

# Critika

CS 30700: Software Engineering 1  
Design Document

**Team 8:**

Brandon Sung  
Connor Todd  
Cyrus Santiago  
Keith Tan  
Ken Sodetz  
Julien San Diego

# Index

- Purpose Page 3
  - Functional Requirements
  - Non-Functional Requirements
- Design Outline Page 6
  - High-Level Overview
- Design Issues Page 7
  - Functional Issues
  - Non Functional Issues
- Design Details Page 11
  - Class Level Diagrams
  - Sequence Diagrams
  - State Diagrams
  - UI Mockups

# Purpose

There are currently numerous online forums for users to interact with each other, learn about anything they want, and share their personal experiences and stories. Large examples of these are Reddit, Facebook, Twitter, and Instagram among others. They all have their own strengths and weaknesses. For example, a large number of subscribers can be a strength as the platform will have ongoing interaction. However, this can also be a weakness, as any given post or tweet can get lost in a sea of other user's posts and comments. For example, a post on a popular subreddit or a tweet using a trending hashtag will lose visibility as thousands of other people are posting on the same subreddit and tweeting using the same hashtag. Users on large forums looking for advice or a comment on their post will struggle as their post will be lost in the feed. Our platform aims to help users looking for advice by increasing the visibility of their post, and by encouraging other users to provide thoughtful and thorough critique.

Critika requires that if users want to make a post looking for feedback or commentary, the user must first critique other's posts. If you don't comment on others posts, then you will not be allowed to post and therefore receive critique. This will encourage users to give feedback to others, and reduce the number of posts in the queue. Critika will help users receive feedback, and encourage collaboration.

# Functional Requirements

1. Users can create an account and login
  - a. As a user, I would like to be able to register for a Critika account.
  - b. As a user, I would like to be able to login and manage my Critika account.
  - c. As a user, I would like my password to be reset if I forget it.
2. Users can submit a form to be critiqued
  - a. As a user, I would like to be able to submit a form for submission.
  - b. As a user, I would like to be able to remove submissions as I see fit.
  - c. As a user, I would like to be able to easily access my submissions.
  - d. As a user, I'd like to be able to showcase some of my best works on my profile.
  - e. As a user, I would like a system to further specify my submission within the general category.
  - f. As a user, I would like a way for people to see my submission and comment on it but not necessarily give me feedback.
3. Users can rate other submissions
  - a. As a user, I would like to be able to rate the feedback I've been given.
  - b. As a user, I would like to see the best critiquers for a given category.
  - c. As a user, I would like to specify what kind of feedback I'm looking for.
  - d. As a user, when I'm critiquing I would like my rating to be hidden from the other user.
  - e. As a user, when I'm critiquing, I would like the option to have my information anonymous from the person I'm critiquing.
  - f. As a user, I would like a way for people to see my submission and comment on it but not necessarily give me feedback.
  - g. As a user, if I don't consider myself good enough to get feedback, I want a way to be able to get feedback through the give and take system.
4. Users can interact with other users
  - a. As a user, I would like to be able to add other users so that I can collaborate with them further.
  - b. As a user, I would like to be able to message other users I've added.
  - c. As a user, I would like a way to find groups of people who share the same types of work.

- d. As a user, I would like a way to make more specialized groups of people who share the same types of work.
- e. As a user, I would like to be able to report bad feedback.

## Non Functional Requirements

### 1. Client Requirements

As a developer

- I want the application to be used on the web

### 2. Server Requirements

As a developer

- I want the server to save the user submissions to a local database
- I want the server to save the Critikas to a local database
- I want the server to save all comments to a local database
- I want the server to save all users and userinfo to the local database

### 3. Design Requirements

As a developer

- I want the build scripts to deploy the server from the Git repository

### 4. Performance Requirements

As a developer

- I want the server to be able to handle as many concurrent users at once as allowed
- I want the server to not create excessive API requests
- I want the server and client to gracefully handle errors

### 5. Appearance Requirements

As a developer

- I want the application to be aesthetically appealing for all users

### 6. Security Requirements

As a developer

- I want a way to keep user information secure

# Design Outline

## High Level Overview

Our application will be a web app that allows users to post links to different kinds of media, and post text which other users will critique. The application will run on a client server model, with the server simultaneously handling information from a large number of clients. The client will send a request to the server, which will then send a query to our database in order to access the correct data. The database will return the data to the server, and the server will send a response back to the client.



- Client
  - Provides user with an interface
  - Sends request to the server
  - Receives the response from the server, and parses the response to display the correct information on the interface
- Server
  - Receives and handles request from the client
  - Server parses request and forwards a query to the database
  - Receives data from the database, and forms a response to send to the client
  - Sends the response to the client
- Database
  - Stores all user info, along with communities, messages, notifications, submissions, feedback, comments, etc
  - Returns the correct information to the server

# Design Issues

## Functional Issues

1. What information do we need for a user to sign up for an account?
  - Option 1: Username and password only
  - Option 2: Username, password, email address
  - Option 3: Username, password, email address, security question

Choice: Option 3

Justification: Username and password are necessary for accounts in order to create a layer of protection and identification for each user. Obtaining the user's email address will assist in implementing verification for the user accounts. Furthermore, it can be used for notifications, news updates on the application and password reset information. A security question we feel would add a good layer of security for our users and would not be very difficult to add. So, we added it to the information needed. Overall, option 3 should provide the necessary information for signing up for a users.

2. What kind of personal messaging system do we want to use?
  - Option 1: Instant messaging
  - Option 2: Non real time messaging system (like emails or private messages)

Choice: Option 2

Justification: When thinking about how users will interact with the application, we don't believe users would be frequently chatting with other users. Thus, having an instant messaging feature would not be used as much. Also, our backend framework isn't realtime as well which makes implementing instant messaging very difficult. So, going with a non real time messaging system is ideal as we still need a messaging esque feature for users to interact with each other.

3. How do we want to calculate user's Critika score?
  - Option 1: Take an average of all the scores (scores are out of 5) their critiques have received (with a minimum number of comments)
  - Option 2: Use the total number of scores they've ever received

Choice: Option 1

Justification: The user's score is an important feature in our framework. Users will not receive a score until they've critiqued a certain number of times, but once they hit that threshold, taking the average will provide us with the best information about how useful their individual critiques are. In order to increase their score, users will have to offer a lot of good feedback. If we used option 2, in the long run everyone would have a very high score, making that method less sustainable.

4. How should we match users to submissions?

- Option 1: By category, user feedbacks given and skill level for that particular category
- Option 2: By rating of two users
- Option 3: By the number feedback the critiquer has given for that category

Choice: Option 1

Justification:

We don't believe that rating should be used to match users to another user's submission. This would pose a problem as users with lower ratings would only get feedback from other users with lower ratings which could be detrimental to them improving as they may get low quality feedback. Furthermore, we don't want to solely use the number of feedback that a critiquer has given. So, the combination of category, user feedbacks given and skill level for that particular category we found was the optimal option.

