>Delete</button> |
| 5 | JeremyTheDuke | jeremy@gmail.com | v1637575628/media/zc1ymgr42x9qkj3s6ipr.jpg | On | User | <button>Edit</button> <button>Delete</button> |
| 6 | Getsuga | getsuga@gmail.com | v1637579527/media/sbj3hpsi583nt2ktfu9e.jpg | Off | User | <button>Edit</button> <button>Delete</button> |

[Create New User](#) [Return](#)

Admins are able to view all users from the database.

7.8.2 Delete User

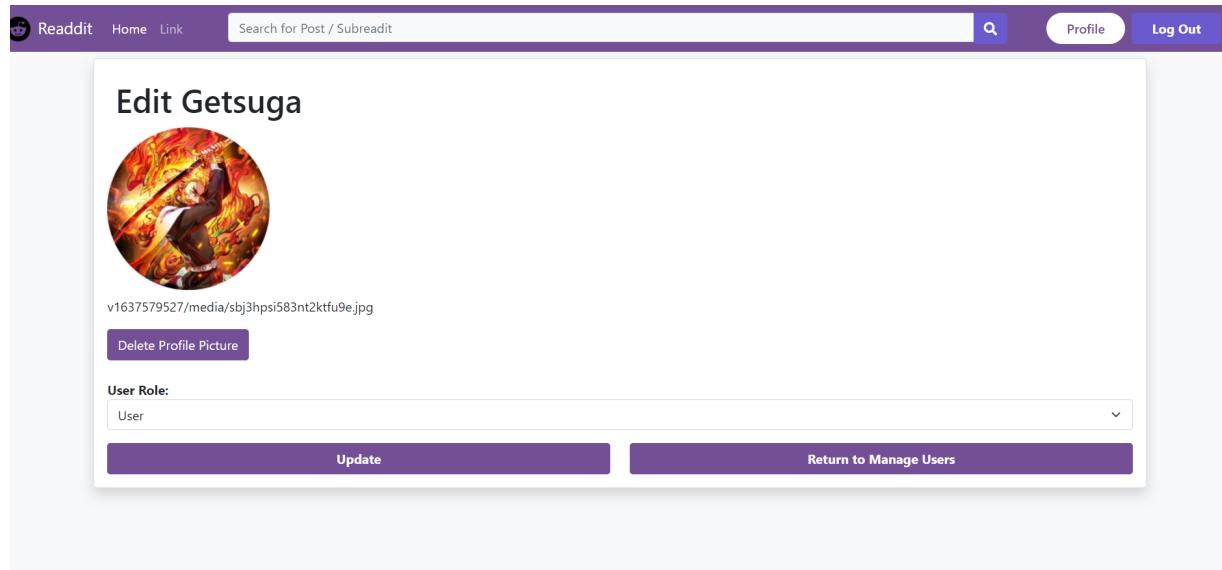


The screenshot shows the Reddit Admin interface. A modal dialog box is centered over the page, asking "Delete Getsuga?". Below the modal, the "Manage Users" table is visible, showing a list of users with columns for #, Username, Email, Profile Pic URL, 2FA, and User Type. The user "Getsuga" is listed at the bottom of the table. At the bottom of the page are two buttons: "Create New User" and "Return".

| # | Username | Email | Profile Pic URL | 2FA | User Type | | |
|---|---------------|-------------------|---|-----|-----------|----------------------|------------------------|
| 1 | Klyntar5 | klyn@gmail.com | v1636971937/media/gasaipe78loyud6upsfz.jpg | Off | Admin | Edit | Delete |
| 2 | TYRPRO | tyrpro@gmail.com | v1637482000/media/t2zhlujsjwqmsjvd3h1.png | On | Admin | Edit | Delete |
| 3 | lolgreatjcc | jacob@gmail.com | https://res.cloudinary.com/redditmedia/image/upload/v1635600054/media/reddit_jjs25s.png | On | Admin | Edit | Delete |
| 4 | Johnny123 | john123@gmail.com | | On | User | Edit | Delete |
| 5 | JeremyTheDuke | jeremy@gmail.com | v1637575628/media/zc1ymgr42x9qkj3s6ipr.jpg | On | User | Edit | Delete |
| 6 | Getsuga | getsuga@gmail.com | v1637579527/media/sbj3hpsi583nt2ktfu9e.jpg | Off | User | Edit | Delete |

Admins are able to delete users.

