



CS 30700
Design Document

Team 17

Ethan Ling

Leo Gu

Matthew Sigit

Nicholas Song

Table of Contents

1. Purpose.....	3
1.1 Functional Requirements.....	3
1.2 Non Functional Requirements.....	7
2. Design Outline.....	8
2.1 Client-Server Model.....	8
2.2 High-Level System Overview.....	8
2.3 Detailed Component Overview.....	9
AWS.....	9
VPC/Virtual Networks.....	9
Route 53.....	9
Load Balancers.....	10
Autoscalers.....	10
Frontend EC2.....	10
Backend EC2.....	10
Relational Database.....	10
S3 Bucket.....	11
Caching.....	11
Machine Learning App.....	11
3. Design Issues.....	12
Functional Issues:.....	12
Nonfunctional Issues:.....	13
4. Design Details.....	16
4.1 Class Diagram and Descriptions.....	16
4.2 Description of Classes and their Interactions.....	16
4.3 Sequence Diagrams.....	19
4.4 Navigation Flow Diagrams.....	22
4.5 UI Mockups.....	23

1. Purpose

Facial recognition is a popular and effective tool employed by applications and search engines to fill many needs. In fact, there exist several readily available and free-use image recognition softwares such as Google Lens, Bing Visual Search and Camfind. Some of these softwares are well integrated into search engines and are good at general purpose recognition for a host of items. However, while these image recognition softwares excel at recognizing images in real life, their accuracy drops significantly when prompted with animated characters. This is because they are trained on real, much more noisy images, meaning their accuracy on animated characters is not as high.

Our solution to this problem is Annix, a tool specifically geared towards anime watchers. With the popularity of anime increasing, there subsequently follows the need for anime recognition software. Many anime watchers may find a clip of an anime online, but don't know what the name of the characters or anime are. This makes it much harder for them to find the anime online and watch on their own. That's where our tool comes in. Annix is a multi platform service that aims to use Machine Learning and Convolutional Neural Networks trained specifically on animated characters to fulfill this need.

1.1 Functional Requirements

1. User Account

As a user,

- a. I would like to register an account on Annix.
- b. I would like to login and logout from my account on Annix.
- c. I would like to link and unlink my account to an email address.
- d. I would like to change critical information about my account like my password, username, email, and profile picture.
- e. I would like to reset my password if I have forgotten it with security questions and an email sent to my inbox.

- f. I would like to write a description for myself and have it displayed on my profile.
- g. I would like to change my profile picture to a recent search.
- h. (If time allows) I would like to have the security to login using a 2-factor authentication service.
- i. I would like to be able to delete my account and all of the information Annix has stored in it.

2. Analytics

As a user,

- a. I would like to have a method of exporting a backlog or history of previous images uploaded and their corresponding characters.
- b. I would like to be able to know where to watch the show that a character I've searched appears in and have the links to the services where the show is hosted.
- c. I would like to be able to tell the model if it has misidentified a character, as well as send the correct character if I know it.
- d. I would like to be able to easily disable search history and other data collecting functions on Annix.
- e. I would like to know generalized data about the user base such as most commonly recognized characters, shows, and hosting services.
- f. I would like analytics specific to me like characters frequently searched.
- g. I would like to know statistics about my usage, such as characters searched, images recognized, people helped, etc.
- h. I would like to know what personal data Annix has stored and how it is being used.
- i. I would like to identify different characters by image with an accuracy of at least 80%.

3. User Interface

As a User,

- a. I would like to be able to seamlessly go between profile, service and data pages as well as other function pages.
- b. I would like for the app to automatically switch from light and dark modes depending on my system settings, and I would like to be able to switch it back if I choose to do so.
- c. I would like to seamlessly upload an image and obtain a response according to the image that is accurate.
- d. I would like to be able to access a guide to use this application.
- e. (If time allows) I would like to see the functionality of each button when hovering the mouse over it.
- f. I would like to access the application over a mobile device
- g. I would like to be able to see Frequently Asked Questions for any features I don't understand.
- h. I would like to be able to see a Contact Us page to contact the developers if I find something wrong.

4. Help System

As a User,

- a. If the search function is unable to find my character, I would like to be able to ask other users if they recognize a character.
- b. I would like to be able to help other users if they request the details of a character that I recognize.
- c. I would like to receive some form of rating or ranking system dependent on how many users I have accurately helped.
- d. I would like to be able to see top ranked helpers based on time increments.
- e. I would like to receive some form of rating or ranking system dependent on how many users I have accurately helped.
- f. I would like to be able to apply to be a moderator when I reach a certain rating.
- g. I would like to be able to see top ranked helpers based on time increments.
- h. As a moderator, I would like my moderator status to show on my profile.

- i. As a moderator, I would like to be able to verify any responses in the help section.
- j. I would like to be able to search for other users and see details relating to ranking, account status and more.
- k. I would like to be able to filter my search depending on use time, ranking, or date of profile creation.

5. General Features

As a User,

- a. (If time allows) I would like to be able to help improve the model by labeling data if I feel the service is lacking.
- b. I would like a button on the application to review the application if I am satisfied with its performance.
- c. I would like to be able to request new features.
- d. I would like a button on the application to send constructive criticism to the developers if I would like a certain aspect of the application to be improved.
- e. (If time allows) I would like to not only identify characters from their source material but also fan-made versions of them as well.
- f. I would like to be able to recognize characters, even if they are slightly edited with filters or Photoshop.
- g. (If time allows) I would like for the application to tell me if my image is incompatible
- h. I would like to know if the application is unsure of my requested character.
- i. I would like to add example images and descriptions of characters that I would like added.
- j. I would like to be able to use the product without signing in.
- k. I would like to know the source material of the image I uploaded displayed in the model.
- l. I would like to be able to easily replace an image if I realize I uploaded the wrong one.
- m. I would like to be able to search for characters and see details about the characters.

1.2 Non Functional Requirements

As a developer:

1. I would like to be able to run robustness tests and see the results of such tests.
2. I would like to train or modify the model in a timely manner that does not interfere with other development time.
3. I would like the changes I made to the version in source control to be reflected on our system.
4. I would like the application to run on multiple platforms

As the frontend:

1. I would like to correctly display information passed to me from the backend, regardless of the platform the user is using.
2. I would like to have a color theme that is appealing to the eye of the user.
3. I would like to be able to switch between lighter and darker modes.
4. I would like to make the transitions between pages smooth and nice to look at.

As the backend:

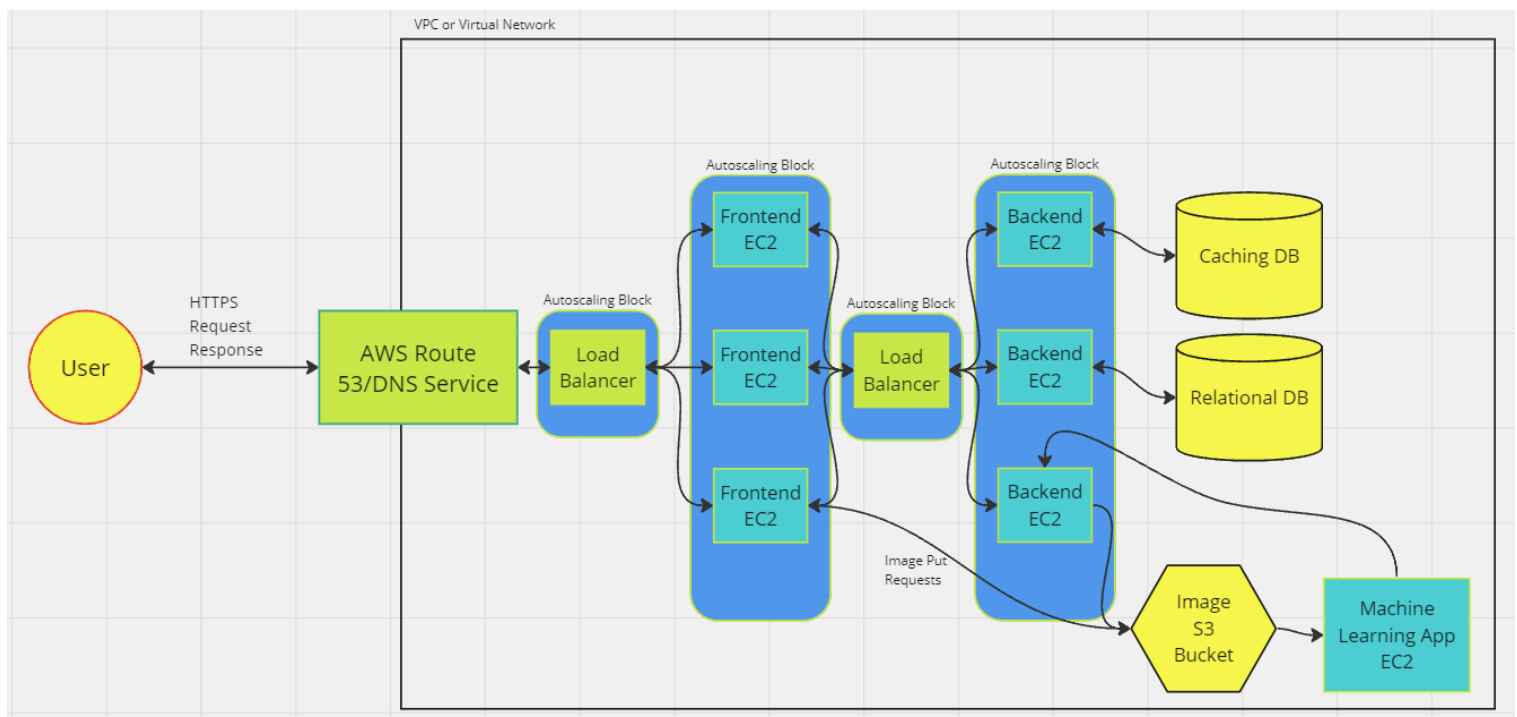
1. I would like to seamlessly connect with the database to access previous searches.
2. I would like to be able to receive flags from the frontend that describe the image submitted.
3. I would like to be able to get results from the Machine Learning model as a response for each user prompt.
4. I would like to send the results from the Machine Learning model to the front end for display.
5. I would like to be able to route user information to the frontend

2. Design Outline

2.1 Client-Server Model

Our system requires a bit more intricacies than a standard web application due to the inclusion of a Machine Learning Model. However, we still follow the same standard client-server model that many web applications still use today. The client will send requests to our server to login, request results, upload images and more. The server will handle how the client's information is handled, routing image uploads to the S3 Bucket, user information to the Relational Database, and use requests to the Machine Learning App. The Database will store relevant user information and also images references.

2.2 High-Level System Overview



Our system consists of a modified 3 tier AWS system. We route our users HTTPS requests through AWS Route 53 which then sends it to the load balancer to find an available EC2 that contains our frontend web application. This application is going to be written in ReactJS and route to another layer

of EC2s, this time containing our backend, through a load balancer layer. The backend written in Flask will then handle http requests that contain JSON data that will tell it the command. It can interact with reads and writes on the Caching and Relational Database, as well as route PUT requests into our S3 Bucket and make calls to our Machine Learning Application on another EC2. The S3 Bucket in this case will only contain images. Our Machine Learning application will likely be written in Tensorflow and Keras, but can also be implemented in Pytorch. It will contain a wrapper that handles input images and requests, inputs them into our model, and sends the results in JSON form back to our users.

2.3 Detailed Component Overview

AWS

AWS contains many great features for hosting our web application and also is cheap because they have a student credit feature which nullifies most of the cost.

VPC/Virtual Networks

VPC's provide a necessary layer of security in our system because internal IP addresses are not exposed to outside users. This allows us to use non-encrypted REST methods for data transfer and provides us mechanisms to prevent attacks like SQL injection and more.

Route 53

Route 53 is AWS's DNS service which allows users to access our frontend EC2s in our Virtual Network without exposing DNS data to the internet. Users would be able to enter our domain name into the search bar and it would resolve to our Web Application.

Load Balancers

Load Balancers allow us to evenly split requests between multiple EC2s so that no EC2 becomes overloaded with requests. It is a key part in handling routing in our system.

Autoscalers

Auto Scaling units allow us to only pay for what we need in this situation and allow our system to be dynamic and handle more and more users if the need ever arises.

Frontend EC2

EC2s in AWS are essentially virtual machines which we can use to run our applications. Our frontend EC2 will contain our frontend application written in ReactJS. This will contain multiple components and modals which users can interact with to go to different pages, post, put, and get information, as well as use our model. These EC2s will route into a load balancer which connects to our backend layer.

Backend EC2

Our backend will be hosted on the Backend EC2s. The backend will be written in Flask and consist of routes that will handle http requests to read, write, update, and delete to the database. It will also handle requests of S3 bucket access and transferring images to the machine learning application.

Relational Database

Our RDBMS will contain relevant user information in multiple tables. For example, there will be a user table which contains username, password, email, and reference to profile image. This information will be easily exported through the JSON format which Python and JS can parse very nicely.

S3 Bucket

The S3 Bucket will contain images that users upload. In it will be profile images and images that will be used for classification in our machine learning app. These can easily be passed to our model through reference.

Caching

Caching is an important tool to stop bottlenecking and improve response times. We will probably have a caching database like Redis which will cache recently called information for faster retrieval.

Machine Learning App

Our machine learning application will be hosted on an EC2 and contain a wrapper that parses in input images and converts the results into JSON to be sent back to the user. The model will be written in either Tensorflow and Keras or Pytorch transformers have a goal accuracy of 80 percent. The response will be in the form of a top 5 format with confidence accuracies displayed with the resulting character.

3. Design Issues

Functional Issues:

1. What does the user need to log in?

- a. Username, Password, Email
- b. Single Sign-On (SSO)
- c. No login required

Choice: Both A and C

Justification: We wanted users to be able to use our application quickly without saving any of their data so we didn't want to force every user to make an account. However, we still wanted users to be able to access their history and analytics of the site specific to them so we wanted to give users the option of creating an account. We decided against SSO as it would add unnecessary complexity to our relational database. However, if time allows, we could still add SSO.

2. How should the application present the characters returned from the model?

- a. Show the top result
- b. Show the top 5 results
- c. Give a list of all the characters and accuracies

Choice: Show the top 5 results

Justification: We decided that showing the top 5 choices would be the best because in low confidence situations, the model's first choice might not be the correct choice. However, with 5 top choices, the likelihood of the correct character appearing would be greatly increased. We didn't want to show all of the characters and their accuracies because we thought it would be too much information and would not help the user find their character.

3. What happens if the model gets a choice wrong?

- a. Do nothing
- b. Wait for a developer update to the model
- c. Allow the user to enter the correct character

Choice: Allow the user to enter the correct character

Justification: With the release of new characters and varying art forms, it is not always possible for our model to accurately identify every single character. Therefore, if the model returns an incorrect character, we can allow users to enter in the correct name, which would allow our model to continuously improve. With this, users could earn points and know how much they are helping other users.

Nonfunctional Issues:

1. What frontend language/framework should we use?

- a. Angular
- b. React
- c. HTML + Javascript (Native)
- d. Next.js

Choice: React5342563433434534563

Justification: We decided to use React as it is very easy to learn and comes with many tutorials for our members who had never used it before. React also has Virtual DOM, which makes rendering our application much more efficient. React also handles all the updates to the DOM, making it much more lightweight compared to others.

2. What backend language/framework should we use?

- a. Flask (Python)
- b. Django (Python)
- c. Sprint Boot (Java)
- d. Node.js (JavaScript)
- e. PHP

Choice: Flask (Python)

Justification: We decided to use Flask as it was what most of our members had experience in Python and for its simplicity and ease of use. Its ease of use will allow us to create RESTful APIs to interact with the rest of our application. It is also very flexible with components for database integration, authentication, and APIs. Many of the other options are very bloated and not as beginner friendly.

3. What web host service should we use?

- a. AWS
- b. Azure
- c. Google Cloud
- d. Heroku

Choice: AWS

Justification: We Chose AWS for its ease of use and the fact that it is widely used across the industry. We wanted everyone on the team to get hands-on experience using it. With AWS, we could use autoscalers to easily scale our backend with the number of users. The main downside of AWS is the cost, but could be easily nullified by using student credit.

4. What database should we use?

- a. MySQL
- b. MongoDB
- c. Redis
- d. Postgres

Choice: MongoDB

Justification: We chose MongoDB for its schema-less design, which allows us to store data in BSON. It is also widely used and integrates well with many other codebases. It is also highly scalable and performant, which is great for a web application. As for cost, MongoDB has a student pack which also includes free credit for us to use.

5. What image recognition algorithm should we use for identifying faces?

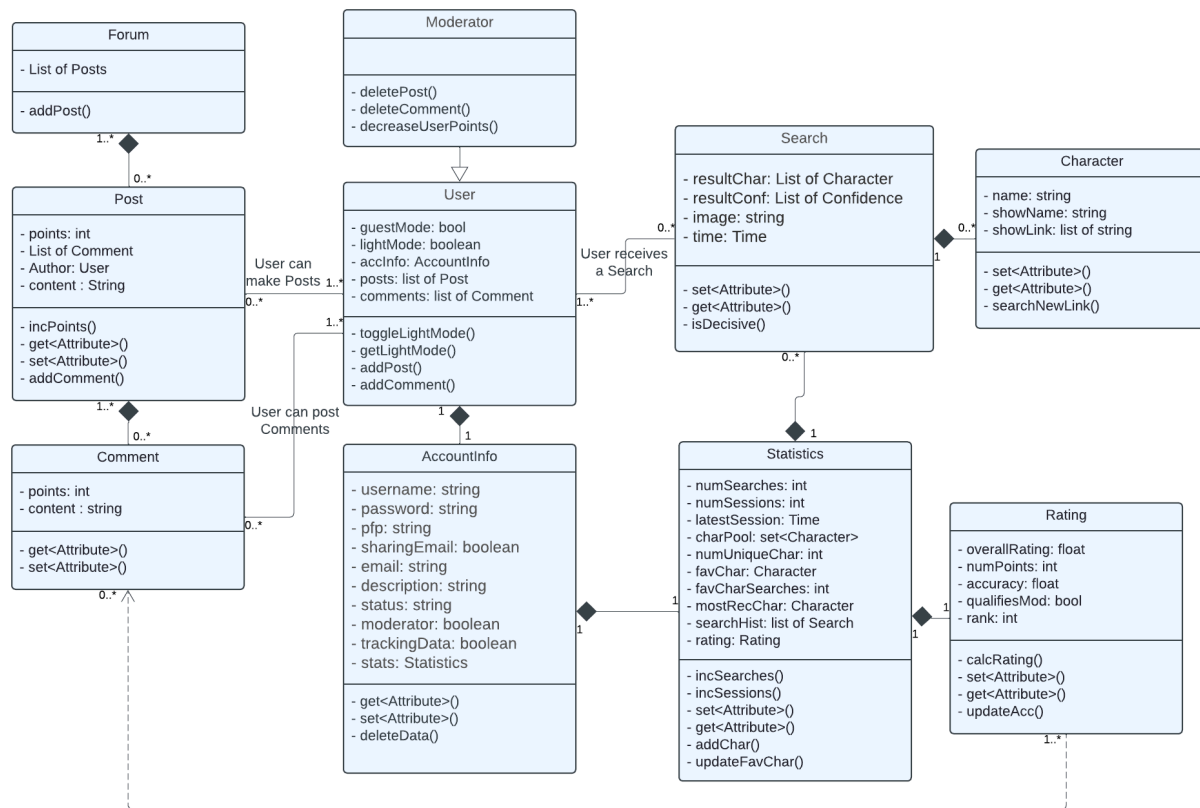
- a. Convolution Neural Network (CNN)
- b. Single Shot Detector (SSD)
- c. You Only Look Once (YOLO)

Choice: Convolution Neural Network (CNN)

Justification: We chose to use a CNN for our model as it specialized in identifying specific features of an image, which is what we need most when identifying a specific character of a show. A SSD would not have been as ideal as it mainly specializes in images with varying aspect ratios but our application mainly focuses on the faces of a character. We also decided against using YOLO as we did not need to identify the images instantly in real time and were not as accurate as a CNN.

4. Design Details

4.1 Class Diagram and Descriptions



4.2 Description of Classes and their Interactions

- **User**
 - This class is created when a user opens the application
 - Each user can turn on light mode or dark mode
 - Each user has account info if they did not use guest mode
 - Each user has a list of Posts if they did not use guest mode
 - Each user has a list of Comments if they did not use guest mode
 - User receives a Search (result) whenever they ask for one

- Moderator
 - This class is a subclass of User
 - Each moderator has special functions to help to make sure all posts and comments are accurate
- AccountInfo
 - This class holds all of a user's information if they log in
 - Each AccountInfo has a username, password, and profile picture
 - Each user can link and unlink their email from their account
 - Each user can add a description and status to their profile
 - Each user can choose for the app not to track their analytics, in which case the app will delete the data it has tracked
- Statistics
 - This class holds all of the user's analytics, which will be set to null if the user requests their statistics not to be tracked
 - This data structure contains the number of searches a user has performed, the number of times the user has used Annix, and the most recent time the user has opened Annix
 - This data structure also contains the set of characters that this user has found, which is used to find the number of unique characters the user has searched for
 - This data structure also contains the user's most searched character, which the app tracks by recording the number of times it has been searched
 - This data structure also records the user's most recent character searches, in the case that the character wishes to change their profile picture to this character
 - This data structure also has the user's rating, which may be used to apply to be a moderator
 - All of this information can be shown in a Data/Analytics page, if the user wishes to see it

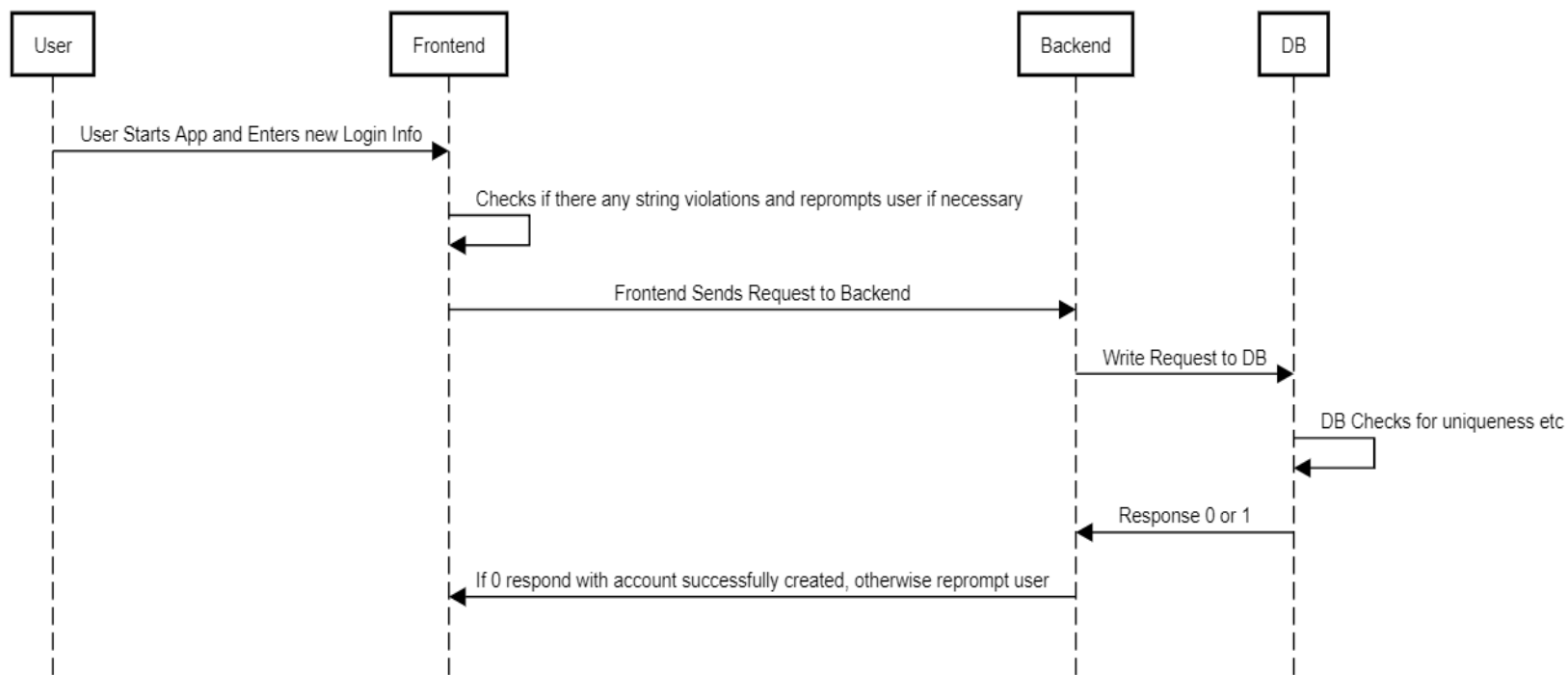
- Rating
 - This class contains the user's overall rating, which can be used for moderator application
 - The rating is calculated from the number of points they have, as well as the accuracy of the help they give
 - The rating determines their rank and whether they can be a mod
- Search
 - This class represents the data given when an image is passed to Annix
 - A Search contains a list of potential characters in decreasing order of confidence, as well as the respective confidences
 - A Search also contains metadata regarding the time this search took place
- Character
 - This class represents the character that users search for
 - A character consists of the character's name, the show they're from, and links that signify where to watch this show
- Forum
 - This class represents a group of posts, which users can look at
 - There will only be 1 shared forum, which will contain all posts made by users
- Post
 - This class represents a singular post
 - Each post will contain the author, the content of the post, a list of comments, and the number of points the user got from making this post
 - Posts can be deleted by moderators if it is not appropriate for Annix, whether it is derogatory or factually incorrect
- Comment
 - This class represents a comment on a post
 - Comments will also contain points that the user got from commenting, as well as the content itself
 - Comments can also be deleted by moderators for the same reasons as posts

4.3 Sequence Diagrams

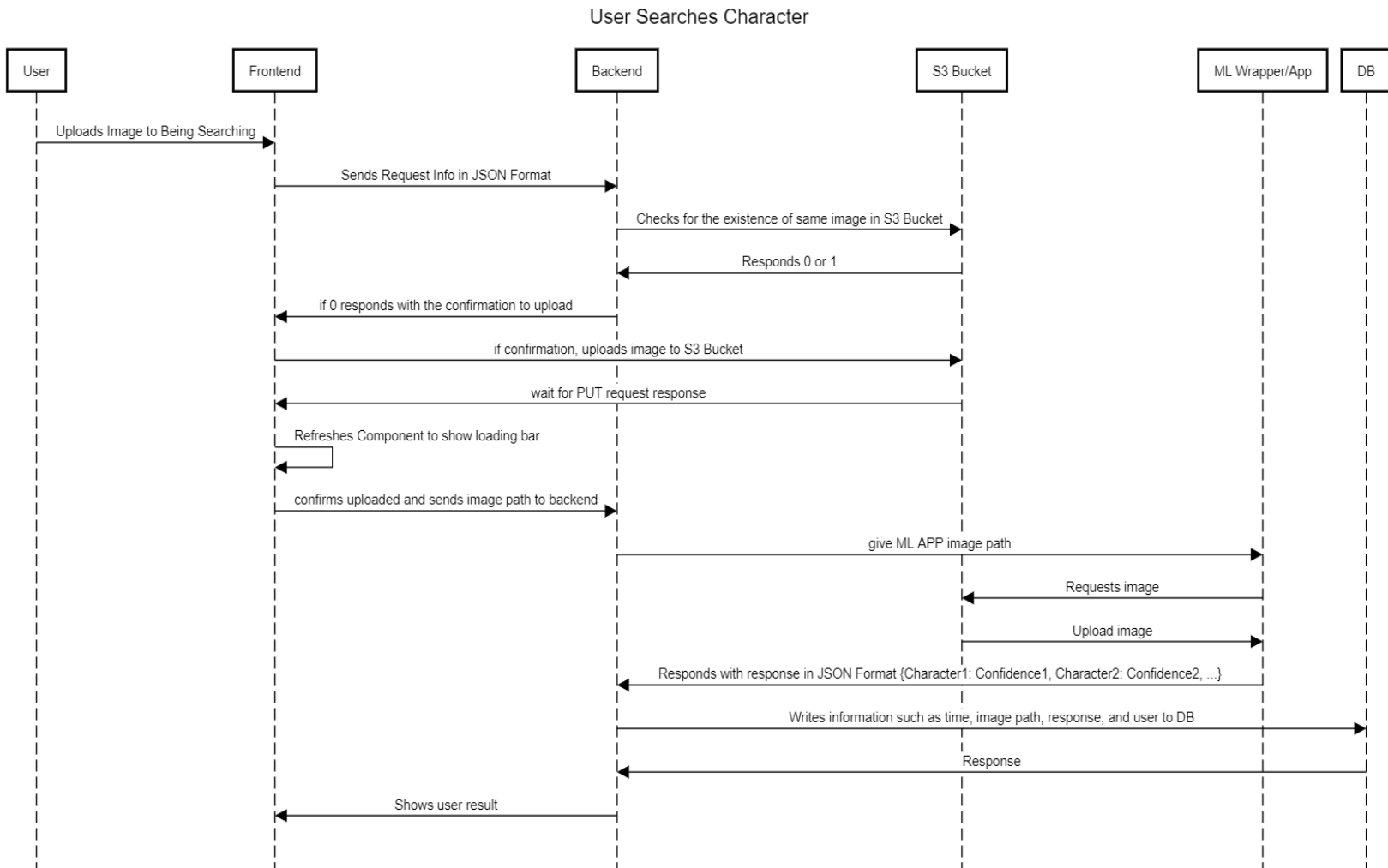
The diagrams below show the sequence of events happening within our system and applications following a user's desire for the system to perform an event.

1. User Creating Account

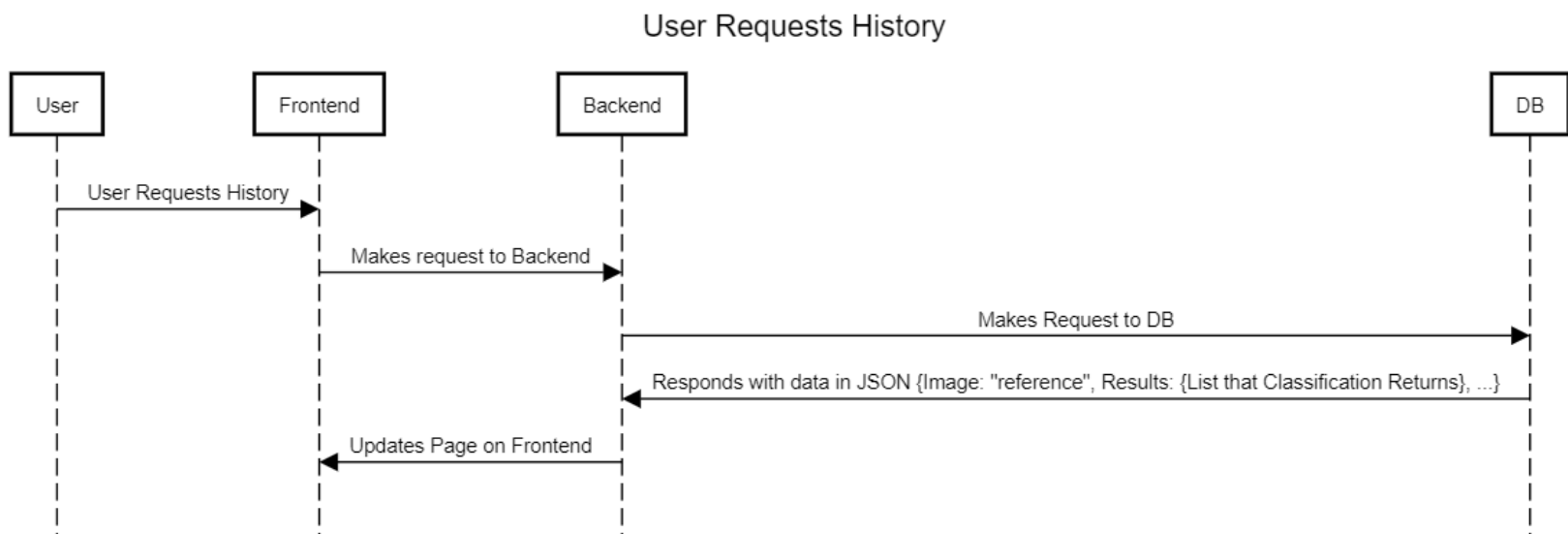
User Creating Account



2. User Searching Character

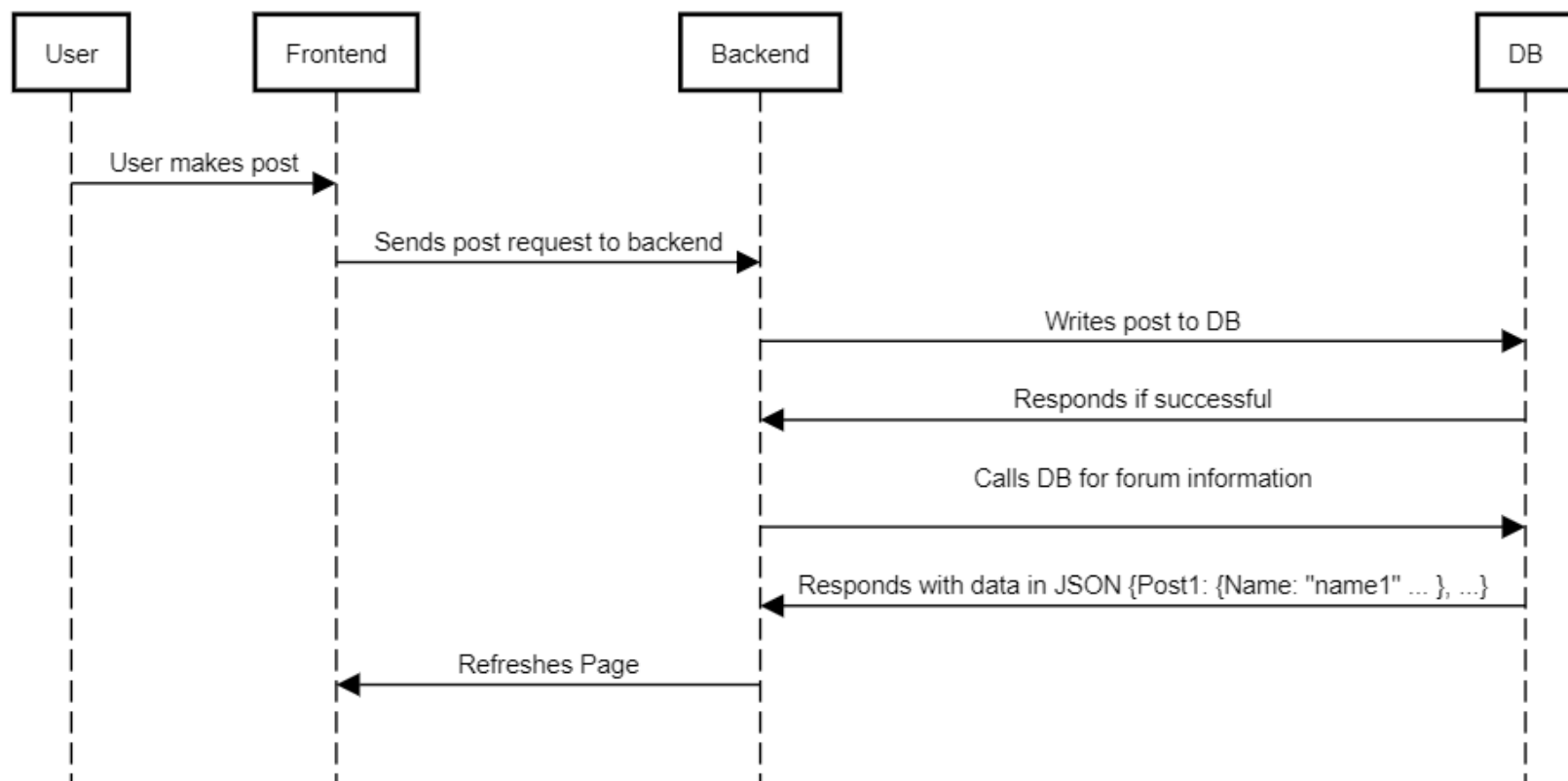


3. User Requests History

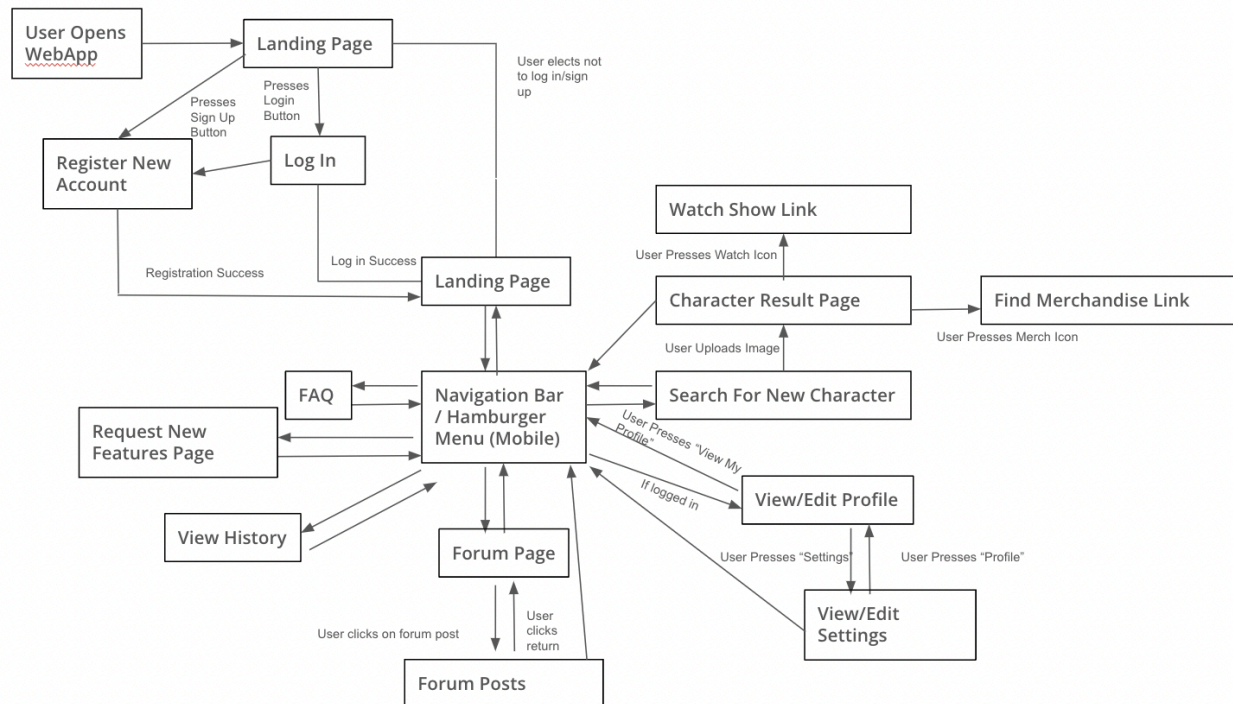


4. User Makes Post/Comment

User Makes Post



4.4 Navigation Flow Diagrams

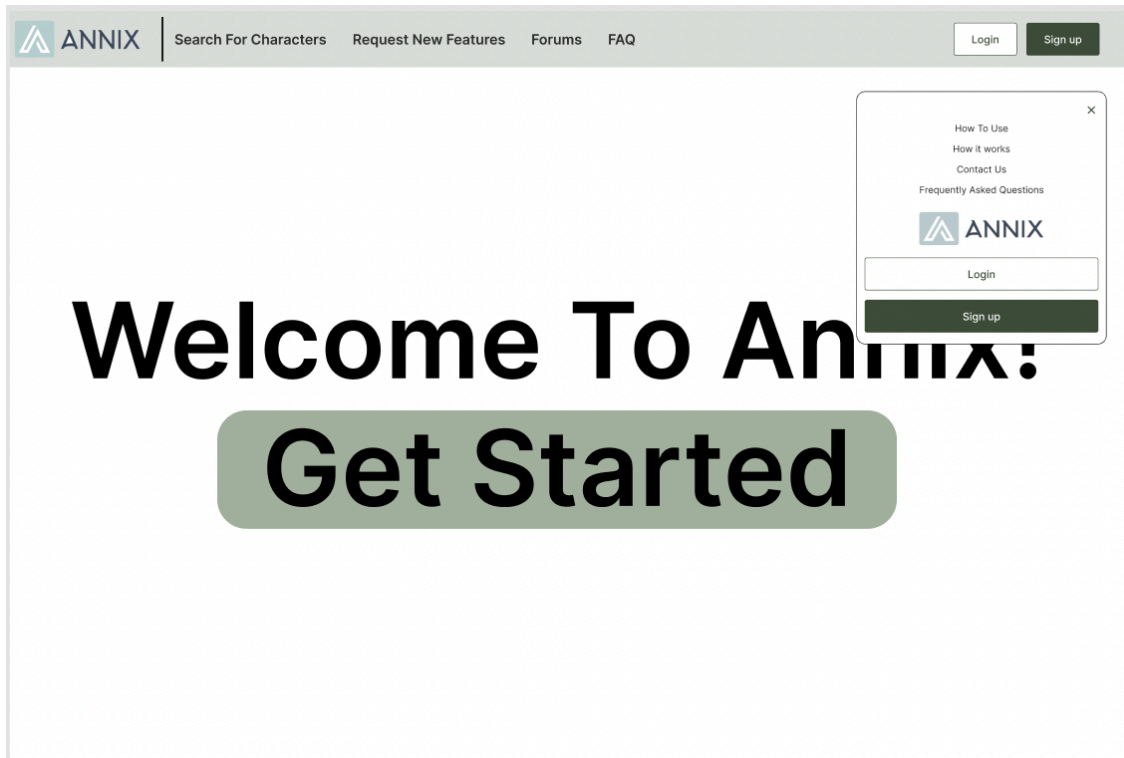


The Web App navigation consists of a Navigation Bar / Hamburger Menu that is present at the top of every page. (Note that the Navigation Bar isn't its own page – rather, it is a feature included above every page, but it was added to the diagram for clarity). This allows users to navigate freely and easily between the main pages on both desktop and mobile platforms with minimal problems, since it is standardized and easy to understand.

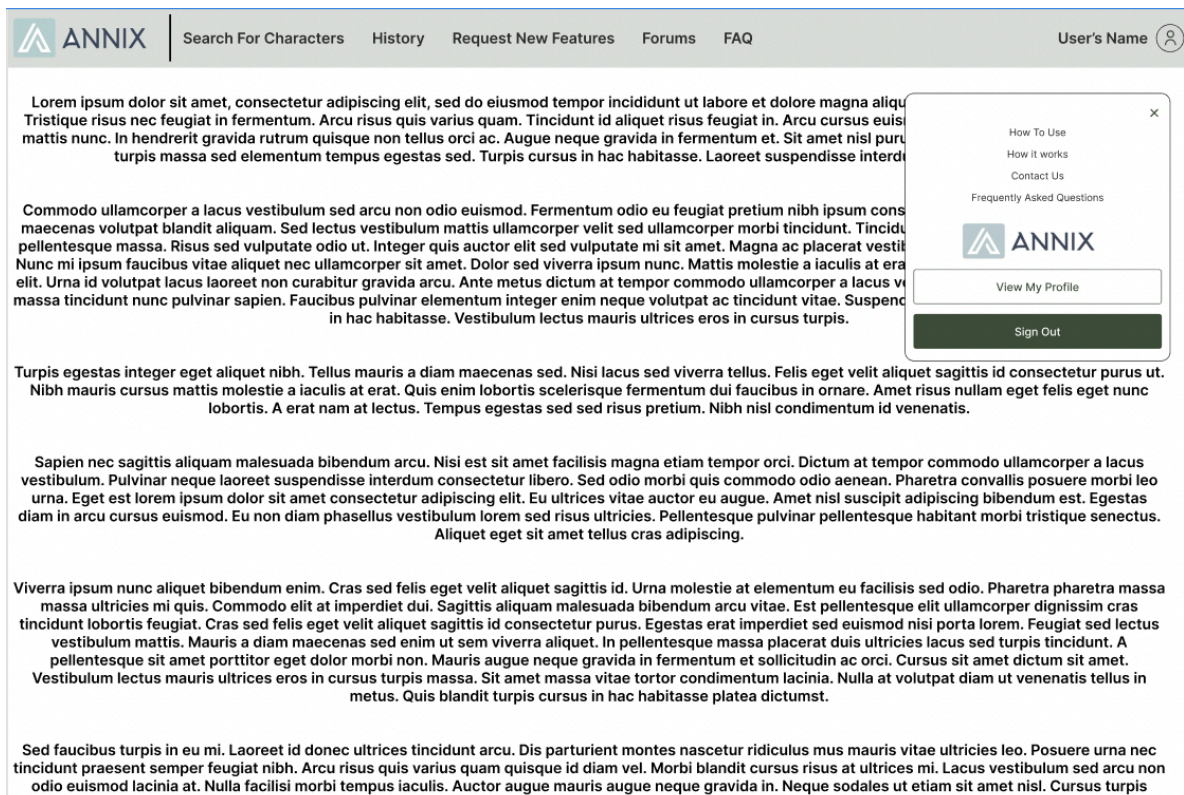