5. Should we allow users to upload their content for submissions to the site?

- Option 1: Yes
- Option 2: No, they should upload on external sources

Choice: Option 2

Justification: No. While hosting user content for submissions may make the user experience a lot more seamless, if our user base gets very large, it will be very expensive to store all the files for submission. So, we are opting to not allow users to upload content onto our application for submissions.

## Non Functional Issues

1. Which frontend framework should we use?

- Option 1: React
- Option 2: Angular
- Option 3: Vue



Choice: Option 1

Justification: We chose React as the majority of our team has some kind of experience with React which will make more time for building the application and less time learning how to use the actual framework. Furthermore, it's in Javascript which lends well to our backend of node/express which also uses Javascript. Lastly, its component architecture will allow us to save time during development through code reuse and help keep our code organized.

2. Which backend framework should we use?

- Option 1: Django
- Option 2: Node
- Option 3: Spring

Choice: Option 2

Justification: We chose Node for the backend as it is extremely lightweight and like our frontend framework uses Javascript which will help ease the learning curve for the developers on our team not well versed in the backend as most know Javascript and allows for the teams to be cross functional. Furthermore, Node has a lot of modules accessible through its package manager that we can leverage to decrease development time.

3. Which database should we use?

- Option 1: MongoDB
- Option 2: MySQL

Choice: Option 1

Justification: Since we are going to have submissions for different mediums with a variety of different types of attributes for each, having a schemaless database such as MongoDB is very useful. Also, MongoDB is very easy to scale and is frequently used with the frontend framework we have chosen in the form of the MERN stack. Furthermore, many of our members have experience using this database and thus it would require the least amount of setup time.

4. Which web service should we use?

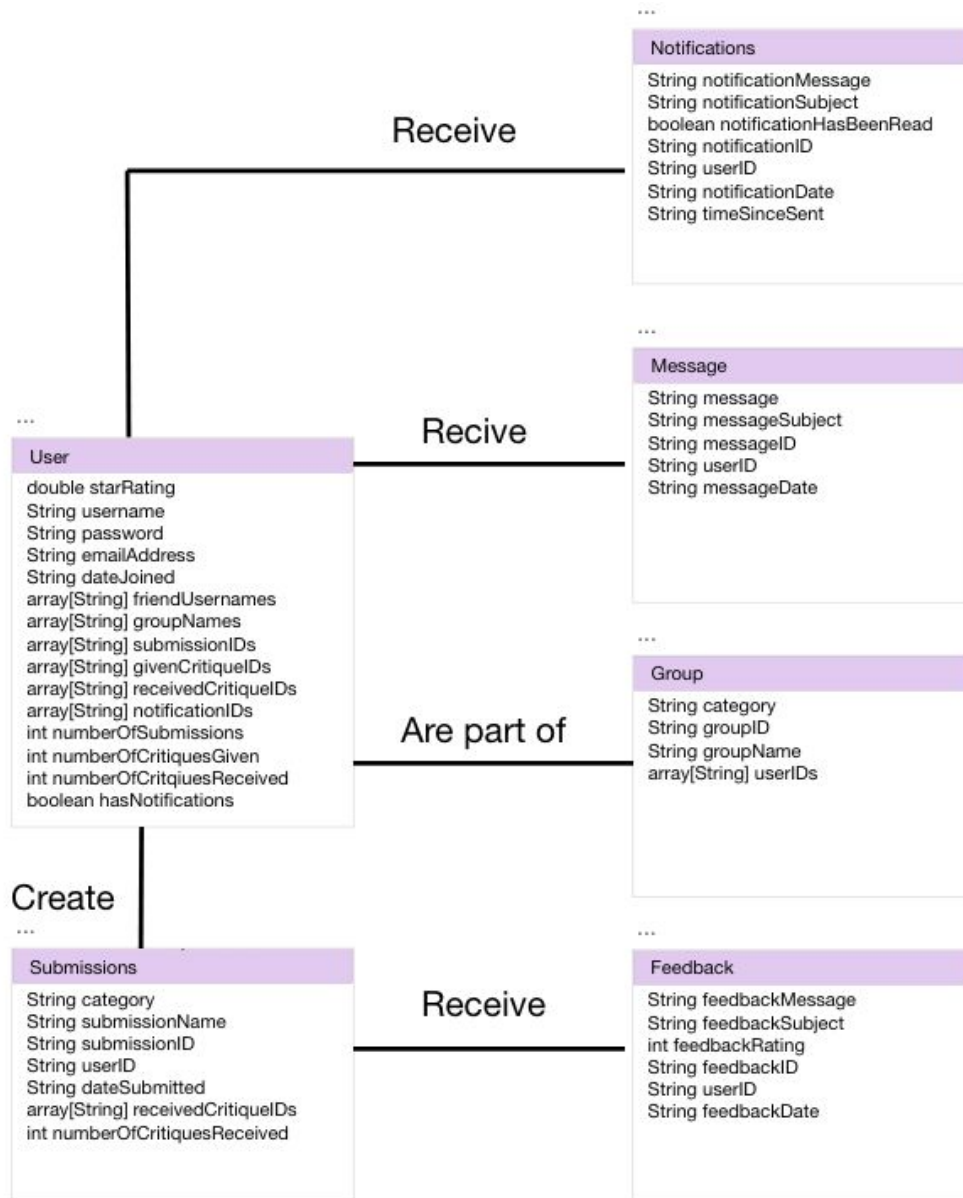
- Option 1: Azure
- Option 2: AWS
- Option 3: Heroku
- Option 4: Google Cloud

Choice: Option 3

Justification: Heroku out of the options listed allows for the simplest setup as it is a PaaS which will allow us to focus on building the actual application and less time worrying about the infrastructure. Furthermore, it is free and includes continuous deployment with our version control of choice (GitHub)

# Design Details

## Class Level Diagrams



## Description of Data Classes and their Interactions

### User

- User object is created when someone up in our application
- Each user will be assigned a unique user ID
- Each user will have a username, password, and email for login purposes
- Each user will have an array to store data to reference their friends, submissions, critiques, and notifications
- Each user will have counters to keep track of their submissions and critiques
- Each user will have a flag to determine whether or not they have unseen notifications

### Group

- Group object is created when a user creates a group
- Each group will have a category attribute
- Each group will be assigned a unique group ID
- Each group will have a group name
- Each group will have an array to store data to reference users who are a part of the group

### Submission

- Submission object is created when a user uploads a submission
- Each submission will have a category attribute
- Each submission will have a name and a unique submission ID
- Each submission will store the userID of the user who submitted
- Each submission will store the date it was created
- Each submission will have an array to store data to reference critiques on the submission
- Each submission will keep track of the number of received critiques

### Feedback

- Feedback object is created whenever a user submits feedback
- Each feedback object will have a feedback message. This is the critique itself
- Each feedback object will have a unique feedback ID

- Each feedback object will have a subject for users to easily know what the feedback is about
- Each feedback object will have a rating based on how helpful users found the feedback to be
- Each feedback object will store the ID of the user who wrote the feedback
- Each feedback object will store the date it was created