7.8.3 Modify User



The screenshot shows the Reddit Admin interface for editing a user profile. The title "Edit Getsuga" is displayed above a circular profile picture of a character in a dynamic pose. Below the picture is the URL "v1637579527/media/sbj3hpsi583nt2ktfu9e.jpg". A button labeled "Delete Profile Picture" is present. A dropdown menu for "User Role" is open, showing the option "User". At the bottom of the page are two buttons: "Update" and "Return to Manage Users".

Admins are able to delete user profile pictures or change the role of a user

The screenshot shows the 'Edit testicle' page. At the top, there's a placeholder profile picture for the user 'testicle'. Below it is the URL: https://res.cloudinary.com/redditmedia/image/upload/v1635600054/media/reddit_jjs25s.png. There's a 'Delete Profile Picture' button. A dropdown menu for 'User Role' is open, showing 'User' as the selected option. At the bottom are two buttons: 'Update' and 'Return to Manage Users'.

Users with deleted profile pictures will have the default profile picture.

7.8.4 Add User

The screenshot shows the 'Create New User' page. It has fields for 'Username' (with placeholder 'Enter username'), 'Email' (with placeholder 'Enter email'), and 'Select Image for Profile Picture (Optional)' (with a 'Choose File' button showing 'No file chosen'). Below these is a 'Password' field (placeholder 'Enter password') and a checkbox for '2 Factor Authentication'. At the bottom are two buttons: 'Create' and 'Return to Manage Users'.

Admins are able to manually create a new user.

7.8.5 View Subreddits

The screenshot shows a web application interface for managing subreddits. At the top, there is a navigation bar with a logo, 'Readdit', 'Home', 'Link', a search bar 'Search for Post / Subreddit', and user profile links 'Profile' and 'Log Out'. Below the navigation bar is a section titled 'Manage Subreddits'. This section contains a table with the following data:

| # | Name | Description | Owner | Created At | Edit | Delete |
|---|--------|--------------------------|-------------|--------------------------|-----------------------|-------------------------|
| 1 | Gaming | A subreddit for gaming! | Klyntar5 | 2021-11-15T10:33:56.000Z | <button>Edit</button> | <button>Delete</button> |
| 2 | News | A subreddit for News! | lolgreatjcc | 2021-11-15T10:33:56.000Z | <button>Edit</button> | <button>Delete</button> |
| 3 | Tech | A subreddit for Tech! | TYRPRO | 2021-11-15T10:33:56.000Z | <button>Edit</button> | <button>Delete</button> |
| 4 | Anime | A subreddit for Anime!! | Klyntar5 | 2021-11-18T14:23:05.000Z | <button>Edit</button> | <button>Delete</button> |
| 5 | Movies | A subreddit for movies!! | Klyntar5 | 2021-11-22T14:09:59.000Z | <button>Edit</button> | <button>Delete</button> |

At the bottom right of the 'Manage Subreddits' section is a purple 'Return' button.

Admins are able to view all Subreddits.

7.8.6 Modify Subreddit

The screenshot shows a web application interface for editing a subreddit. At the top, there is a navigation bar with a logo, 'Readdit', 'Home', 'Link', a search bar 'Search for Post / Subreddit', and user profile links 'Profile' and 'Log Out'. Below the navigation bar is a section titled 'Edit Subreddit'. This section contains a form with the following fields:

Name:

Description:

At the bottom of the form are two buttons: 'Update' and 'Return to Manage Subreddits'.

Admins are able to modify subreddit details

7.8.7 Delete Subreddit

The screenshot shows a web application interface for managing subreddits. At the top, there's a navigation bar with links for 'Home' and 'Link', a search bar, and user profile options like 'Profile' and 'Log Out'. Below the navigation is a table titled 'Manage Subreddits' with columns for '#', 'Name', 'Description', 'Owner', and 'Created At'. The table lists five subreddits: 'Gaming', 'News', 'Tech', 'Anime', and 'Movies'. Each row has 'Edit' and 'Delete' buttons. A modal dialog box is open over the table, asking 'Delete Gaming?'. It has 'OK' and 'Cancel' buttons. In the bottom right corner of the main content area, there's a 'Return' button.

| # | Name | Description | Owner | Created At | | |
|---|--------|--------------------------|-------------|--------------------------|-----------------------|-------------------------|
| 1 | Gaming | A subreddit for gaming! | Klyntar5 | 2021-11-15T10:33:56.000Z | <button>Edit</button> | <button>Delete</button> |
| 2 | News | A subreddit for News! | lolgreatjcc | 2021-11-15T10:33:56.000Z | <button>Edit</button> | <button>Delete</button> |
| 3 | Tech | A subreddit for Tech! | TYRPRO | 2021-11-15T10:33:56.000Z | <button>Edit</button> | <button>Delete</button> |
| 4 | Anime | A subreddit for Anime!! | Klyntar5 | 2021-11-18T14:23:05.000Z | <button>Edit</button> | <button>Delete</button> |
| 5 | Movies | A subreddit for movies!! | Klyntar5 | 2021-11-22T14:09:59.000Z | <button>Edit</button> | <button>Delete</button> |

Admins can delete subreddits.