Some pages are only accessible from other pages – e.g. the character result page is only accessible by searching for a new character – but will still have the navigation bar to get back to any other functionality. This way, the bar is concise and secondary pages are navigated to by their related functions.

Users will have the option to log in and stay as a guest: the only difference will be that users who did not sign in would be unable to view or edit their profile, as one doesn't exist.

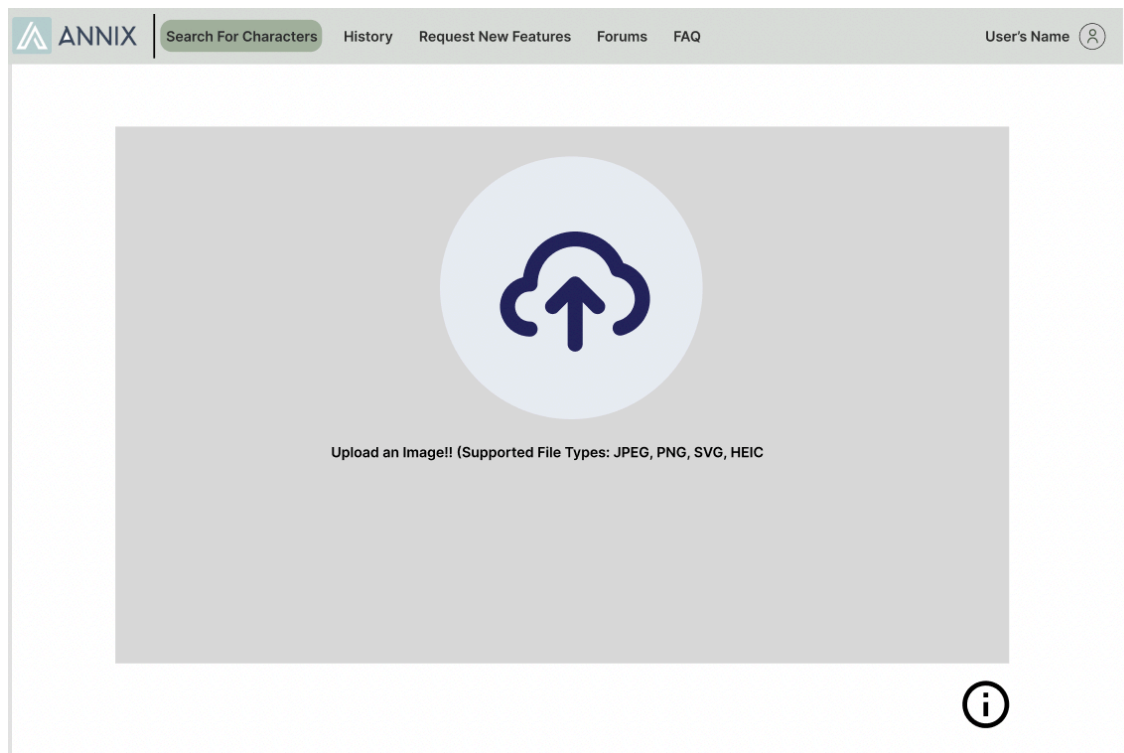
4.5 UI Mockups



Pictured above is a landing page of the site, which users will be directed to when they type in any invalid link beginning with the website url, or when they type the website url without any additional directories. Users will also be redirected to this page when a function that may not return to the previous page is completed – for example, logging in or creating an account.

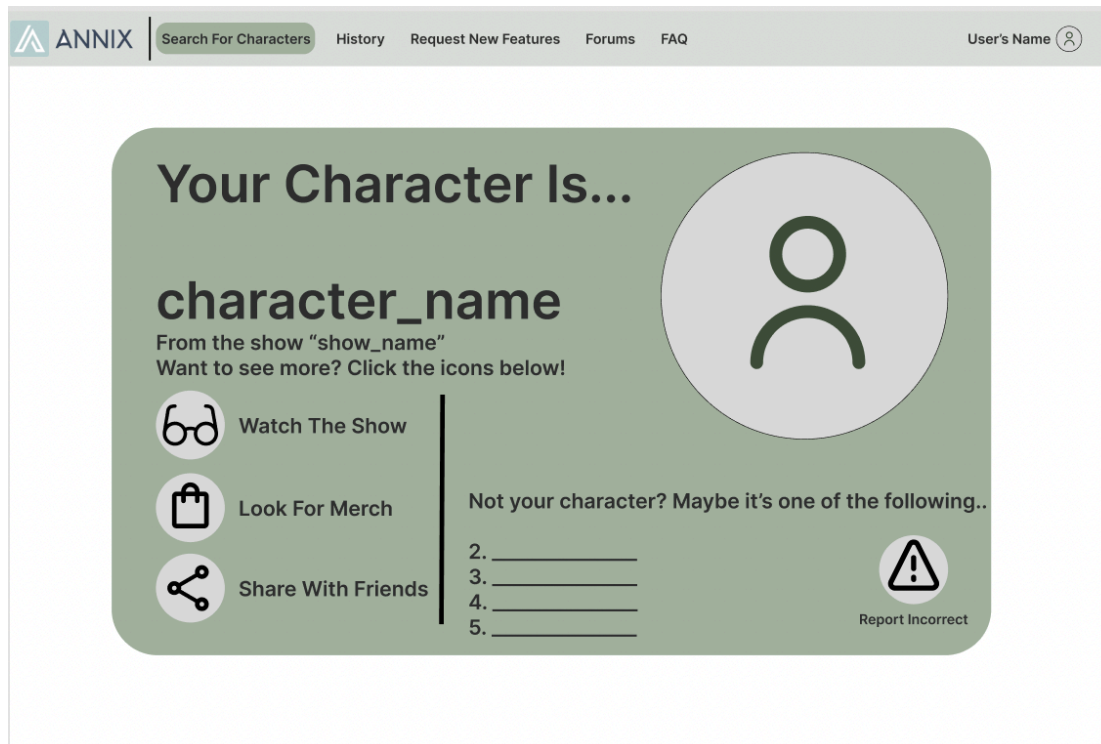


Above is a sample page to demonstrate how the website changes after the user logs in. More specifically, information pertaining to the user's details (name and profile picture) replaces the previous buttons of logging in and signing up. The buttons in the account pop-up also change to allow users to view their profile and sign out.



This is the main page for searching for characters, which can be reached by clicking the “Search For Characters” button in the navigation bar present on every page. The navigation bar highlights the button to signify that it is the page that the user is currently on.

The page allows the user to upload any image they would like by clicking on the icon to open their storage or by dragging and dropping. It gives brief information on the supported file types (JPEG, PNG, SVG, HEIC) as well as an additional icon that will pop up with additional information.

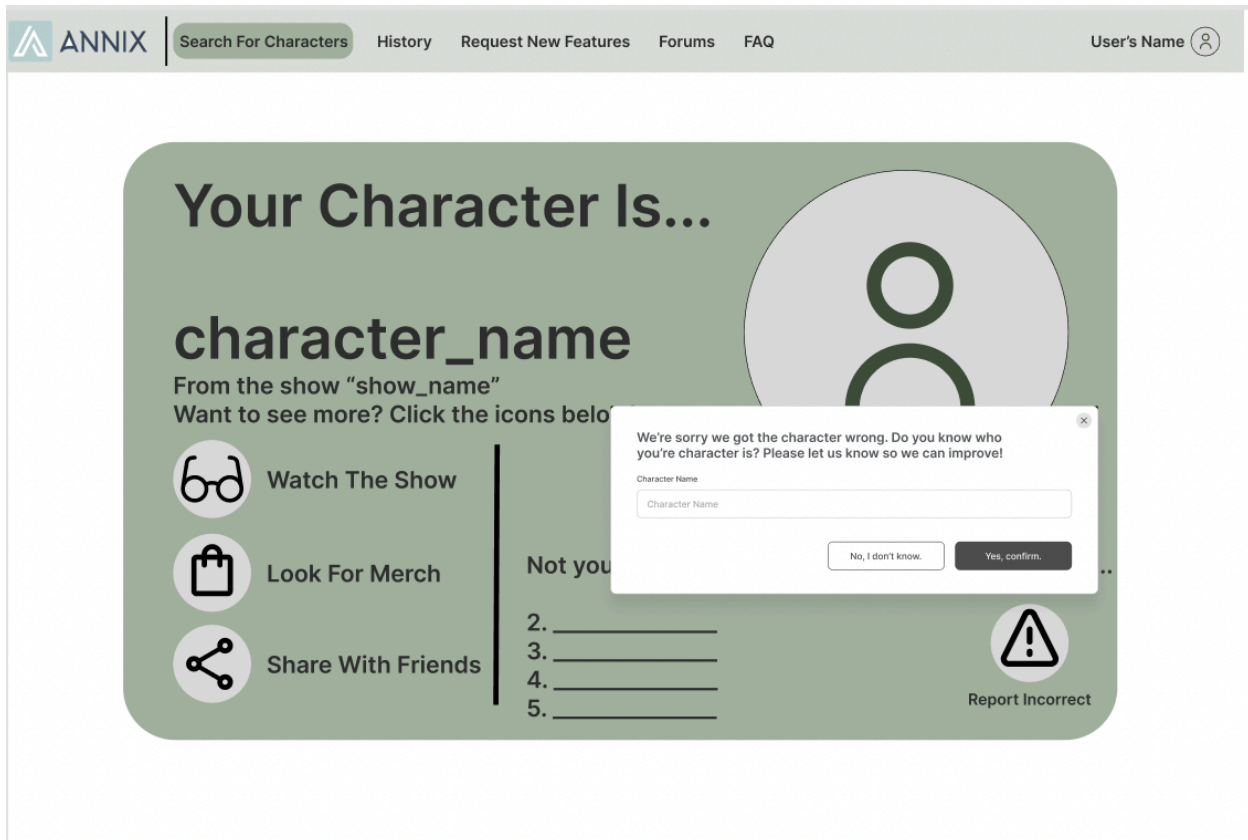


This is the page showing the user the character that they searched for. It appears after a user submits an image of a character, which is sent to the machine learning model. Data is received back in a JSON format, which is parsed and displayed on the page.

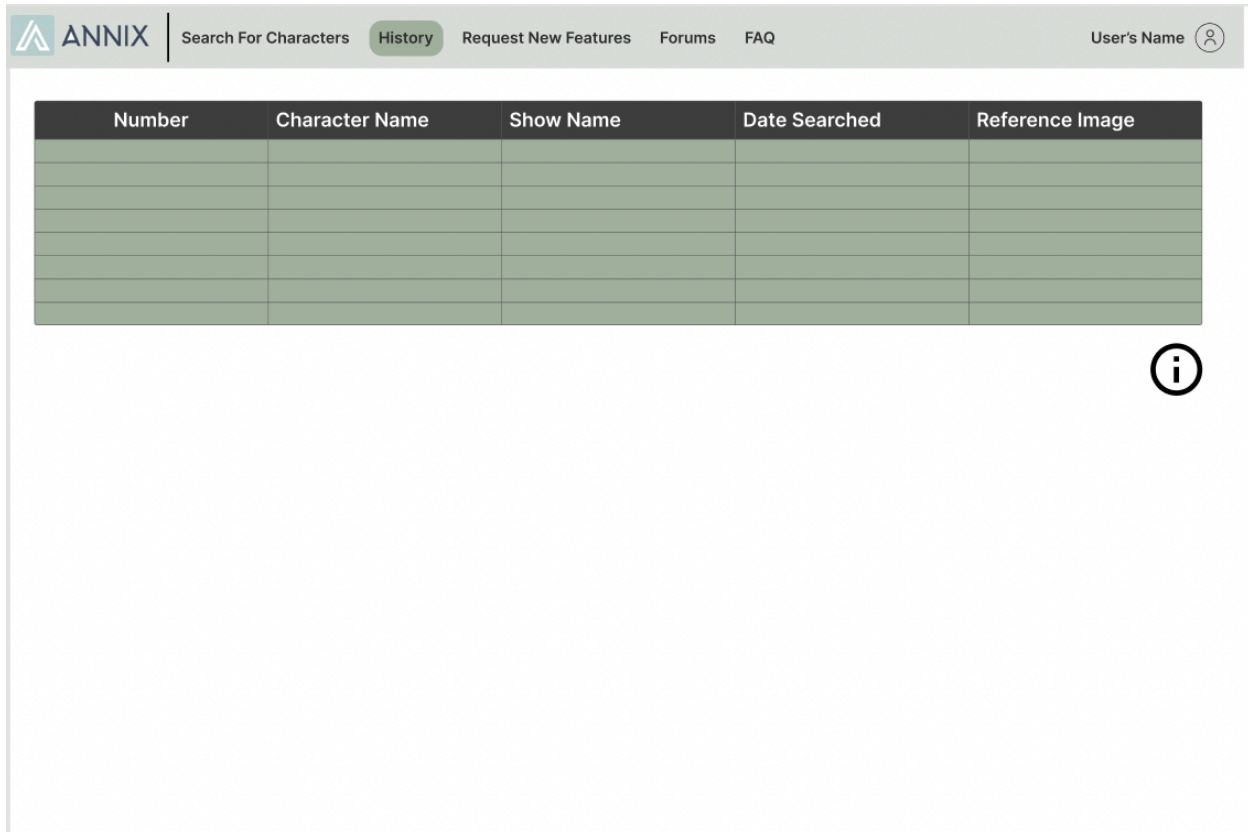
The page displays the character that the model thinks is most likely given, as well as the show it is from. It also contains links to watch the show or look for merchandise (likely on amazon), as well as a button to generate a link which allows users to share the page with friends.

The page also displays four other characters that the model thinks the image is likely to be (in descending order of likelihood), which gives a higher chance of satisfying users' queries.

Lastly, the page displays a button that will bring a pop-up, which the user should press if none of the characters are correct. This is displayed below.




Above is a demonstration of the pop-up for reporting incorrect character guesses. This is similar to how most pop-ups will look on the page, with text boxes and buttons to submit or cancel. In this specific case, there is text about the purpose of the pop-up, and a text field that users can fill in to help correct the model.




Above is an image of the history page, which can also be navigated to via the navigation bar. Again, The navigation bar highlights the button to signify that it is the page that the user is currently on.


It will store the user's previous searches (if they have allowed the storage of this data in settings) and display details about them in an organized field. These details include the returned character name, show name, the date of search and the reference image given, which will be stored in an S3 bucket. There is also an information button, which users can click to learn more information.


 ANNIX

Search For CharactersHistoryRequest New FeaturesForumsFAQ

User's Name 

Number	Character Name	Show Name	Date Searched	Reference Image






This is a sample info popup that shows up when you press the information icon. Many of these will be scattered around the website so that users can clear up any confusion as to what is going on quickly.


There will be a learn more button below if a quick summary doesn't fully cover all bases.

[Learn More](#)

This is an example of a page after a user presses on the information icon. It will display a closeable text field that contains information, as well as a learn more page if there is more information to be disclosed that does not fit concisely. These information icons are scattered throughout the website to hold any information that can be useful to the user – for example, answers to frequently asked questions or guides on how to use the website functions.


ANNIX

[Search For Characters](#)
[History](#)
[Request New Features](#)
[Forums](#)
[FAQ](#)

User's Name 

Here's a list of all the features we're working on!

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Tortor consequat id porta nibh. Tristique risus nec feugiat in fermentum. Arcu risus quis varius quam. Tincidunt id aliquet risus feugiat in. Arcu cursus euismod quis viverra nibh cras pulvinar mattis nunc. In hendrerit gravida rutrum quisque non tellus orci ac. Augue neque gravida in fermentum et. Sit amet nisl purus in mollis nunc sed id. Leo in vitae turpis massa sed elementum tempus egestas sed. Turpis cursus in hac habitasse. Laoreet suspendisse interdum consectetur libero.

Commodo ullamcorper a lacus vestibulum sed arcu non odio euismod. Fermentum odio eu feugiat pretium nibh ipsum consequat nisl. Ut enim blandit volutpat maecenas volutpat blandit aliquam. Sed lectus vestibulum mattis ullamcorper velit sed ullamcorper morbi tincidunt. Tincidunt augue interdum velit euismod in pellentesque massa. Risus sed vulputate odio ut. Integer quis auctor elit sed vulputate mi sit amet. Magna ac placerat vestibulum lectus mauris ultrices eros in. Nunc mi ipsum faucibus vitae aliquet nec ullamcorper sit amet. Dolor sed viverra ipsum nunc. Mattis molestie a iaculis at erat pellentesque adipiscing commodo elit. Urna id volutpat lacus laoreet non curabitur gravida arcu. Ante metus dictum at tempor commodo ullamcorper a lacus vestibulum. Commodore quis imperdiet massa tincidunt nunc pulvinar sapien. Faucibus pulvinar elementum integer enim neque volutpat ac tincidunt vitae. Suspendisse sed nisi lacus sed viverra tellus in hac habitasse. Vestibulum lectus mauris ultrices eros in cursus turpis.

Turpis egestas integer eget aliquet nibh. Tellus mauris a diam maecenas sed. Nisi lacus sed viverra tellus. Felis eget velit aliquet sagittis id consectetur purus ut. Nibh mauris cursus mattis molestie a iaculis at erat. Quis enim lobortis scelerisque fermentum dui faucibus in ornare. Amet risus nullam eget felis eget nunc lobortis. A erat nam at lectus. Tempus egestas sed sed risus pretium. Nibh nisl condimentum id venenatis.

Don't see the feature you want? Suggest One!

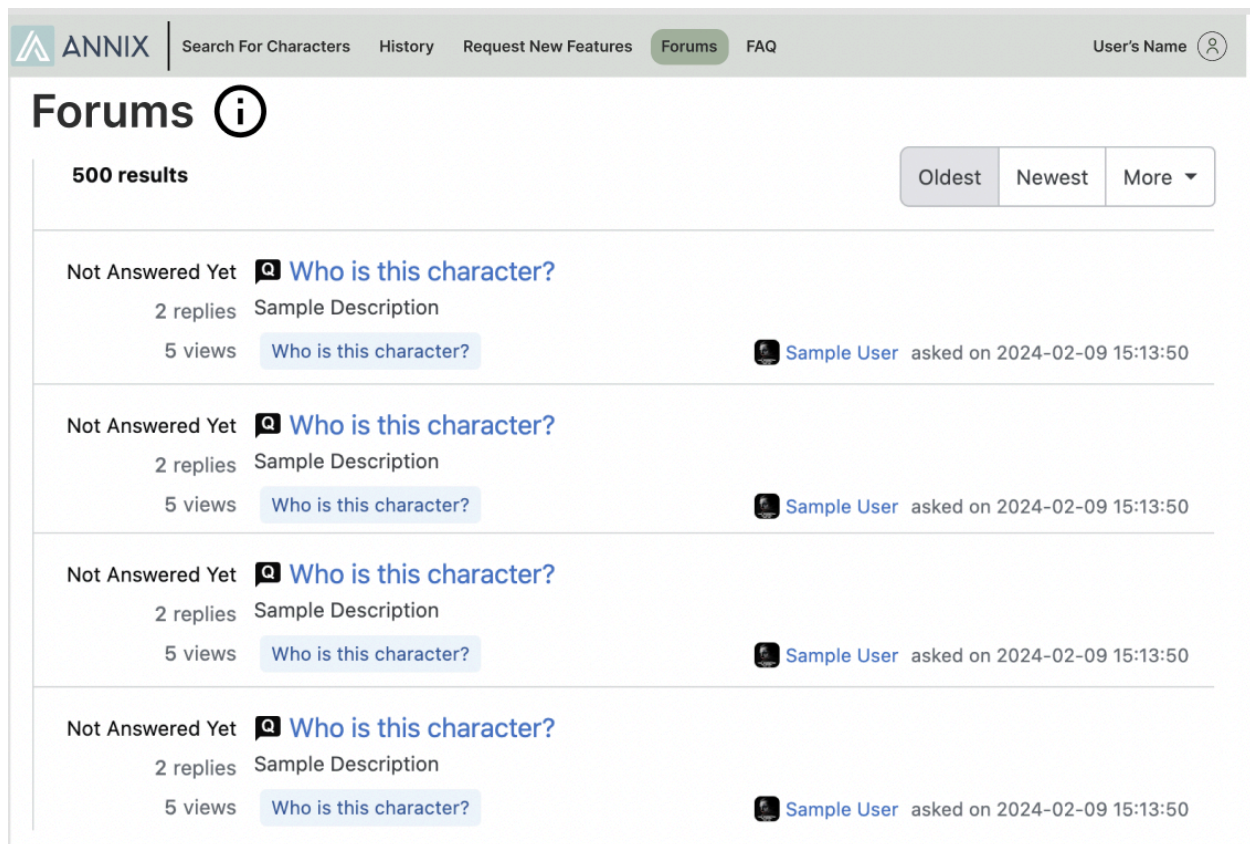
Suggest A Feature!

Write your name! (Optional)

Write your suggestion!

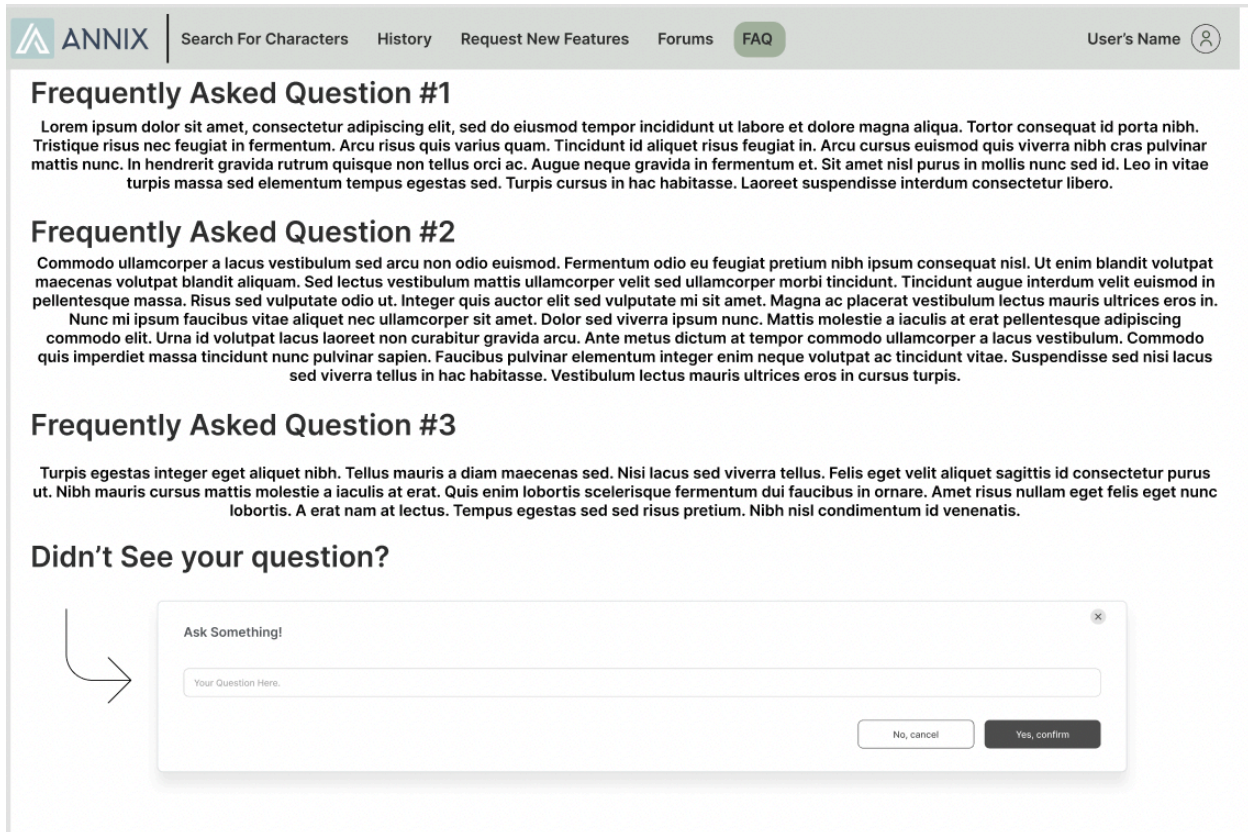
Above is an image of the “Request New Features” page, which can be navigated to via the navigation bar. Again, The navigation bar highlights the button to signify that it is the page that the user is currently on.

This page contains information on features that are currently being worked on by the team, as well as a text box and field that allows users to submit feature requests. This helps users find out more information, and also helps the developers to look through features as users are less likely to send in repeat suggestions.



Above is an image of the “Request New Features” page, which can be navigated to via the navigation bar. Again, The navigation bar highlights the button to signify that it is the page that the user is currently on.

The forum page contains links to a variety of user generated posts which any user can interact with. This involves replying and saving certain posts. There will also be a search bar to look through the forum to find specific forum posts.



Above is an image of the “Frequently Asked Questions” page, which can be navigated to via. the navigation bar. Again, The navigation bar highlights the button to signify that it is the page that the user is currently on.

This page contains frequently asked questions that the developers think will be useful information to have. It also has an option for users to ask questions, and for these questions to be added to the page.

ANNIX | Search For Characters | History | Request New Features | Forums | FAQ

User's Name

ANNIX

Username Here

My Profile | Settings | Request Deletion | Cancel | Save Changes

Username: Password: [Request Change](#)

Email:

Upload A Banner Image or Profile Picture!

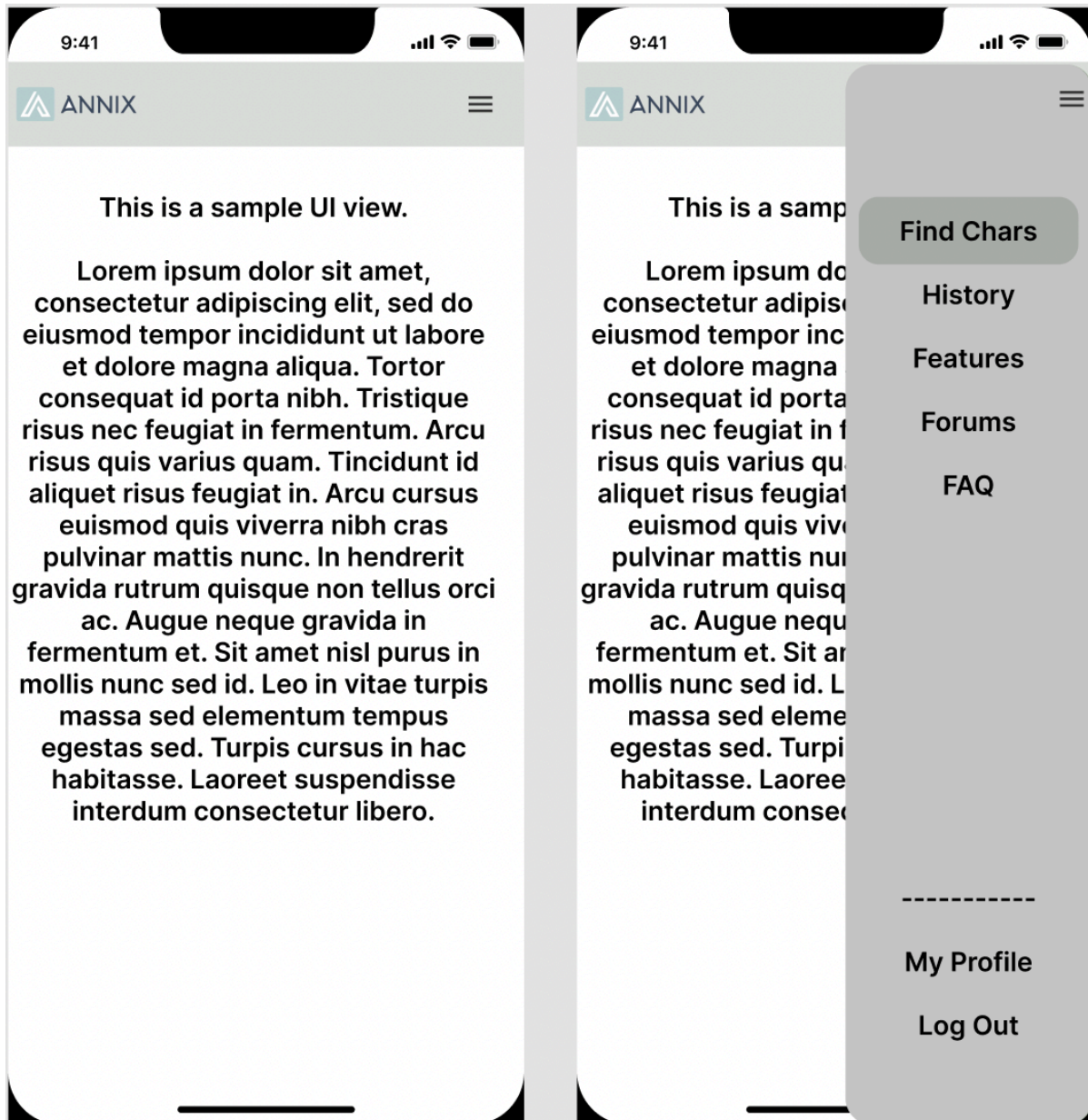
Click to upload or drag and drop
SVG, PNG, JPG or GIF (max. 800x400px)

Alternatively, Choose a character you've searched:

About Me

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Tortor consequat id porta nibh. Tristique risus nec feugiat in fermentum. Arcu risus quis varius quam. Tincidunt id aliquet risus feugiat in. Arcu cursus euismod quis viverra nibh cras pulvinar mattis nunc. In hendrerit gravida rutrum quisque non tellus orci ac. Augue neque gravida in fermentum et. Sit amet nisl purus in mollis nunc sed id. Leo in vitae turpis massa sed elementum tempus egestas sed. Turpis cursus in hac habitasse. Laoreet suspendisse interdum consectetur libero.

Above is an image of the Profile page, which can be navigated to by clicking the user icon in the top right, and then clicking “View My Profile” in the ensuing pop-up. The profile contains user data that the user can modify, as well as the settings page that allows the users to modify their web experience. It also contains ways to view and change various user data, like usernames, passwords, etc, and a way for the user to request deletion of any and all of their data.



Above is a sample of how the web app will look on mobile. Mostly, pages, contents and their pop-ups remain the same. The key difference that is being illustrated here is the way users navigate the website – instead of using a navigation bar, which will not be able to fit ergonomically, we will employ the use of a hamburger menu on mobile. The image in the right shows what will show up when the hamburger menu is pressed – that being buttons to direct users to all pages that are accessible via the navigation bar on their desktop.