### **Message**

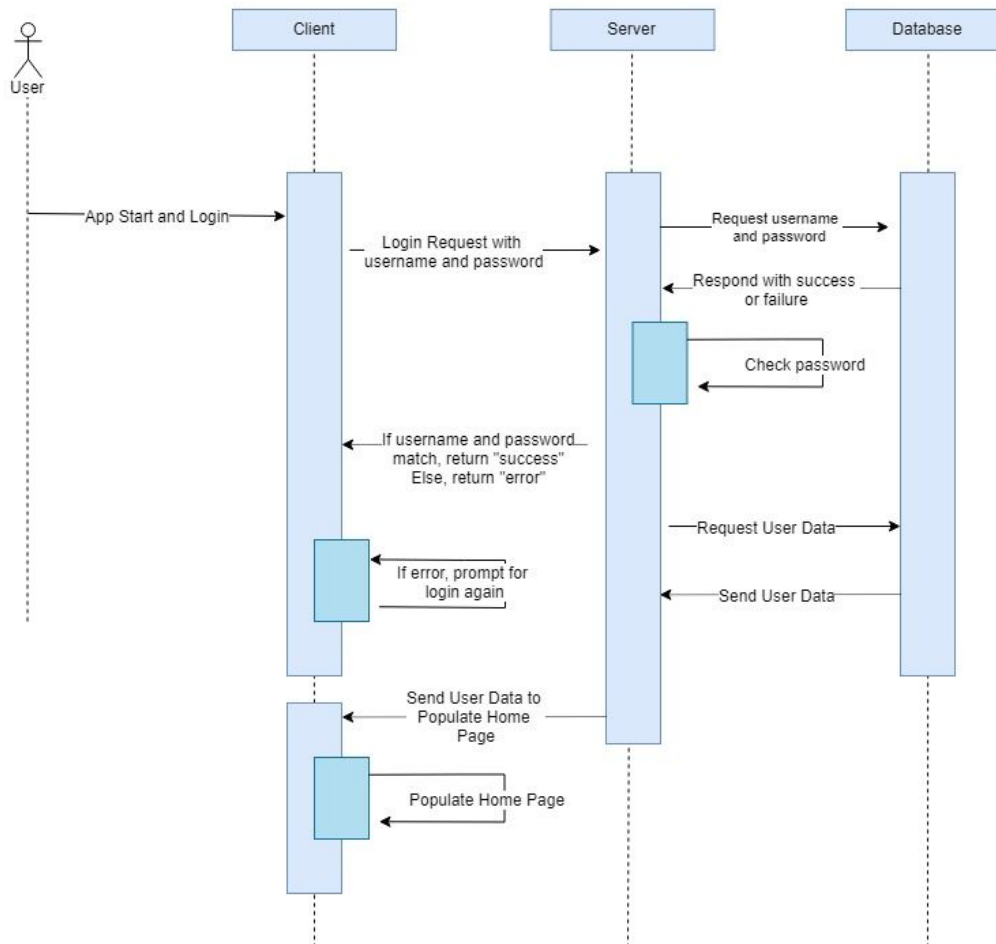
- A message object is created whenever a user writes a message
- Each message object will have a message stored as a String
- Each message object will have a subject for users to easily know what the message is about
- Each message object will have a unique message ID
- Each message object will store the ID of the user who wrote the message
- Each message object will store the date it was created

### **Notification**

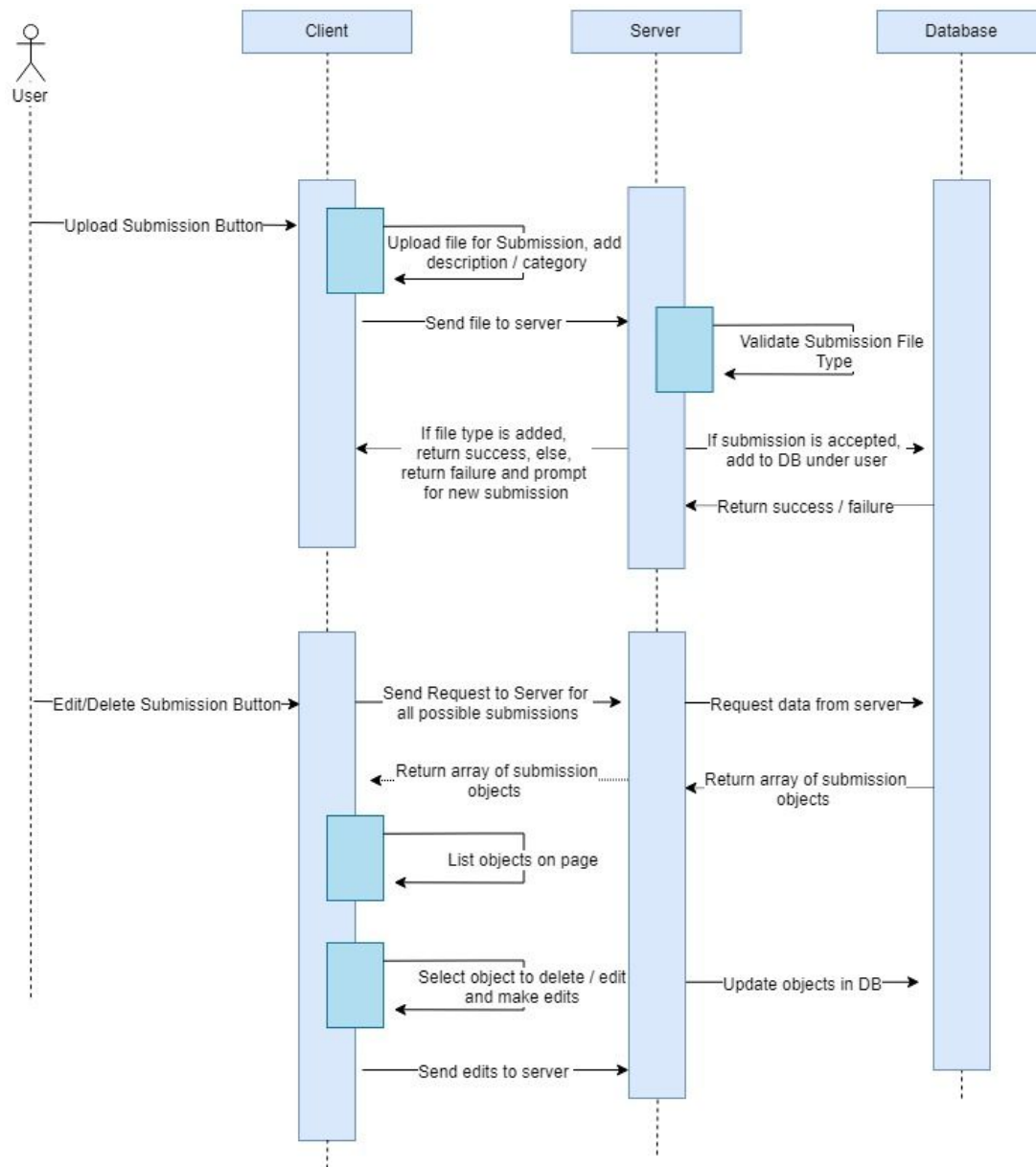
- A notification object is created whenever a user receives a message
- Each notification object will have a message of the notification
- Each notification object will have a subject
- Each notification object will have a boolean flag to determine if it has been seen or not by a user
- Each notification object will have a unique notification ID
- Each notification object will store the ID of the user who received it
- Each notification will store the date it was created
- Each notification will track how much time has elapsed since it was sent

# Sequence Diagrams

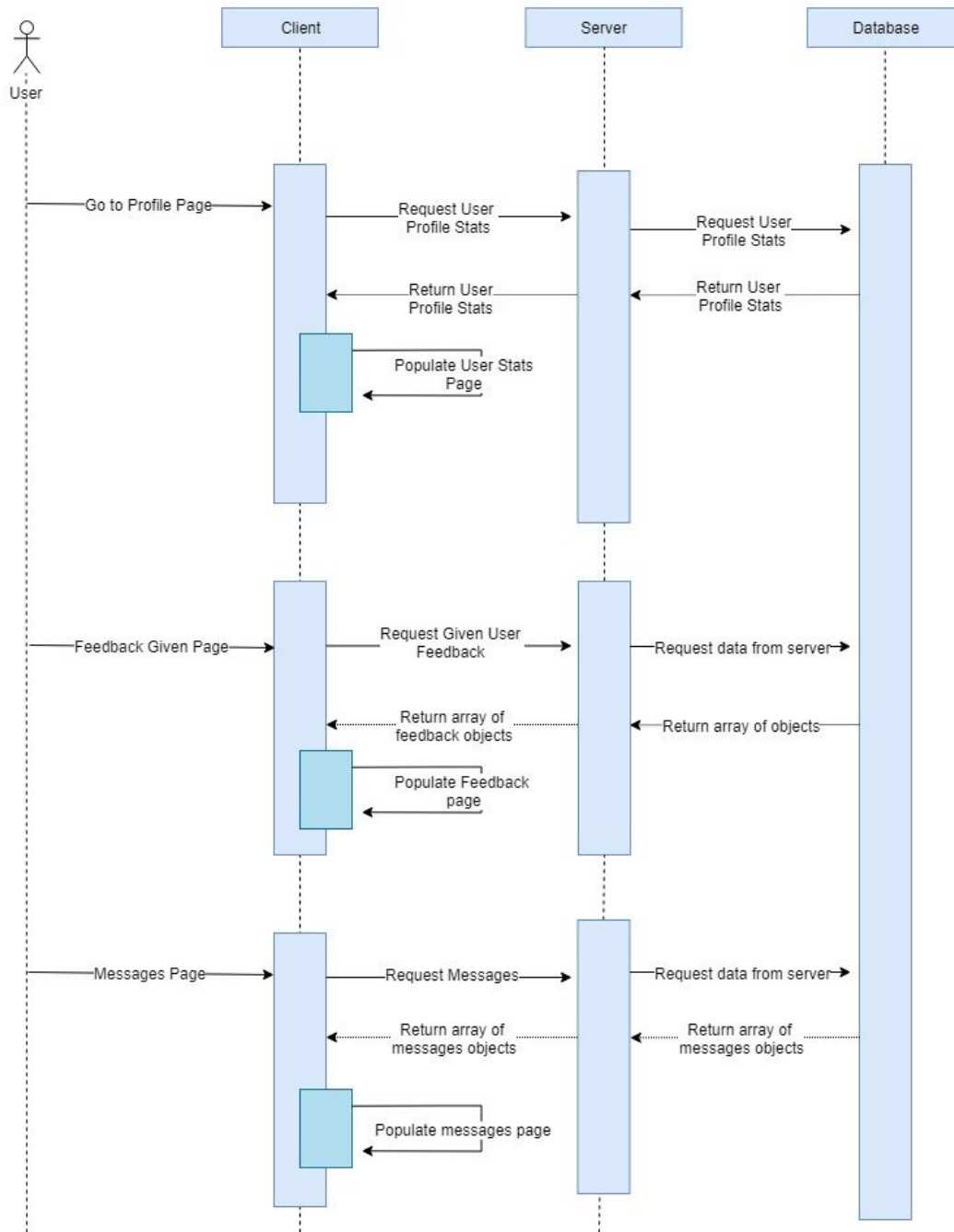
- Sequence of events when the user starts the program



- Sequence of events when user uploads, edits, or deletes a submission



- Sequence of events when a user goes to profile page, gives feedback, or messages other users



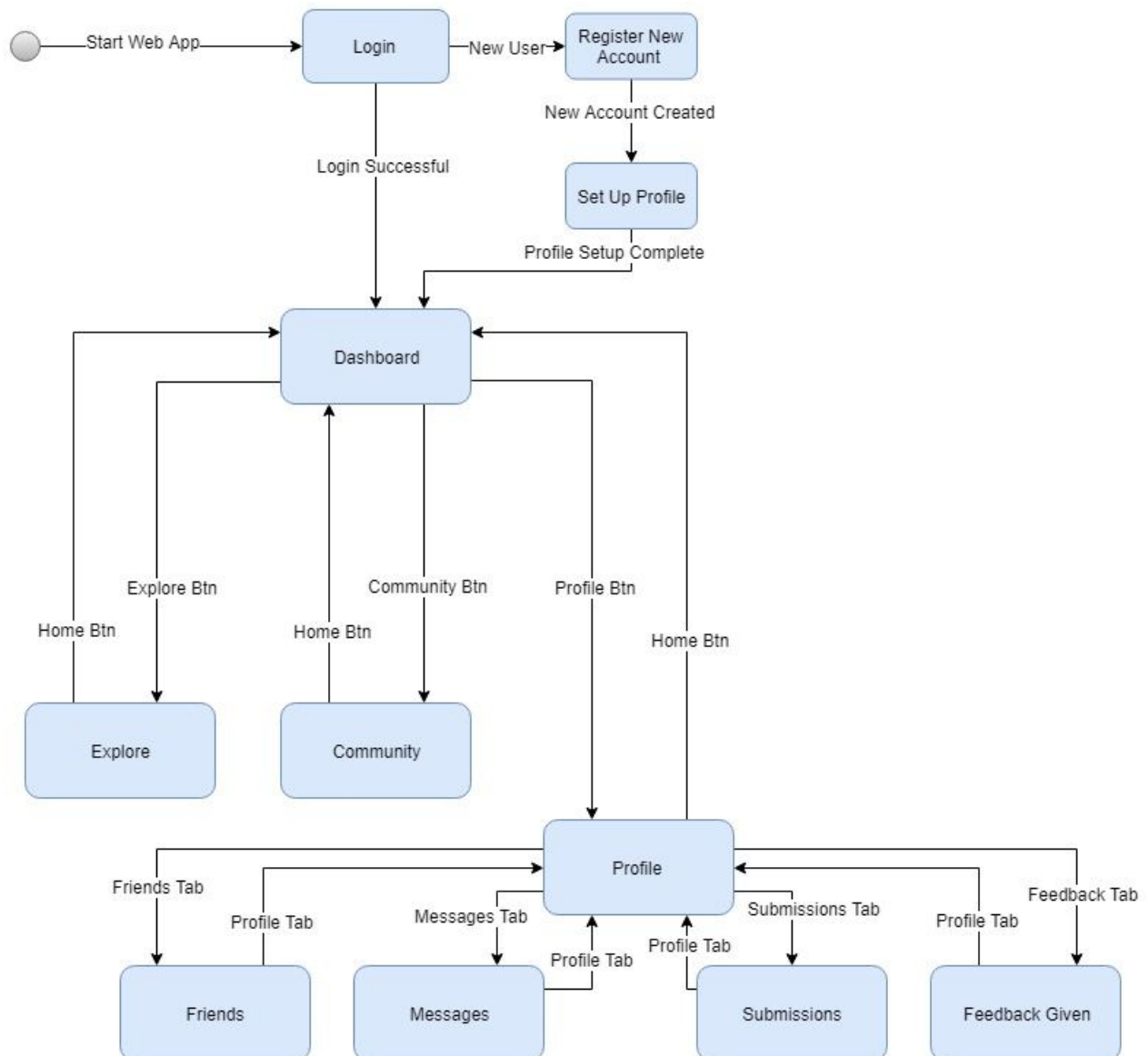


## State Diagrams

### Overview of Critika

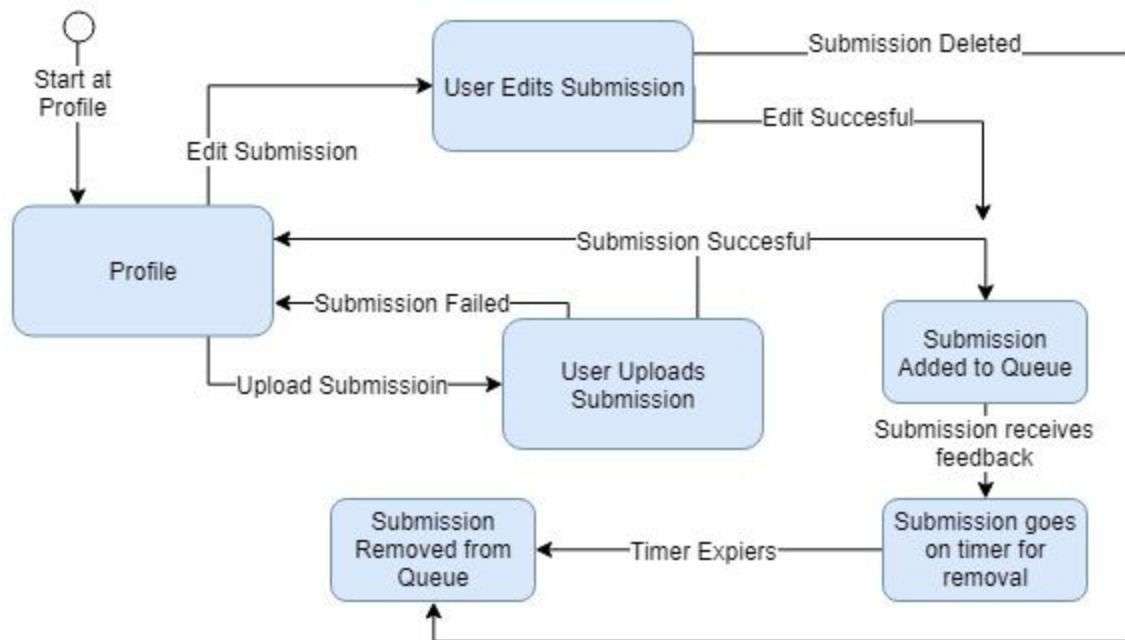
The design of Critika is simple and easy to use. The user is prompted to login using his or her username and password. However, if the user is new and does not have an account, he or she is encouraged to register a new account. Once a user has registered or logged in, the user is taken to the Dashboard page. The Dashboard page allows the user to look at the Explore, Community, and Profile pages. When a user clicks on the Explore button, he or she is taken to the Explore page where he or she can look at public submissions, most popular submissions, and highly critiqued submissions. When the user clicks on the Community button, the user can find different communities or sub groups within Critika. The Profile button takes the user to his or her profile where he or she can navigate to the Submissions, Feedback Given, Friends, and Messages pages. When the user navigates to the Submissions page, the user is able to add a new submission, look at his or her submissions, and see the critiques and comments provided for a specific submission. In the Feedback Given page, the user can scroll through the submissions he or she has critiqued or commented. When the user navigates to the Friends page, the user is able to see the users that he or she had added. Lastly, the user is able to navigate to the Messages page where the user is able to see any messages he or she sent or received from other users.

## Navigational Flow Map



## Submission Flow of Critika

Users will be able to upload new submissions through their personal “Submissions” page. Their submission will be automatically added to a feedback queue. As soon as it is added to the queue, a removal timer will start which will remove that specific submission from the feedback queue after a certain amount of time. Users will also be able to edit the name, description, or any other details about their submissions or delete their submissions through their “Submissions” page.



## UI Mockups

- Login Page



The mockup shows a login page for 'CRITIKA'. The title 'CRITIKA' is centered at the top in a large, dark, sans-serif font. Below it is a light purple rectangular box containing the login form. Inside this box, the label 'Username' is above a white input field. Below that, the label 'Password' is above another white input field. At the bottom of the box, there are two links: 'Have an account?' and 'Don't have an account?'. Below the first link is a white button with the text 'Login'. Below the second link is a white button with the text 'Sign Up'.

# CRITIKA

Username

Password

Have an account?

Login

Don't have an account?

Sign Up

- Sign In Page



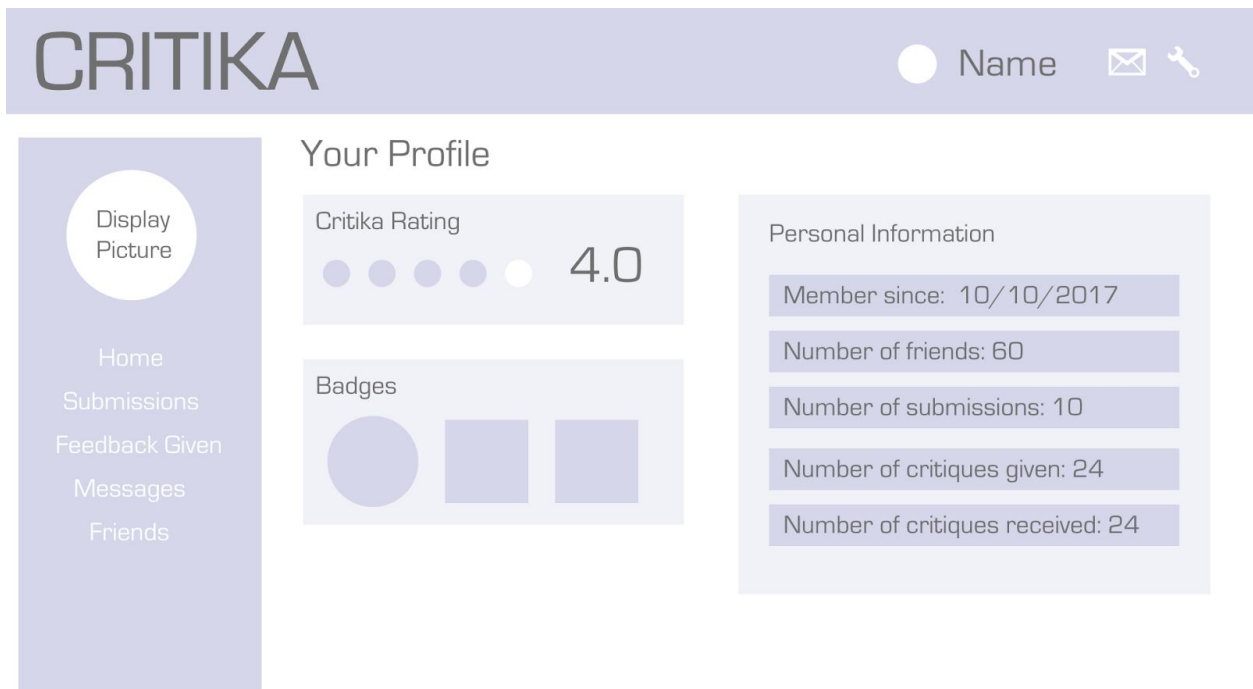
The image shows a sign-up form for a platform named CRITIKA. The form is centered on a light gray background. It features a title 'CRITIKA' in a large, dark blue, sans-serif font. Below the title is a light purple rectangular box containing the form fields. The fields are arranged in a grid: 'First Name' and 'Last Name' are in the first row, 'Username' and 'Password' are in the second row, and 'Email' is in the third row. Each field is a white rectangle with a thin gray border. At the bottom of the purple box is a white rectangular button with the text 'Sign Up' in a dark blue font.

# CRITIKA

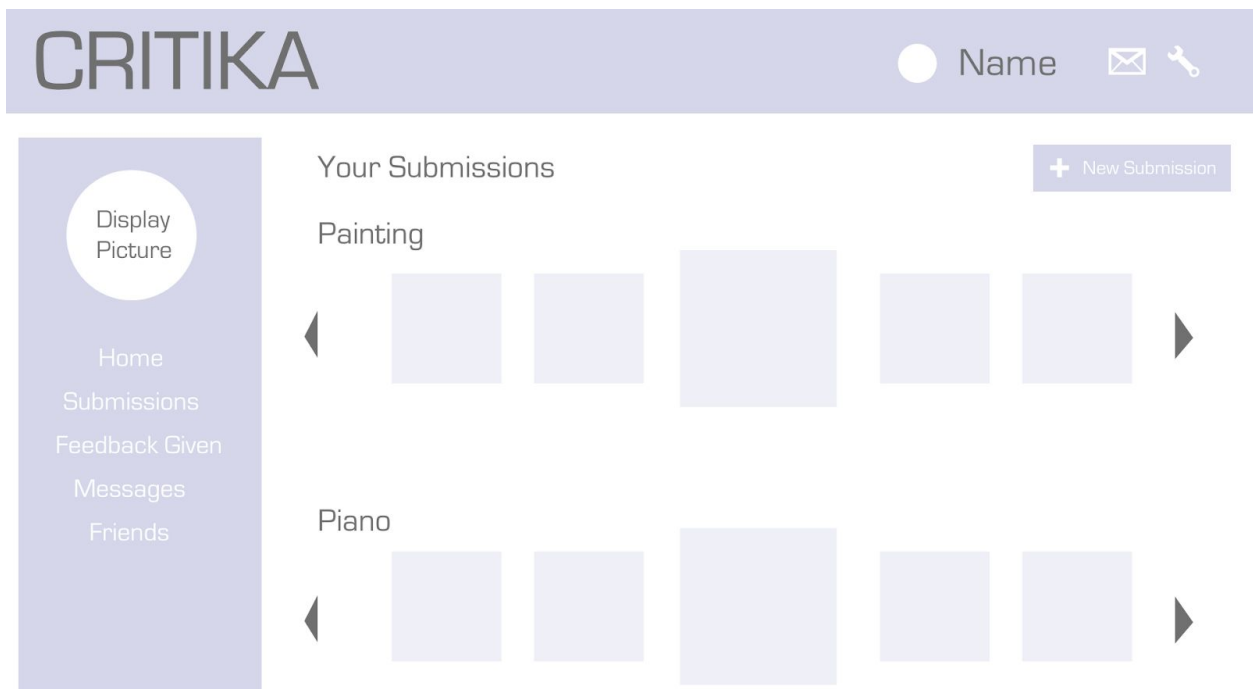
First Name	Last Name
Username	Password
Email	

Sign Up

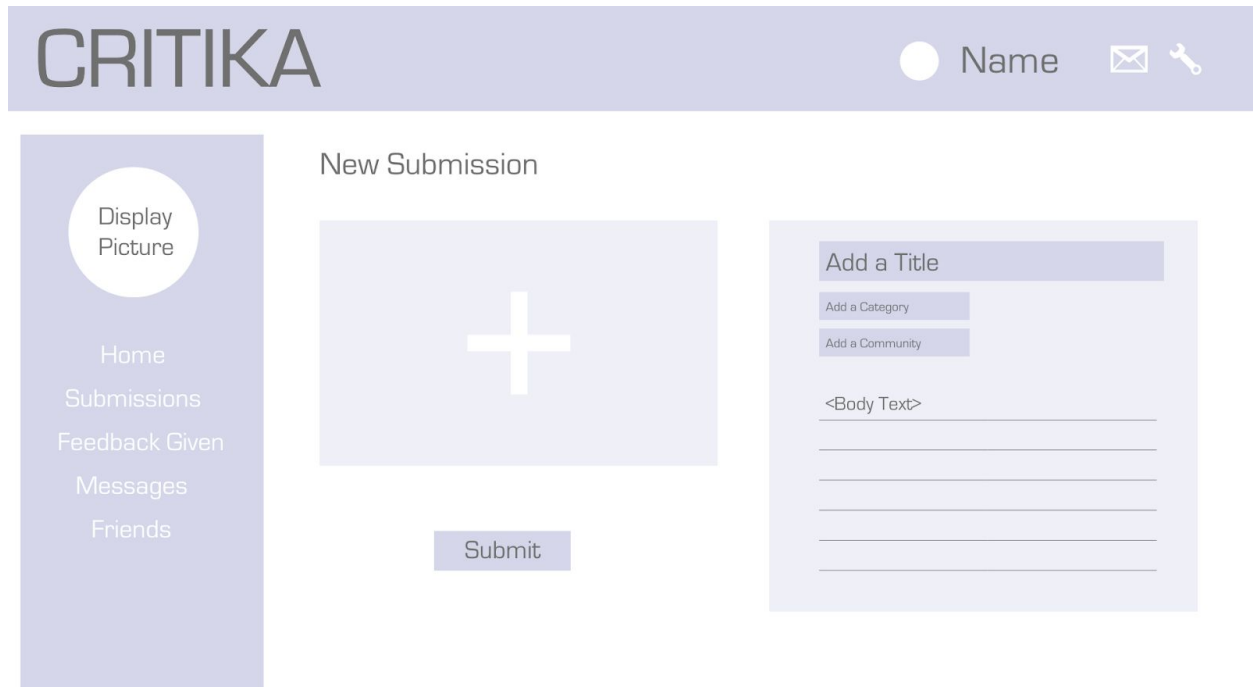
- User profile that shows user's rating, badges, and statistics



- User's past submissions page

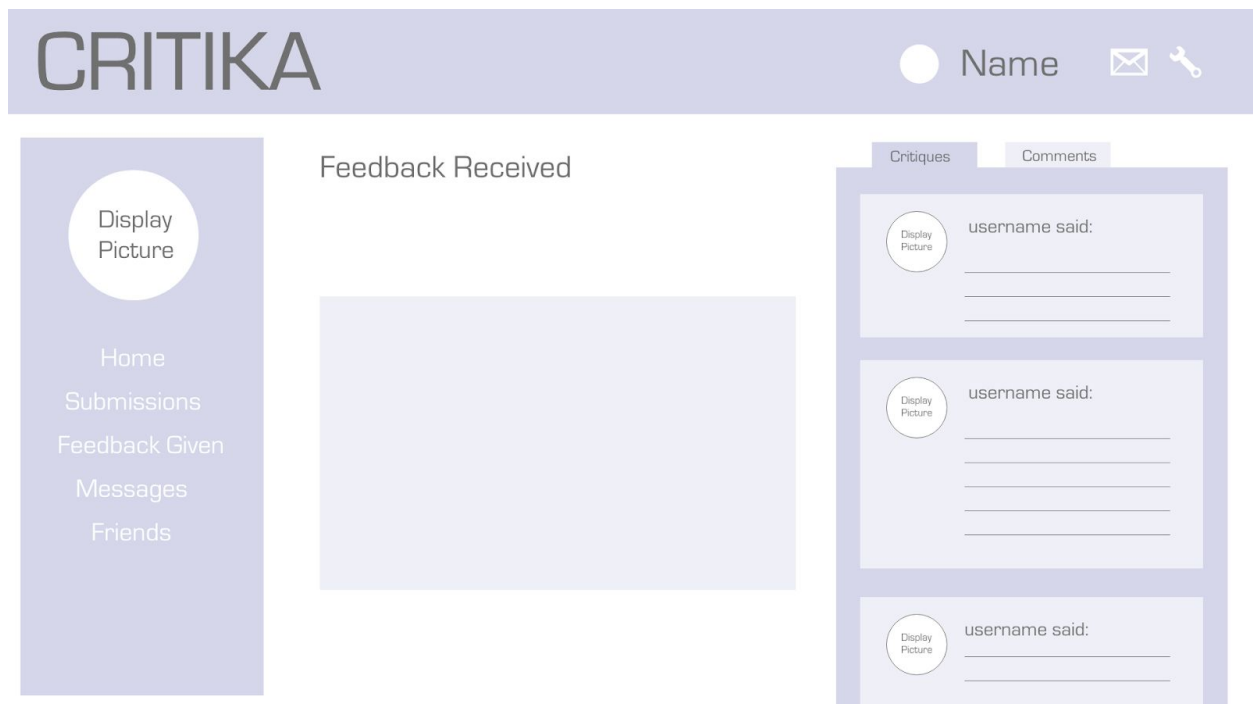


- Page where a user can post a new submission to be critiqued



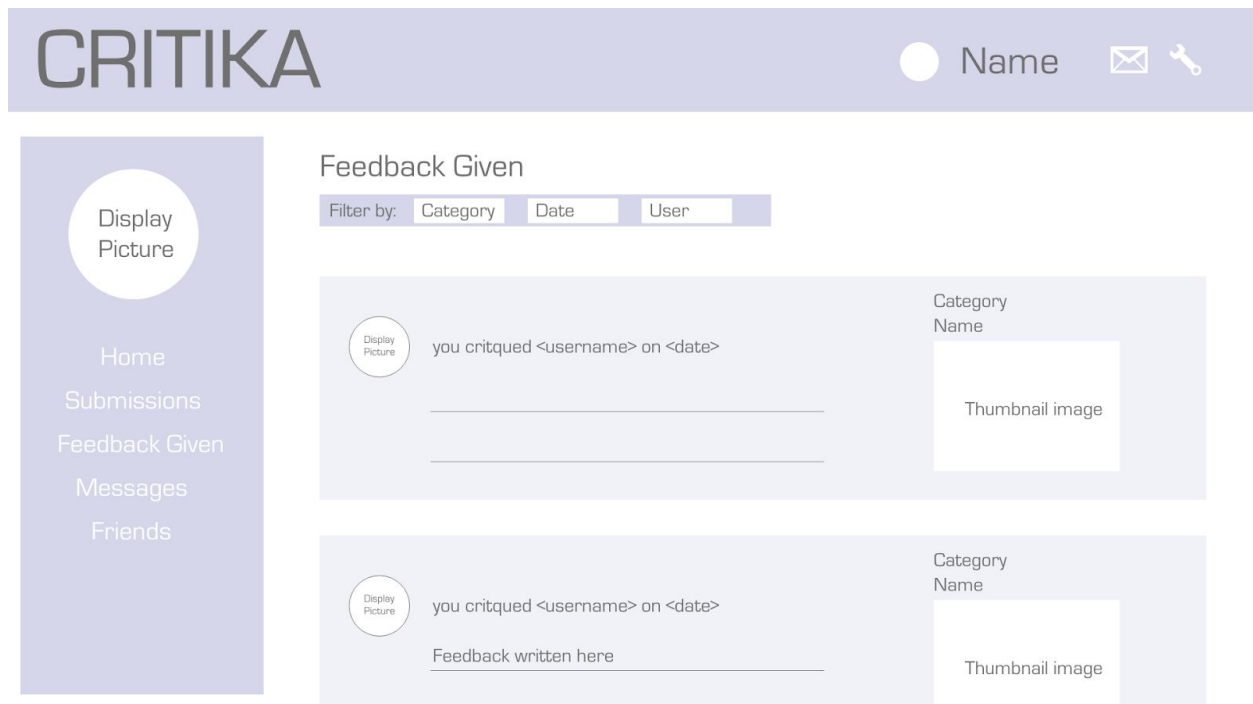
The image shows a web page for 'CRITIKA' with a header bar containing the logo, a user profile icon, the name 'Name', and icons for email and a key. A left sidebar contains a 'Display Picture' button and a list of links: Home, Submissions, Feedback Given, Messages, and Friends. The main content area is titled 'New Submission' and features a large light blue box with a white plus sign. Below this box is a 'Submit' button. To the right of the plus sign box is a form with the following elements: 'Add a Title' (text input), 'Add a Category' (text input), 'Add a Community' (text input), and a text area labeled '<Body Text>' with five horizontal lines for input.

- Critiques and comments given to the user for a specific submission

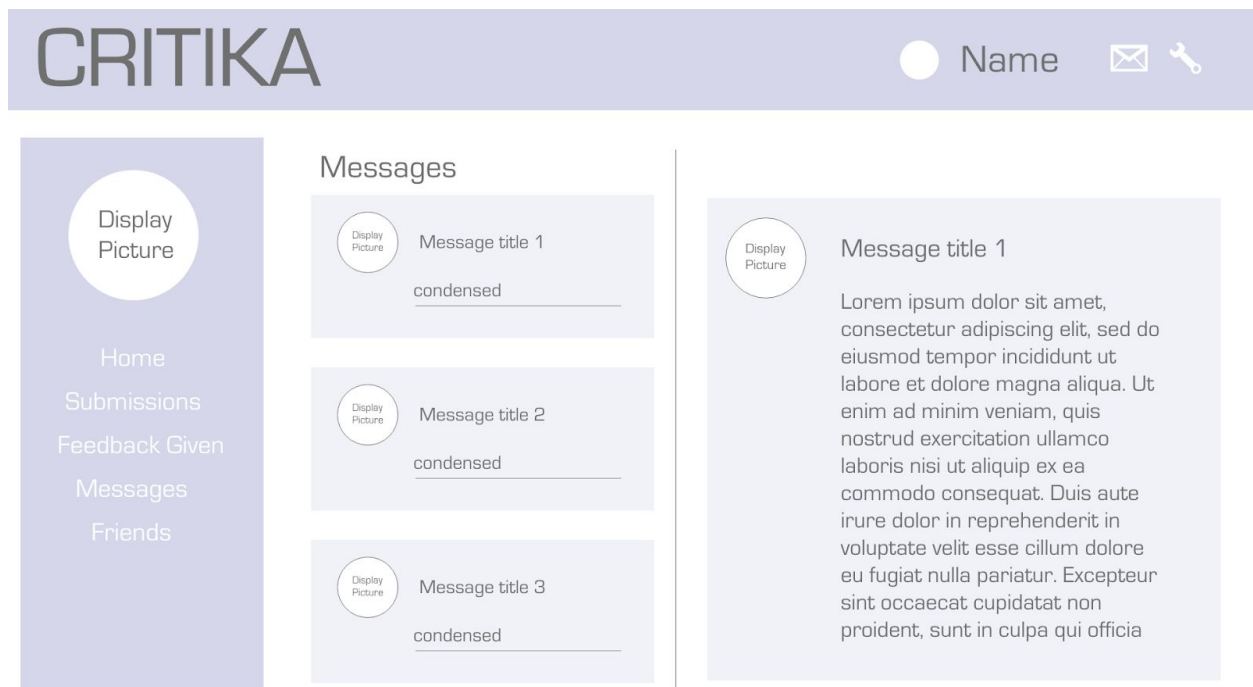


The image shows a web page for 'CRITIKA' with a header bar containing the logo, a user profile icon, the name 'Name', and icons for email and a key. A left sidebar contains a 'Display Picture' button and a list of links: Home, Submissions, Feedback Given, Messages, and Friends. The main content area is titled 'Feedback Received' and features a large light blue box. To the right of this box is a panel with two tabs: 'Critiques' (selected) and 'Comments'. Below the tabs are three identical feedback entries. Each entry consists of a 'Display Picture' button, the text 'username said:', and three horizontal lines for input.

- Page where a user's past critiques will be displayed



- Page displaying a user's messages with others

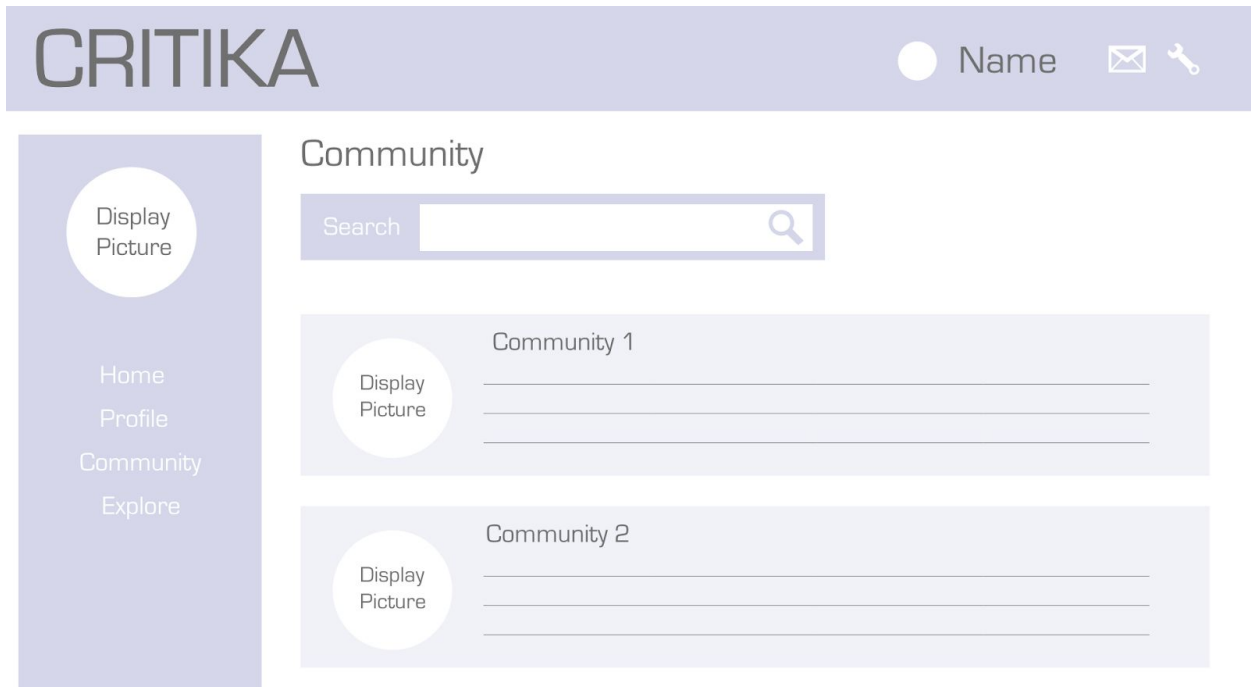




- Home page, showing your notifications and posts to critique



- Community page, where users can search and access communities



- Page that allows user to look at other submissions that are public