7.9 Mobile Application

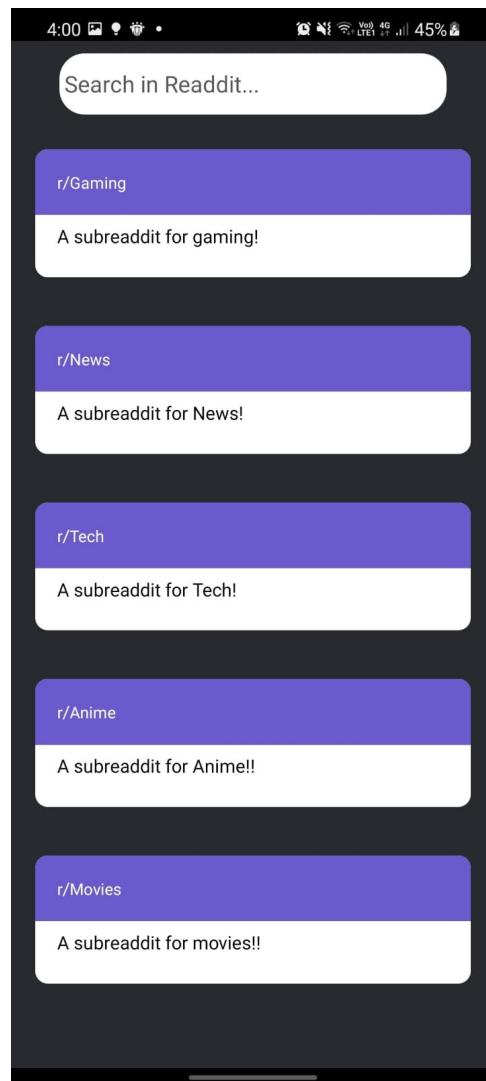
7.9.1 Homepage

When users open the application, they are greeted with the homepage. They can scroll through the same posts that are displayed on the web application's homepage:



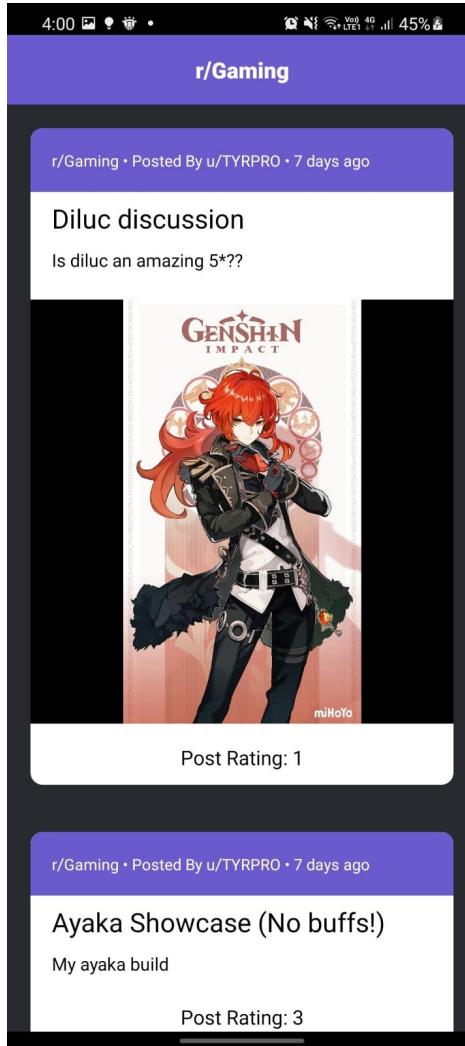
7.9.2 Search Function (Subreddits)

Users can also search for Subreddits in the search bar to return the Subreddits that are similar to the Subreddit names. Entering a blank search will return all Subreddits:



7.9.3 Subreddits

Clicking on a Subreddit shows the posts within the Subreddit. The user can tell which Subreddit he/she is in by the header:



The posts shown in the Subreddit and homepage will show the post title, post content, Subreddit, posted by which user, how long ago the post was posted, the post rating, as well as any media that the post has. The post only has media support for images and GIFs at the moment.

7.9.4 Post Information

Clicking on the post shows the same content but also shows the comments:

