



# Meliora

Good Today, Better Tomorrow

CS 407 - Software Engineering Senior Project

## Design Document

Team 9

*Dithi Saxena* ([saxena52@purdue.edu](mailto:saxena52@purdue.edu))

*Abhishek Gunasekar* ([agunase@purdue.edu](mailto:agunase@purdue.edu))

*Xavier Madera* ([xmadera@purdue.edu](mailto:xmadera@purdue.edu))

*Vidit Shah* ([shah444@purdue.edu](mailto:shah444@purdue.edu))

*Garrett Lee* ([lee2920@purdue.edu](mailto:lee2920@purdue.edu))

## TABLE OF CONTENTS

<b>Purpose</b>	<b>3</b>
• <b>Functional Requirements</b>	<b>4</b>
• <b>Non Functional Requirements</b>	<b>7</b>
<b>Design Outline</b>	<b>9</b>
• <b>Components</b>	<b>9</b>
• <b>High-Level Overview</b>	<b>9</b>
• <b>Activity State Diagram</b>	<b>10</b>
<b>Design Issues</b>	<b>12</b>
• <b>Functional Issues</b>	<b>12</b>
• <b>Non Functional Issues</b>	<b>14</b>
<b>Design Details</b>	<b>17</b>
• <b>Class Diagrams</b>	<b>17</b>
• <b>Description of Classes and Models</b>	<b>18</b>
• <b>Database Design</b>	<b>19</b>
• <b>Description of Database Collections</b>	<b>20</b>
• <b>Sequence Of Events</b>	<b>22</b>
• <b>UI Mockups</b>	<b>28</b>

## Purpose

In the midst of a pandemic where people — old and young alike — are constantly navigating through challenges, mental health has become a major concern. In fact, mental health issues are beginning to deteriorate an individual's value system. Nowadays, in a distressing situation, People are more likely to act in an unethical and unprofessional manner. While several existing products like Vent and Therapy Buddy attempt at improving mental intelligence, they have a major shortcoming. Users cannot share their thoughts and experiences, expecting to hear positive feedback. Mainstream apps like these do not aim to filter out negative behavior, and don't necessarily encourage a positive mindset within the original blogger. In addition, members cannot interact with their communities either through a written method or a visual method.

These are the problems that our app aims to solve. *Meliora* is a web application that helps users share their thoughts and emotions by creating social posts describing their experiences, thereby promoting positivity and good values. Unlike the other competitor apps, *Meliora* provides a follow feature to enables users to follow other members in their community and a reaction feature that allows users to react to posts made by others through positive emojis (hearts, smiley faces, thumbs up, etc.). In addition, as a complementary feature, *Meliora* also displays the most popular posts for a given week in the homepage based on the number of reactions made by users on a given post. To protect the privacy of users, *Meliora* also gives members the ability to post anonymously in the community. More importantly, however, *Meliora* has a negative flag feature and report option that allows members to report any inappropriate posts made by a user. As a frontrunner in this domain, *Meliora* also provides additional features such as motivational quotes and a mental health resources page that serves to inspire and support individuals in their time of need.

*Meliora* seeks to include a broad set of people as its target audience because a positive mindset is a universal goal for many and cannot be limited to one demographic. Any person seeking to improve their emotional intelligence by reading, reacting, and sharing various experiences from their past will benefit from the use of *Meliora*. By reading others' inspiring posts on overcoming specific challenges and commenting on individual posts, *Meliora* aims to promote an optimistic mindset and healthy dialog that provides relief to those struggling with mental health issues in the community either through stress or anxiety. Through its dynamic UI and friendly UX features, *Meliora* will be an accessible app for all.

## Functional Requirements

- Users can register or login in with *Meliora*.

### As a user,

1. I would like to be able to create an account with *Meliora* so that I can access all its features.
2. I would like to reset the password to my account so that I can always have access to *Meliora* and its features.
3. I would like to login to my account using my username/email and password.
4. I would like to receive an email reminder if I have not posted anything within a week of signing up (if time allows).

### As an admin,

1. I would like to display an error message if at least one of the user's credentials already exists in the database when creating an account.
  2. I would like to ensure that only users that are successfully signed up and logged in can access the features of *Meliora*.
  3. I would like to display an error message if at least one of the user's credentials are incorrect.
- Users can easily navigate between pages on the *Meliora* app and easily create a post.

### As a user,

1. I would like to see a navbar containing links on each webpage of the application.
  2. I would like to view a sign up and login button in the landing page.
  3. I would like to view a log out button on every page except the landing page.
  4. I would like to view my profile and profile settings from the header.
  5. I would like to have an option of creating a new post from the header.
  6. I would like to see the logo of the app on the header, clicking on which leads me back to the home page.
  7. I would like to have an option of viewing my recent notifications on the header.
  8. I would like to see all of my recent notifications in the navigation bar.
- Users can easily view and filter posts with categories on the home page.

### As a user,

1. I would like to view the landing page to understand more about the features of *Meliora*.
  2. I would like to see a motivational quote whenever I access the app.
  3. I would like to see a list of dynamic categories that describe the common mental health topics (relationships, grief, stress, etc) tagged by users.
  4. I would like to see a trending posts section that displays the most popular posts of the week based on the total number of reactions.
  5. I would like to follow/unfollow other user profiles in *Meliora* through their profile or I would like to follow/unfollow multiple categories through the homepage.
  6. I would like to see a “from people you follow” section that displays the posts made by people that I follow.
  7. I would like to see a favorite categories section that displays the posts within the categories that I follow.
  8. I would like to be able to show/hide the details of a post like comments, reactions, etc.
  9. I would like to be able to sort posts on the home page or in a category by number of likes or number of comments.
  10. I would like to be able to filter posts based on their location (if time allows).
- Users can customize their posts to be private, create categories, upload images, and more.

#### As a user,

1. I would like to react to other people’s posts with positive emojis (hearts, smiley faces, thumbs up, etc.).
2. I would like to be able to leave comments on others’ posts.
3. I would like to be able to restrict my post to where other users cannot post comments on it.
4. I would like to be able to post anonymously.
5. I would like to be able to delete my posts.
6. I would like to be able to make drafts for my posts (if time allows).
7. I would like to record my location in my post (if time allows).
8. I would like to bookmark posts that I would like to view at a later time.
9. I would like to tag each post with an existing category.
10. I would like to create a category that is relevant to my post if one does not exist already.
11. I would like to share posts on social media applications (if time persists).
12. I would like to be able to see my word count while writing my posts.

13. I would like to be able to hide my posts from the public eye, to where only I can see them.
  14. I would like to upload an image with my post (if time allows).
- Users can easily access information to mental health resources.

As a user,

1. I would like to quickly access helpline information (such as suicide, ptsd, etc.) through the mental health resources page.
- Users can edit their profile and settings information.

As a user,

1. I would like to easily access an editable settings page.
  2. I would like to describe a short bio about myself in the profile page.
  3. I would like to know the number of posts that I have made in total.
  4. I would like to view the total number of reactions that my posts have received.
  5. I would like to share my profile online by generating a public url in the profile page.
  6. I would like to view another user's profile with their public information.
  7. I would like to be able to see my bookmarked posts.
  8. I would like to upload a profile picture for others to see (if time persists).
  9. I would like to change my password once logged in.
  10. I would like the option to sign out of my account.
  11. I would like to view the web app in dark mode.
  12. I would like to be able to delete my account.
  13. I would like to be able to change my notification preferences and settings.
  14. I would like the option to block or unblock another user.
  15. I would like to have the option to make my profile private or public.
- Users can report offensive posts.

As a user,

1. I would like to be able to flag harmful/vulgar/offensive posts.

As an admin,

2. I would like a special backdoor to the app for various maintenance.
3. I would like to be able to see posts that have multiple flags in order to review them.

4. I would like to be able to decide whether multi-flagged posts belong on our app and handle them as such.

## Non Functional Requirements

### Architecture and Performance

Our application would be made robust and efficient so that all of the UI elements load within 100 ms, thereby giving the users a fast and easy user experience. Angular is composed of all the components from an MVC app, and therefore, our app will be implemented in one repository. Netlify, our primary mode of deployment, has the ability to mitigate Distributed Denial of Service (DDoS) attacks, which ensure that our app's performance is not degraded.

### Security

Our backend will include the salting and hashing of various sensitive user information prior to sending it to the database or sending it to the front end. This information may include passwords, date of birth, age, and other such credentials. We also plan to restrict the access of each user so that they may only edit or retrieve their own information.

### Usability

Our web app will be deployed to the internet, making it available for the general public. By using Bootstrap, our web app is dynamically sized, ensuring a reliable user experience regardless of the type of device being used. Our developers will follow industry best-practices when designing the front-end, making *Meliora* one of the easiest and most-user friendly apps that are currently in the market. Navigating through each major section of the application like posting and creating an account will be made effortless by displaying the links on the navbar.

### Scalability

By using Angular/Bootstrap for the frontend, *Meliora's* components can be built to scale automatically. In addition, MongoDB clusters have been developed to scale easily as more entries are added. Since we are using MongoDB as our database, *Meliora* will be able to accommodate several thousands of users at ease. Finally, by using Netlify as our primary deployment tool, the traffic to our web application is load balanced, reducing the latency experienced by end-users.

### Hosting/Deployment

Throughout this semester, we plan to use Git as our version control and GitHub to host our codebase. Upon passing all tests, new versions of our web app will be deployed onto a server. For the purposes of our app, we will be using Netlify for hosting and deployment. This will provide a robust and effective method of hosting our app for our users. To address any issues that arise during development, we plan on using tools such as Postman to help solve such issues.



## Design Outline

*Meliora* is designed so that every time someone logs into the application, they are greeted by only positive influences. With that in mind, we will be using a client-server architecture to design our system, which automatically greets the user with a motivational quote once he or she logs in to *Meliora*. We are using a HTTP server that renders the frontend components to the client and as well communicates to the database.

## Components

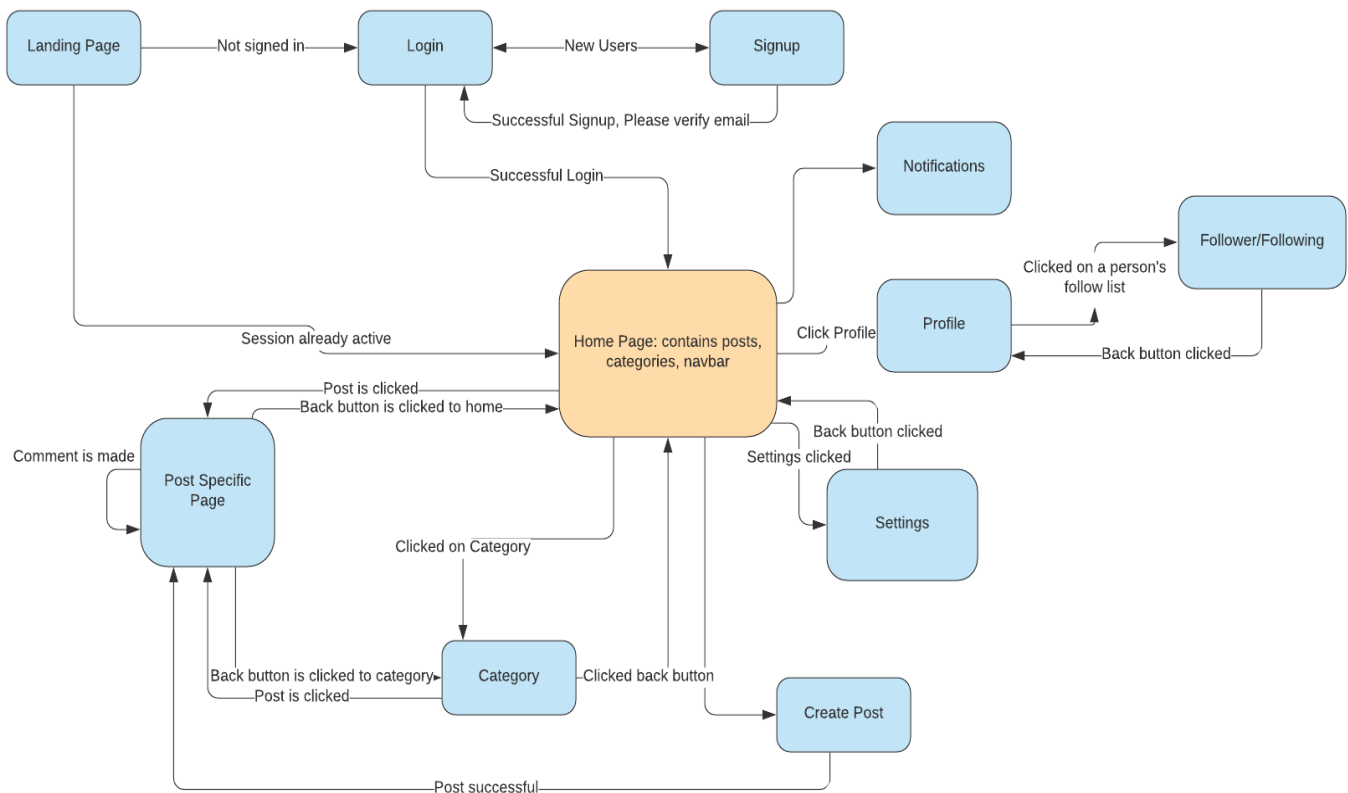
1. Web Client
  - a. The web client will be the web interface for our system and how the user will interact with the system via the internet.
  - b. The web client will display a feed of posts for users to interact with and comment on.
  - c. The web client will also display trending posts based on the number of reactions received from users over multiple communities.
2. Server
  - a. The server will accept HTTP requests from the web client.
  - b. The server will query the database using an HTTP request for information requested from the client.
  - c. The server will return the requested data via JSON format.
3. Database
  - a. A database that will store all the information for the application.
  - b. The database will respond to any queries from the server and send requested data back to the server.
  - c. Typical data stored in the database includes name, username, password, age, sex, etc.

## High-Level Overview

This project will have a many-to-one client server architecture. The client will send requests to the server whenever new posts need to be created and when a user submits a reaction to a post. The server will accept requests from the client and query the database for information. The server will also write data to the database when information needs to be stored. The server will be updating the trending posts section upon each rendering of the web application.



## Activity State Diagram



The user enters our software through the landing page. On the landing page, the user will have an option of logging in to the app. If the user is already logged in, the user will be directly taken to the home page. If it is a first time user, the user will be given an option to go to the sign up page from the login page. If the user realizes that they already have an account, they can also go back to the login page from the sign up page. The user will also be taken back to the login page upon successful sign up. The user will be sent a verification email. Once the user confirms their email, they can login to their account with their credentials.

The home page will have a navbar which will have options to create a new post, view notifications, and access the profile menu, which includes My Profile and the Settings page. The home page will also showcase the timeline where the users can see all the posts from users they follow. These posts will also have the information about the category that the post belongs to. If a user clicks on the category, the user will be taken to the category page where the user can see all the different posts under that category. If the user clicks on the post, the user is taken to the detailed post page, where the user can see detailed information regarding the post including the engagement information such as likes and comments. The user can also add a new comment to the post if comments are enabled for the post. Finally, the profile page will showcase the basic profile information of the user, including the number of followers and the number of people the user is following. The user can click on the counts for each of them to see a list of people whom the user follows or is following.

## Design Issues

### Functional Issues

1. Do users need to login in order to access the features of this application?
  - Option 1: No login required.
  - Option 2: Use Google, Apple, or Facebook to login.
  - Option 3: Allow the user to create a username and password that is unique to our application.

#### **Decision: Option 3**

Justification: Setting up an account should be quick and simple for any user. While using third-party services such as Google, Apple, and Facebook may expedite that process, it is hard to predict what login/password issues can arise in the future. In creating a unique username and password with our service, users' information will be stored directly in our database where we can track all of their information. Lastly, creating a username will also allow our service to have a unique ID to identify customers.

2. How will trending posts be determined?
  - Option 1: Based on the number of reactions to a post.
  - Option 2: Based on the number of views of a post.
  - Option 3: Based on the number of tags on the post.

#### **Decision: Option 1**

Justification: Posts on *Meliora* that should be shared with the broader public should be ones that spread positivity and encouragement. The number of reactions (emojis) will most likely accurately do this. The number of views on a post does not necessarily equate to a post having good content. Tags are so diverse and numerous that they can also be hard to use to determine popular/good posts to share the bigger communities. With emoji reactions (such as hearts, thumbs up, smiley faces, etc.), users can positively interact with posts they like, and these reacted posts can be shared.

3. How will users be able to report harmful/vulgar/offensive posts to keep?
  - Option 1: Users can file a report on a post and describe why it violates *Meliora's* main goal.
  - Option 2: Users can flag harmful/vulgar/offensive posts.
  - Option 3: Users can comment on a user's post and tag *Meliora's* maintenance ID.

### **Decision: Option 2**

Justification: It is important that *Meliora* stays as a positive resort for people to escape to. In order to maintain this environment, we will need the help of users. The best way to professionally report these posts is to allow users to flag harmful/vulgar/offensive posts. This way when the *Meliora* maintenance team sees that a particular post has multiple flags, it can be reviewed and ultimately removed or kept depending on the situation of the post. Option 1 can be tedious for users to have to fill out, and option 3 can create unwanted social situations between users.

4. How long should posts for a user be kept on their profile before being archived?
  - Option 1: Forever, don't move posts to an archive folder
  - Option 2: For 2 years
  - Option 3: For 6 months

### **Decision: Option 2**

Justification: Users should be given an option to archive their old posts. This will save storage space and will avoid overloading of our database. This would also be convenient for users as after two years a user can have up to hundreds of posts cluttering their profile page. To prevent the application from overworking and to provide a user a seamless posting experience on *Meliora*, it will be best to archive posts older than 2 years.

5. How will a user's information be kept private when looking for other users?
  - Option 1: Display all user information (except personal information such as DOB, password, age, address, etc.)
  - Option 2: Allow a user to make certain information in their profile public and other information private
  - Option 3: Allow a user to set their profile to public or private

### **Decision: Option 3**

Justification: Users should be able to determine whether they want their information to be shared or not. Option 3 will allow users to protect their information from other users they are not following. Option 2 can become overwhelming for the user as they will have to select whether each field on the profile page needs public or private information. This could also cause data issues in the database. To avoid this confusion for the user, it would be best to allow the user to make their profile public or private.

## Non-Functional Issues

1. What frontend framework should our team use to develop the web application?

- Option 1: Vanilla HTML, CSS, and Javascript
- Option 2: React
- Option 3: AngularJS
- Option 4: Ember.js

### **Decision: Option 3**

Justification: First of all, using a framework would help the reusability of our code and save time during the development process. Hence, using option 1 would not be a feasible approach to follow. Given that a framework must be used, AngularJS was undoubtedly the best framework because AngularJS is a popular and widely used framework in industry due to its ability to quickly scale based on usage. Moreover, Vidit Shah, one of the team members, has prior experience in developing web applications using AngularJS, and since AngularJS can also easily integrate Bootstrap — the expertise of Abhishek Gunasekar — we chose AngularJS as the best performing and easy-to-use framework for *Meliora*.

2. What backend framework should our team use to develop the web application?

- Option 1: Django
- Option 2: Express.js
- Option 3: Ruby on Rails
- Option 4: Laravel

### **Decision: Option 2**

Justification: Express.js, which is implemented in Node.js, facilitates communication between clients and the backend using JavaScript. This works well because we chose MongoDB as our database and Node.js has features that allow the backend to easily connect to the MongoDB cluster. In addition, since we are using AngularJS as our frontend framework, JavaScript can be utilized to power both the frontend and the backend, thereby creating a standardized programming environment. Unlike the other options like Laravel which are quite outdated, Express.js is widely used in industry and has extensive documentation that our team can use to debug the backend of *Meliora*. Therefore, Express.js is the most appropriate choice to develop the backend of *Meliora*.

3. Which hosting method should our team use to store the contents of the web application?
- Option 1: Local Hard Drive
  - Option 2: GitHub
  - Option 3: Bitbucket
  - Option 4: Cloud providers (AWS, Azure, GCP, etc.)

**Decision: Option 2**

Justification: Based on the system design for our project, GitHub would be the best method to host our application contents. GitHub allows our developers to collaborate easily through its friendly user interface online. Unlike using a cloud provider like AWS and Bitbucket, which has a steep learning curve, GitHub is familiar to all of our engineers since we have all used it in the past for CS 307 and for personal projects. Since a local harddrive is not fault tolerant and scalable to increasing storage, that would not work well with the intended design of *Meliora*. Hence, GitHub will be the best hosting tool to use for *Meliora*.

4. Which deployment tool should our team use to launch our web application?
- Option 1: Netlify
  - Option 2: GitHub Pages
  - Option 3: GitLab
  - Option 4: Firebase

**Decision: Option 1**

Justification: Among the four options listed above, Netlify will be the right choice to deploy the *Meliora* app. Although GitHub pages and Firebase are relatively straightforward to use, they are primarily meant for static websites and not for dynamic web applications like *Meliora*. Contrary to the previous two options, GitLab has all the features required to deploy *Meliora*. However, it also has an extensive suite of other CI/CD tools, which is more than what is required for this project. Since Abhishek Gunasekar and Xavier Madera (our team members) have previously used Netlify to deploy their web applications in the past, Netlify seems to be the wise choice to deploy our application.

5. What development environment or tool should our team use?
- Option 1: Vim
  - Option 2: Sublime Text 3
  - Option 3: Eclipse EE

- Option 4: Visual Studio

**Decision: Option 4**

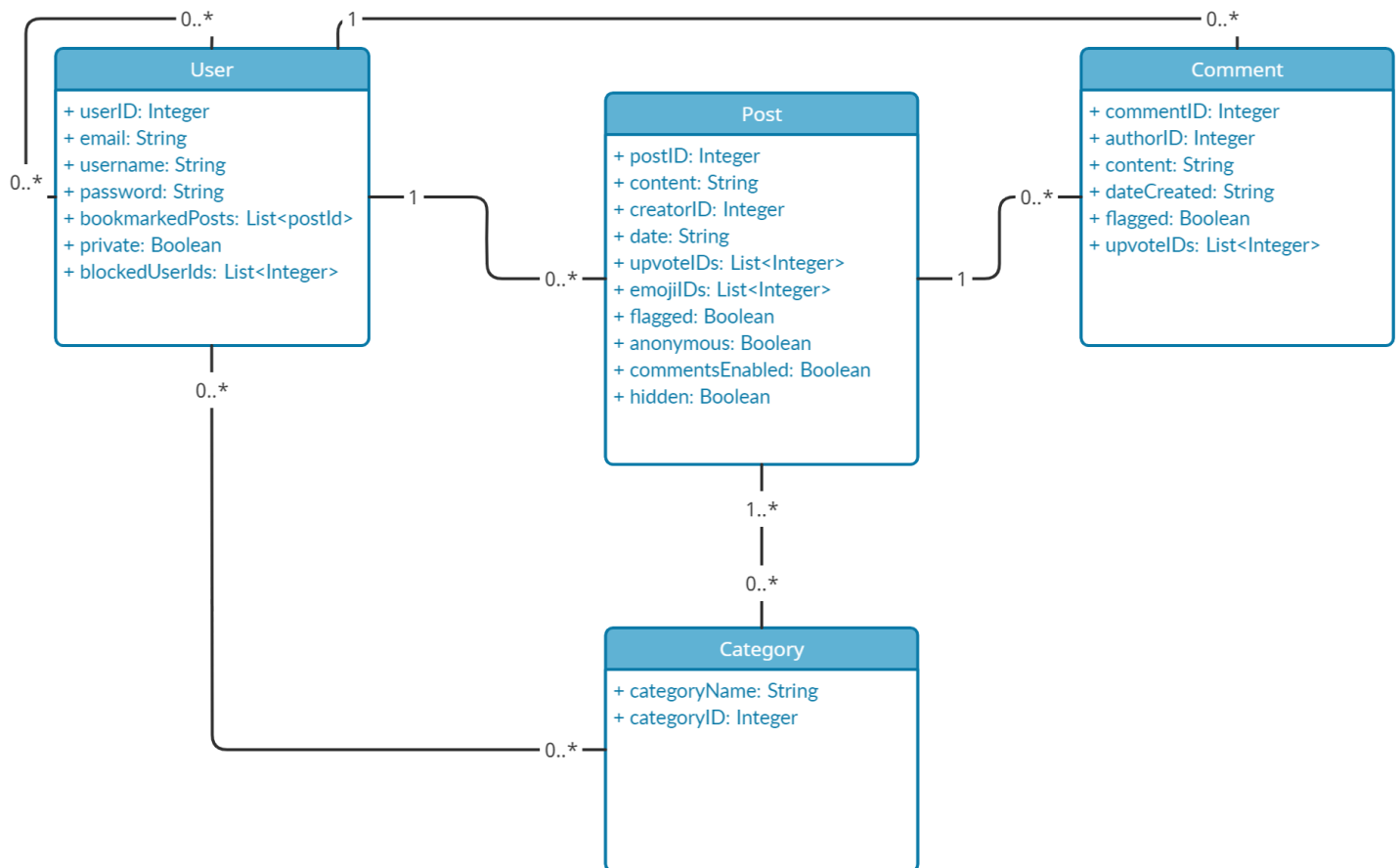
Justification: Our team has worked with all of these development tools in some capacity. Our project has multiple components to it (e.g. Express.js, clients, backend), which would not make Vim or Sublime Text 3 a suitable choice due to the difficulty in managing these multiple parts at the same time and using the terminal to push changes to GitHub. Since the project isn't at the level of an Enterprise Application, Eclipse EE would not be suitable due to the long download and installation time and also the various workstation and directory setups (not needed due to the small size of our project). This leaves Visual Studio Code, which is lightweight, has built in JavaScript support, and has extensions for using other languages and technologies, which makes this our final choice.



## Design Detail

Our App will have four main components. The Users, the topics on which our users will choose to write about, known as categories, the posts they create, and the feedback and replies of other users. All of these will be arranged in a clever fashion within the database and in an elegant fashion via the user interface.

### Data Classes Diagram



This diagram provides the database relationships between classes found throughout the application. Using MongoDB as our database, we will be able to quickly reference each model and each of their relationships to return needed information to the client.

## Description of Classes and Models

- **User**
  - A user is created when someone signs up for our application.
  - A user will have a unique username and password combination to log in with.
  - The user can complete their profile by filling in their age, gender, birthday, email, and phone number.
- **Post**
  - A post is a block of text, a title, an author, a location, and a timestamp.
  - A post can have an anonymous toggle that prevents users from seeing the author and location.
  - A post can also gain comments, likes, hugs, etc.
- **Category**
  - A category has a title and a list of post ID's that are associated with it.
  - Examples of categories in this application would be titled with topics such as Relationships, Work, Stress, Grief, etc.
- **Comments**
  - Will have a Post ID indicating which post this comment is under, an author, a timestamp, and body text.
  - These are blocks of text made by other users under another user's post as a form of feedback or reply to the original Post.

## Database Design

NoSQL database using MongoDB

User	Type
userID	Integer
email	String
username	String
password	String
bookmarkedPosts	List<postId>
private	Boolean
blockedUserIds	List<Integer>

Post	Type
postId	Integer
content	String
creatorID	Integer
date	String
upvoteIDs	List<Integer>
emojiIDs	List<Integer>
flagged	Boolean
anonymous	Boolean
commentsEnabled	Boolean

hidden	Boolean
--------	---------

Comment	Type
commentID	Integer
authorID	Integer
content	String
dateCreated	String
flagged	Boolean
upvoteIDs	List<Integer>

Category	Type
categoryID	Integer
categoryName	String

## Description of Database Collections:

The collections showcased above depict the structure of our MongoDB NoSQL database. Here is a description of each of the collections mentioned above:

### User:

- **userID:** This is a unique ID that represents each user.
- **email:** This is the email that the user has signed up with.
- **username:** This is the unique username that the user sets at the time of sign up.
- **password:** This is the secure password that the user sets at the time of sign up to protect the account.
- **bookmarkedPosts:** These are the set of posts that the user likes and has bookmarked them to be viewed at a later time.

- **private:** This is a boolean that describes whether the user wants the profile to be publicly accessible or not.
- **blockedUserIDs:** These are the set of users that the user has blocked.

#### Post:

- **postId:** The unique ID that represents each post.
- **content:** The content of a post that the user makes.
- **creatorID:** The userID of the person who made the post.
- **date:** The date on which the post was made.
- **upvoteIDs:** The userIDs of the users who upvoted the post.
- **emojiIDs:** The userIDs of the users who reacted with an emoji on the post.
- **flagged:** A boolean representing whether the post is flagged or not.
- **anonymous:** A boolean representing whether the user is anonymous or not to other users.
- **commentsEnabled:** A boolean representing whether the creator wishes for people to comment on the post.
- **hidden:** A boolean representing whether the post is hidden from public view or not.

#### Comment:

- **commentID:** The unique ID that represents a comment
- **authorID:** The userID of the creator that made the comment.
- **content:** The text of the comment posted by the user.
- **dateCreated:** The date on which the comment was posted.
- **flagged:** A boolean representing whether the comment has been flagged or not.
- **upvoteIDs:** The userIDs of users who have upvoted the comment.

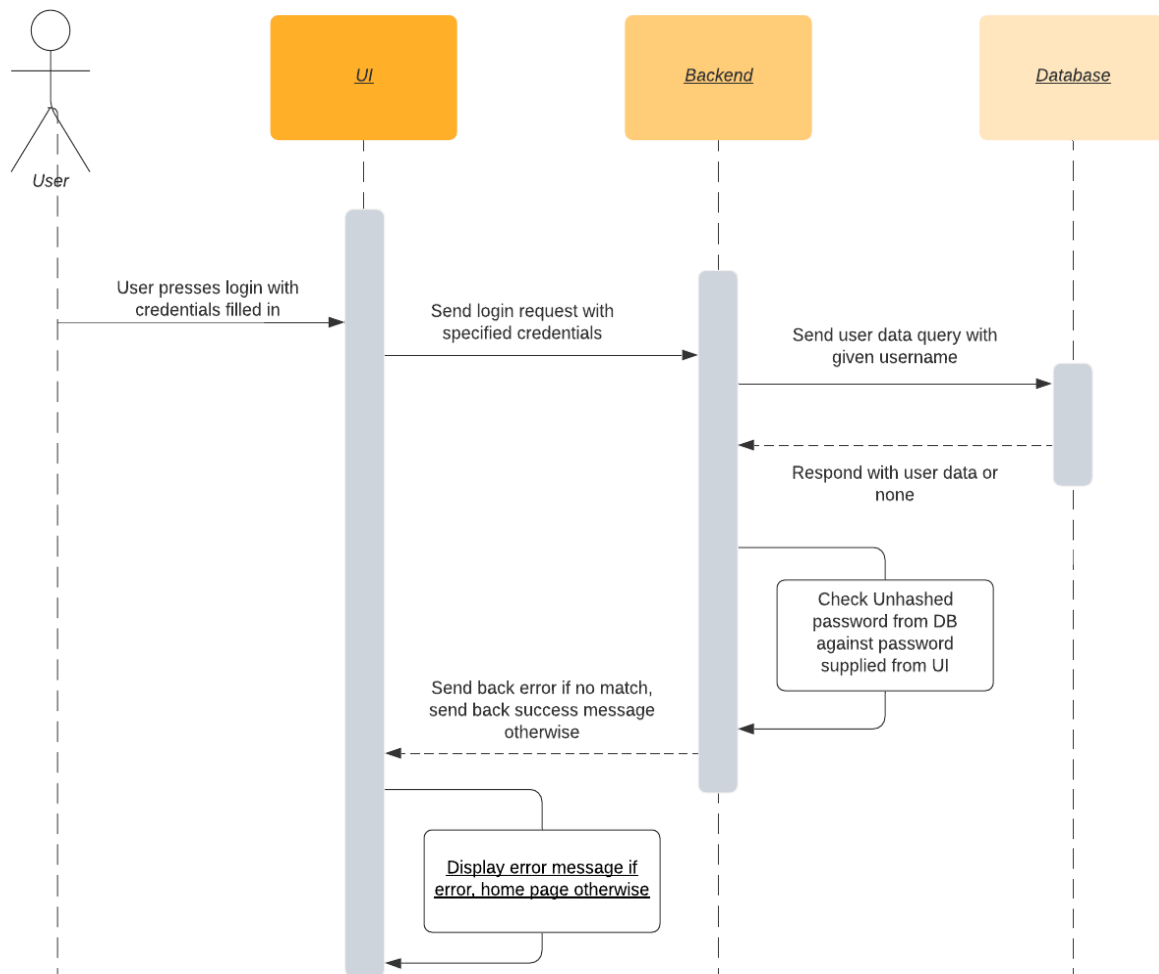
#### Category:

- **categoryID:** The unique ID representing each category.
- **categoryName:** The name of the category.

## Sequence of Event Diagrams

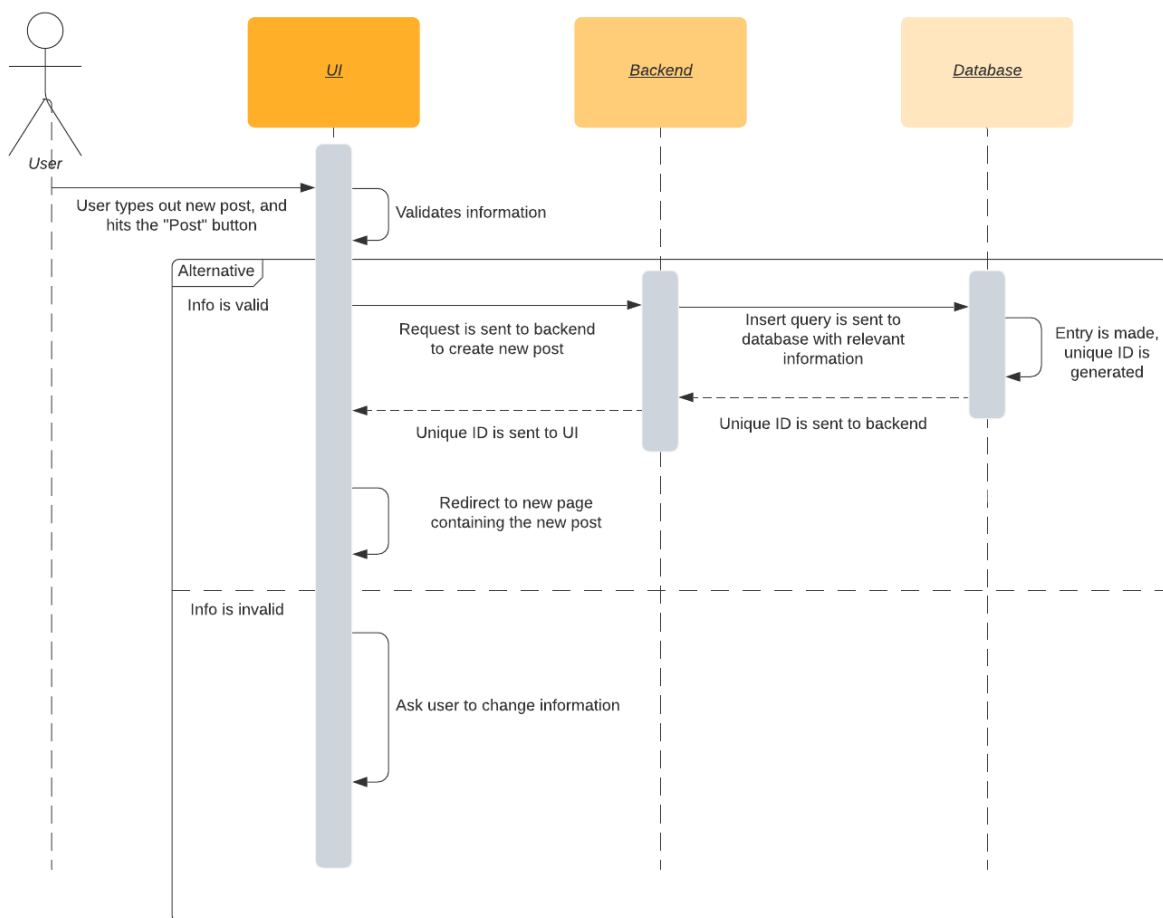
1. Sequence of events when a user logs in to *Meliora*.

When a user enters their username and password in and presses login, a request is sent to the backend, which then queries the database for user info given a username. The database responds with relevant user data, or nothing. If given user data, the backend unhashes the password from the database and checks it against the password provided by the UI. If it matches, a success message/status is sent to the UI, prompting it to redirect to the homepage. If there is any error on the backend, the UI displays an error message asking the user to try again.



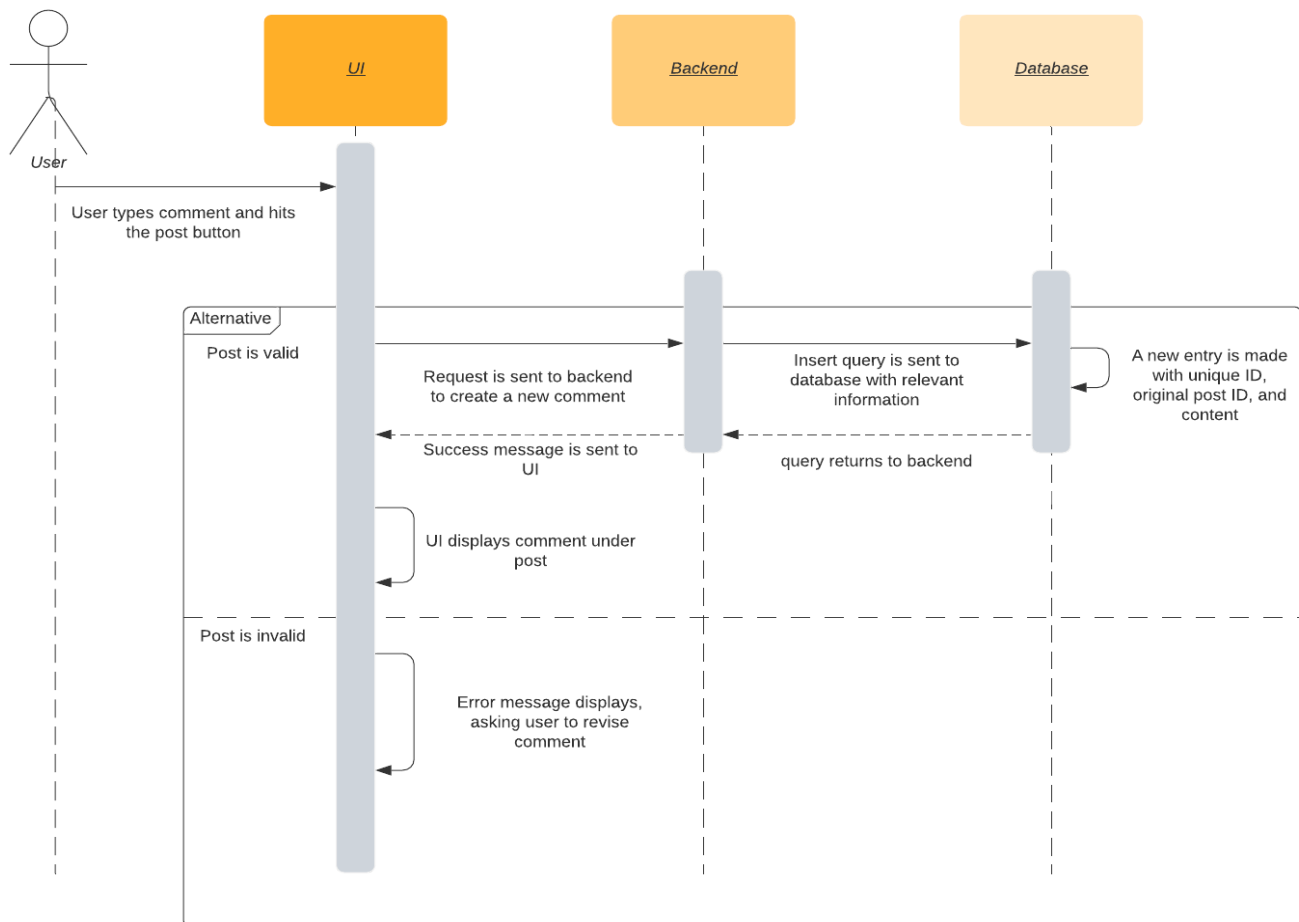
## 2. A sequence of events when a user creates a new post

When a user writes up a new post and clicks the post button, the front end code will validate the information given. If it is invalid, an error message will display asking the user to revise the post. If the info given is valid, a request will be made to the backend to create a new post. The backend will then issue an insert query to the database with all of the relevant information to the post. The database creates a new entry with a unique identifier that is sent to the database. The backend sends a success status and the unique ID to the UI. The UI then redirects to the new page containing the new post.



### 3. A sequence of events when a user leaves a comment on another user's post

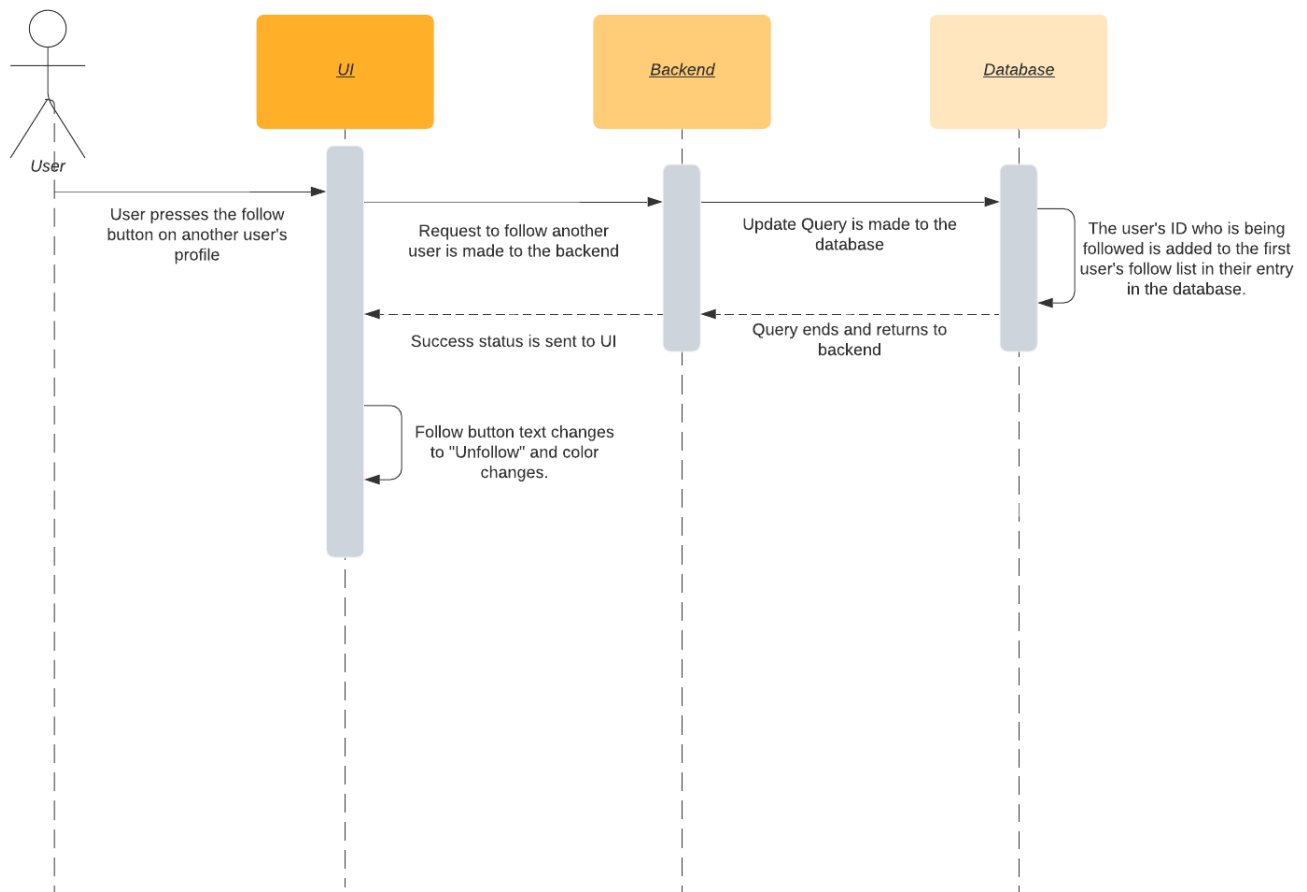
When a user types up a comment to leave on some other user's post, the UI first validates the information given. If it is invalid, the UI will display a message asking the user to revise the comment. If it is valid, a request is made to the backend with the original post's ID, the author, and the content. An insert query is triggered to add a new entry to the database with the given parameters. A unique ID is created for the comment, with the entry containing the original post's ID, the author, a timestamp, and the content.





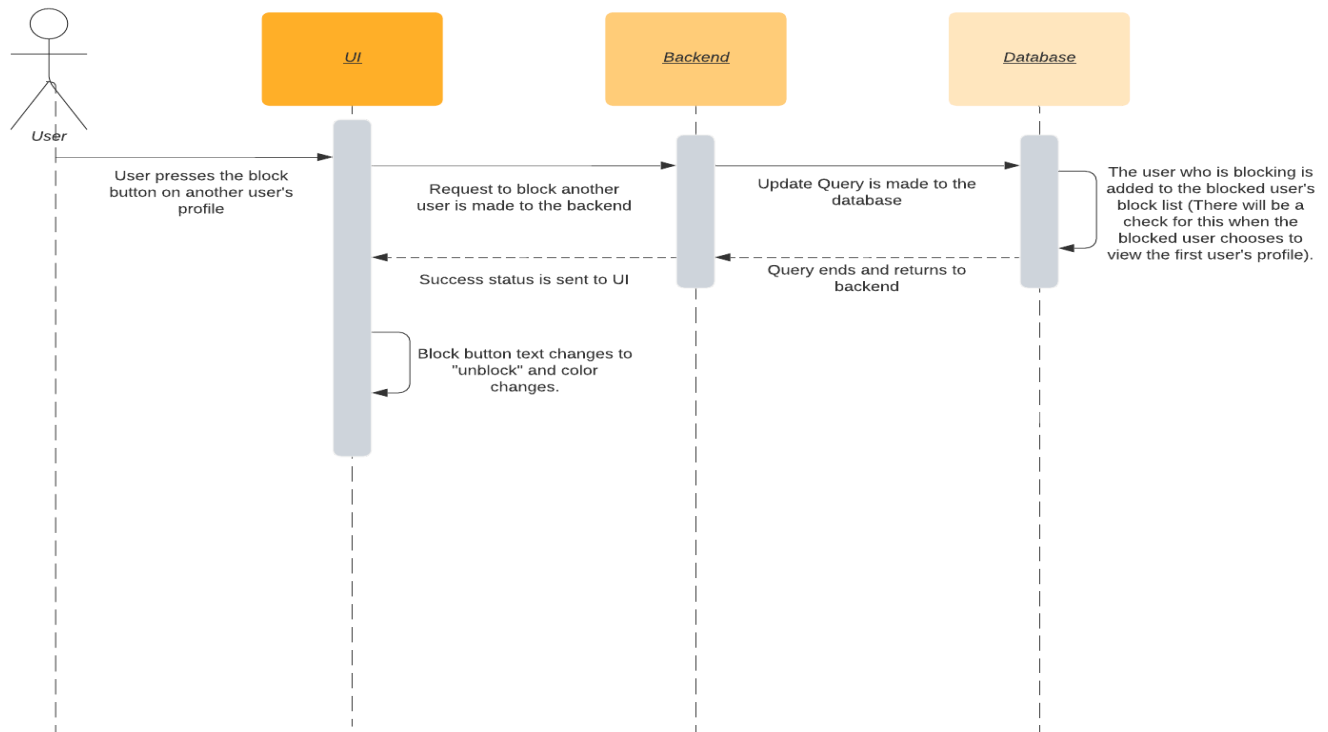
#### 4. A sequence of events when a user follows another user

When the first user presses the follow button on the second user's profile, a request is issued to the backend with both of these users' ID's. The backend runs an update query on the database. Mongo adds the second user's ID to the first user's follow list in their entry in the database. The database returns to the backend, which sends a success status to the UI. The UI changes the color of the follow button and changes the text to "Unfollow."



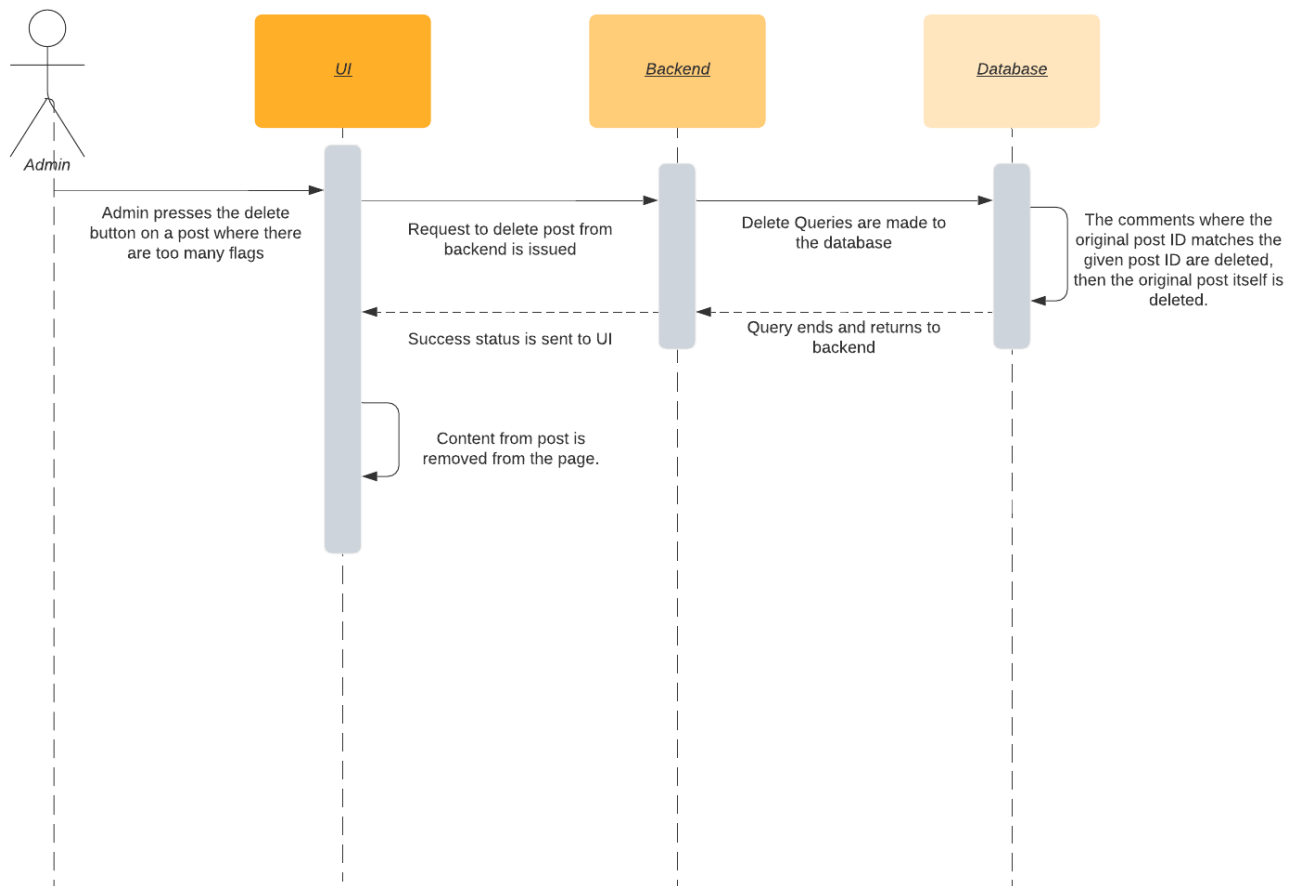
5. A sequence of events when a user blocks another user

When the first user presses the block button on another user's profile, a request is made to the backend with the two user's IDs. The backend then triggers an update query to the database. The first user's ID is added to the second user's block list. This way, the backend will check for this when the second user goes to look at the first user's profile. The query returns to the backend and then the backend sends a success status to the UI. The UI changes the block button text to "unblock" and changes the color.



6. A sequence of events when an admin deletes a post with too many flags

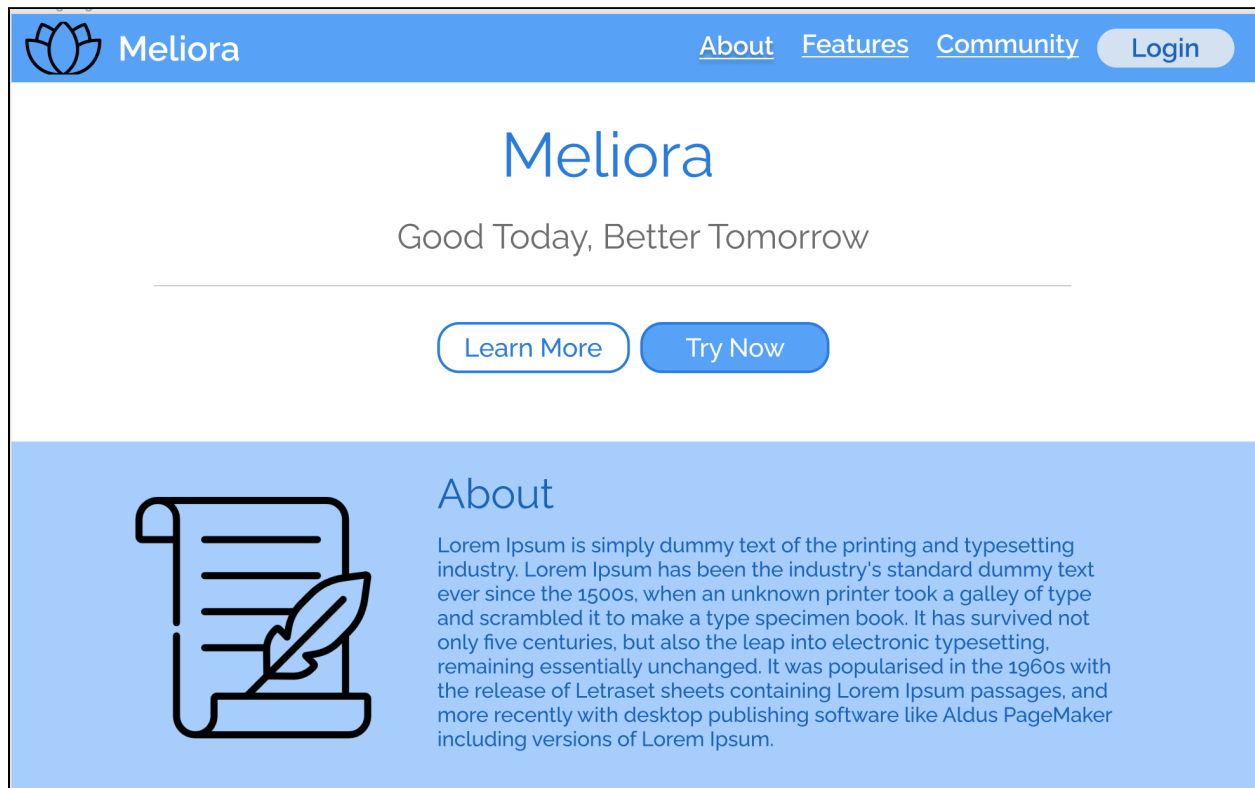
When an admin decides to delete a post with too many flags as maintenance, a request is made to the backend to delete this post and its comments. The backend then issues two delete queries to the database. One to delete every comment where the original post ID matches the ID of the one that the admin intended to delete. Then, the original post itself is deleted. The queries return to the backend, which then sends a success status to the UI. The UI then removes the post from the admin's backdoor page.



## UI Mockups

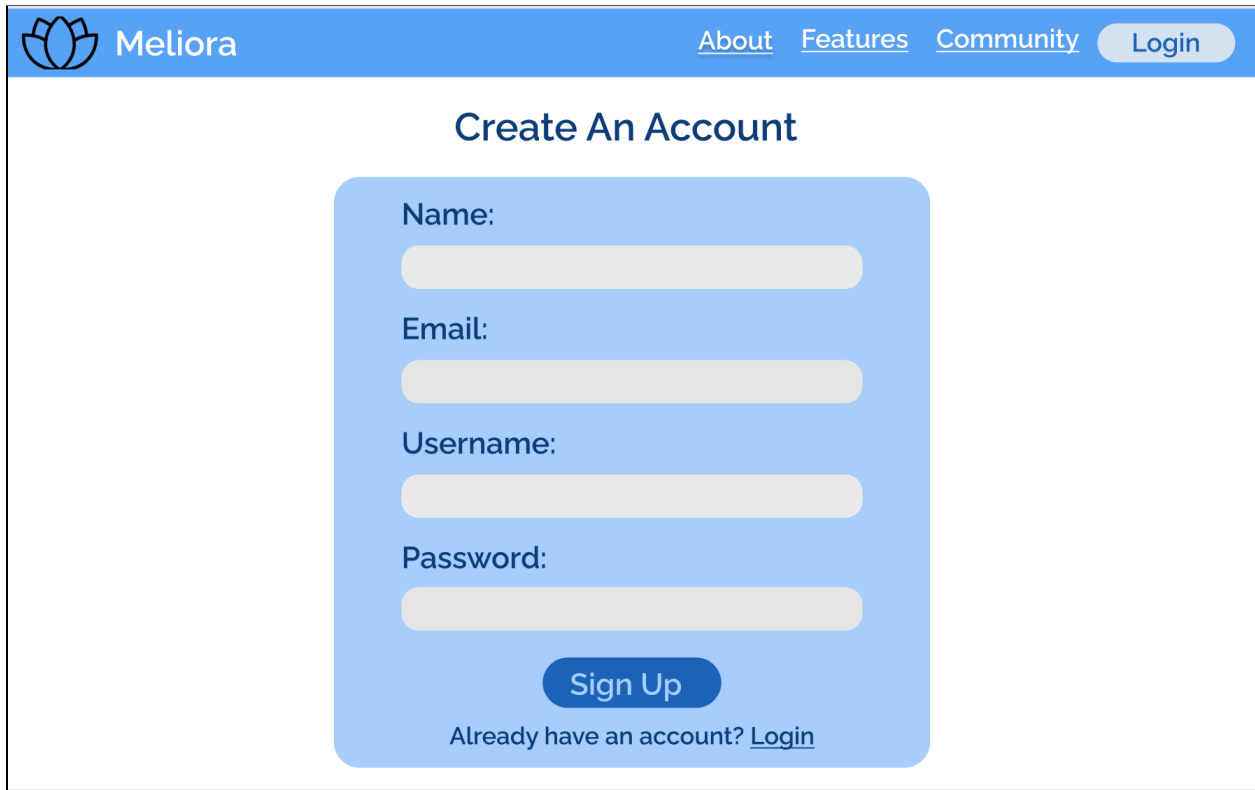
Our product design includes a navbar that is present in all the webpages. This allows the user(s) to toggle between important features of the application such as creating a new post or reading other inspirational posts in the homepage. The navbar also provides easy access to login or sign up for *Meliora*. The logo present at the top left of the navbar can be used to navigate to the homepage of *Meliora*.

### 1 - Landing Page



This is the landing page of *Meliora*. This is where the user(s) can learn more about the features of our application and decide to create an account or login.

## 2 - Sign Up Page



The image shows a web page for 'Meliora' with a blue header. The header contains the Meliora logo (a stylized flower) and the name 'Meliora' on the left, and navigation links 'About', 'Features', 'Community', and a 'Login' button on the right. The main content area is white and features a 'Create An Account' heading. Below this heading is a light blue rounded rectangle containing a sign-up form. The form has four input fields labeled 'Name:', 'Email:', 'Username:', and 'Password:'. At the bottom of the form is a dark blue 'Sign Up' button. Below the button is a link that says 'Already have an account? [Login](#)'.

Meliora

[About](#) [Features](#) [Community](#) [Login](#)

### Create An Account

Name:

Email:

Username:

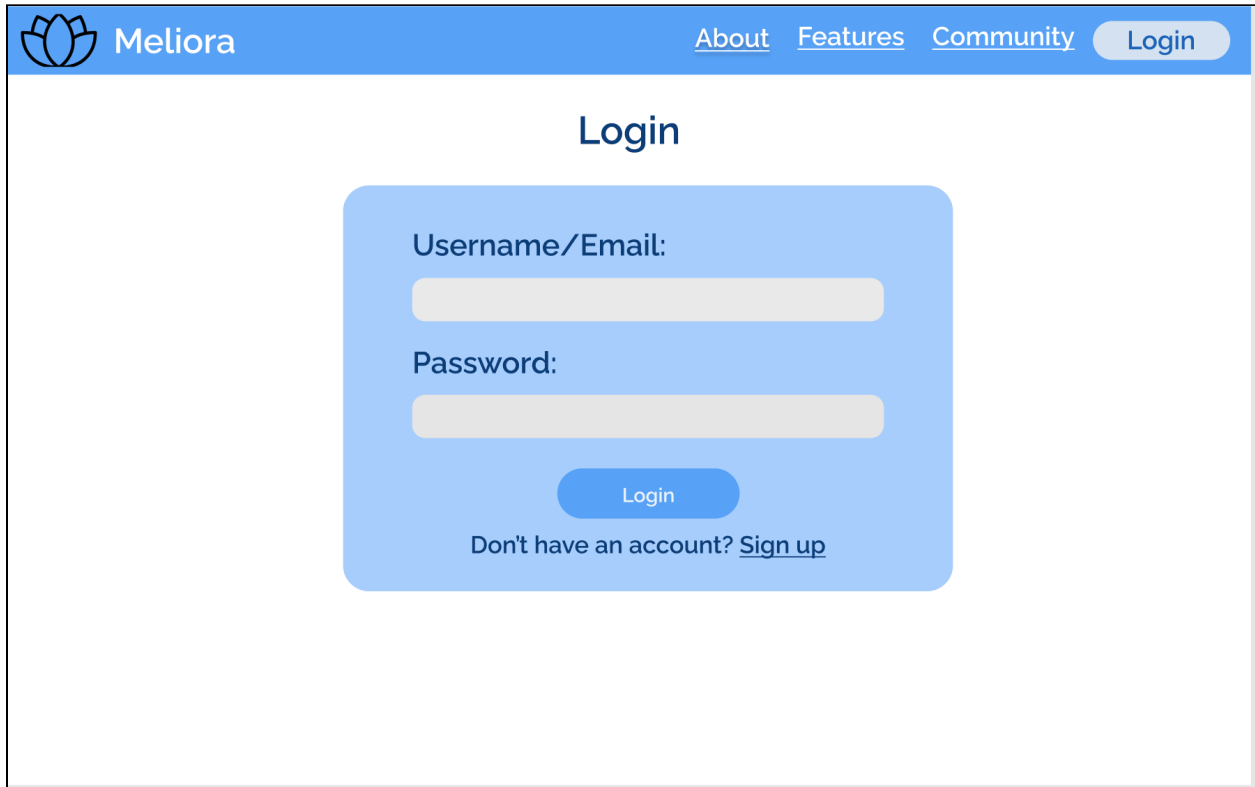
Password:

[Sign Up](#)

Already have an account? [Login](#)

This is the sign up page for *Meliora* that requires a new user to enter their name, email, username, and password. The button all the way to the bottom allows the user to login if he or she has an account already. Otherwise, the user can choose to sign up. An additional feature that our team hopes to implement is to include an additional check whether a particular user decides to sign up twice. If a user decides to sign up twice with the same email, then our application shouldn't allow him or her to do so.

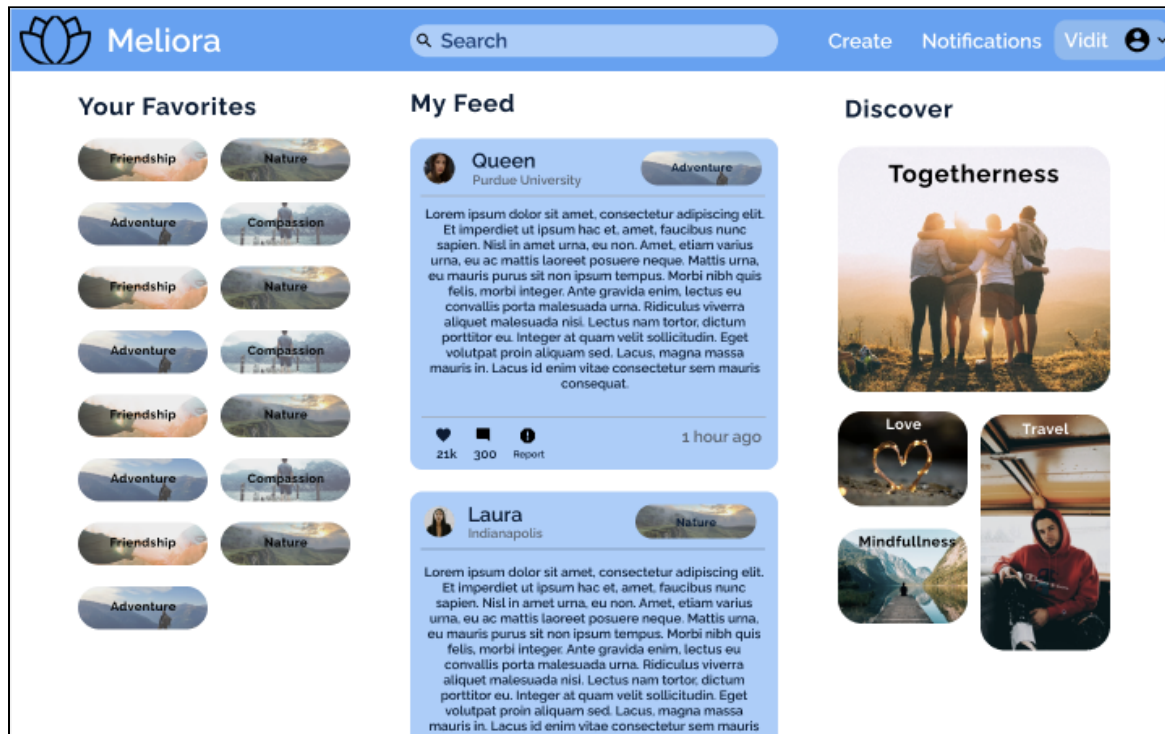
### 3 - Login Page



The screenshot shows the login page of the Meliora web application. At the top, there is a blue header bar containing the Meliora logo (a stylized lotus flower) on the left, the word "Meliora" in white text, and navigation links "About", "Features", and "Community" on the right. A "Login" button is also present in the header. The main content area is white and features a large, light blue rounded rectangle in the center. Inside this rectangle, the word "Login" is displayed at the top. Below it, there are two input fields: "Username/Email:" and "Password:". A blue "Login" button is positioned below the password field. At the bottom of the light blue rectangle, there is a link that says "Don't have an account? [Sign up](#)".

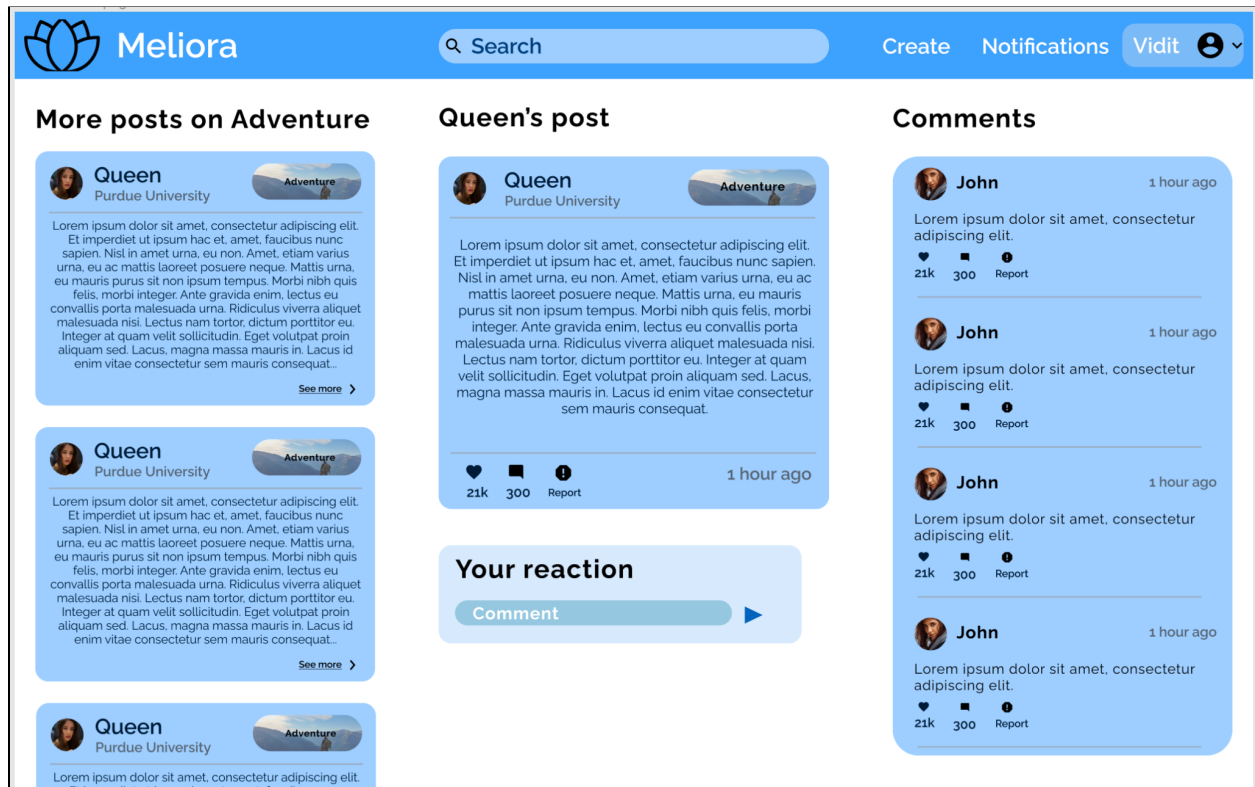
This is the login page of the Meliora web application. The user can login either by using their email or by using the username they created. There is also a sign up button for the user who has not signed up for the application yet.

## 4 - Home Page



This is the homepage of Meliora. Users can view their timeline where recent posts from the people they follow will appear. They also have the option of viewing a few categories that they like and those that are new to them, thereby allowing them to explore new categories. Through the navbar on the top, users can create posts, view their notifications, go to the profile page, and also search for categories and users through the search bar. The profile dropdown menu has options to go to My Profile, Profile Settings, and to logout of the application.

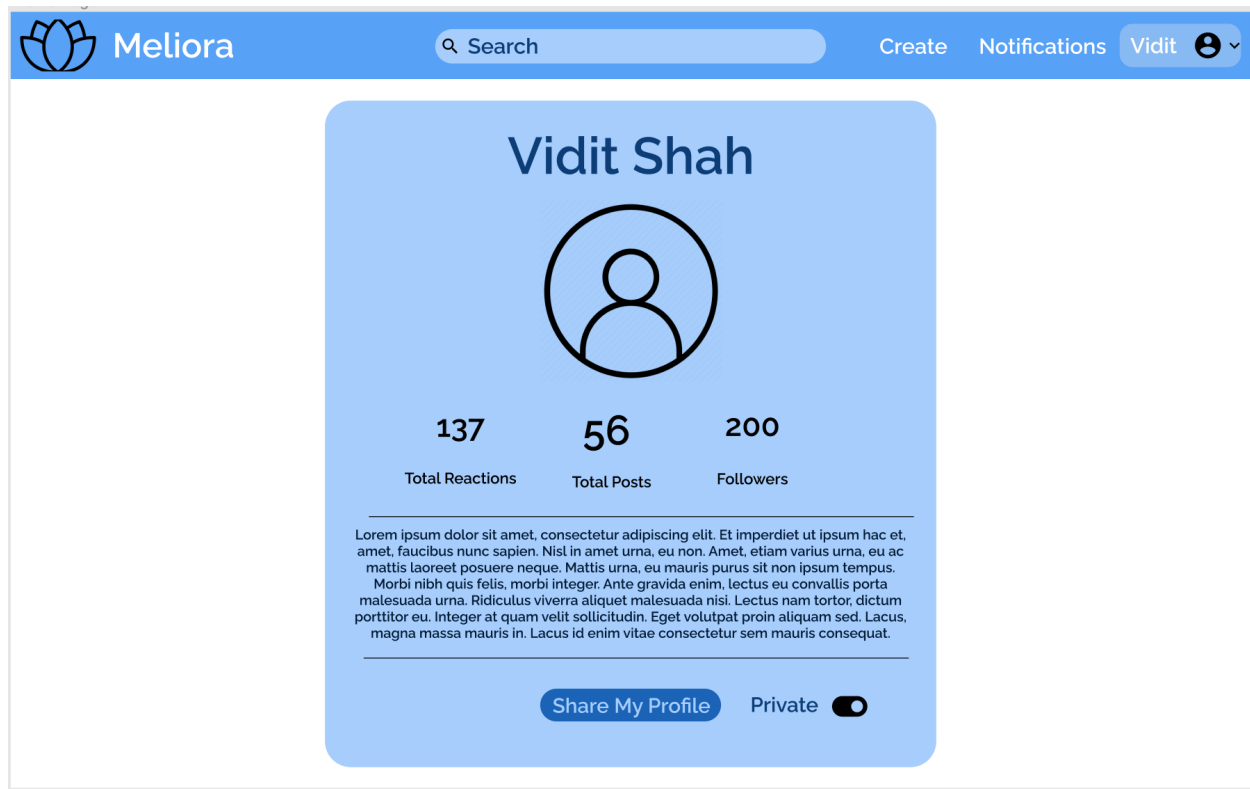
## 5 - Detailed Post Page



This is the detailed post page of Meliora. Users reach this page when they click on one of the posts on their timeline. On this page, the user can see all relevant information regarding the post such as the person who posted it, category the post belongs to, post content, and engagement information (likes and comments). The users can see all the comments to the post posted by other users as well as comment on the post themselves if comments are enabled for that post. On the left hand side, users can see other posts belonging to the same topic, thereby motivating the users to explore other users' experiences regarding the topic at hand.

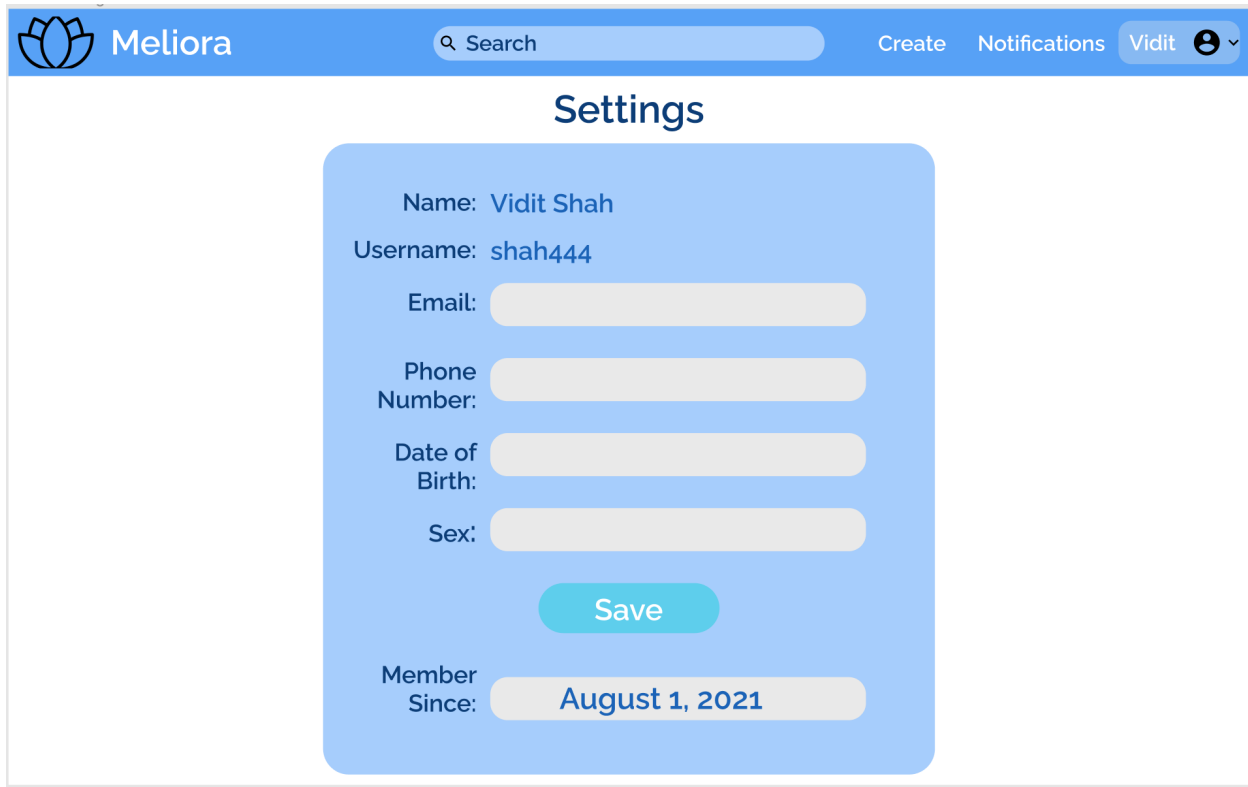


## 6 - My Profile Page



This is the profile page of the Meliora App. The user will be able to see the total number of posts they have created in their time on the app. They can also see the total number of reactions they have received from all of their posts, as well as the number of followers they have. A user can also provide a short bio about themselves on their profile page. Lastly, a user can click on the “Share My Profile” hyperlink to share their profile across multiple platforms.

## 7 - Profile Settings Page



The screenshot displays the 'Settings' page of the Meliora app. At the top, there is a blue header bar with the Meliora logo, a search bar, and navigation links for 'Create', 'Notifications', and a user profile dropdown labeled 'Vidit'. The main content area is titled 'Settings' and contains a light blue rounded rectangle with the following information:

- Name: Vidit Shah
- Username: shah444
- Email: [Input Field]
- Phone Number: [Input Field]
- Date of Birth: [Input Field]
- Sex: [Input Field]
- [Save Button]
- Member Since: August 1, 2021

This is the settings page of the Meliora App. The user will be able to see all their personal information on this page. A user will only be able to view their name and username, but they can edit their email, phone number, date of birth, and sex. When they make these changes, they can click the “Save” button to update these changes to the database. Lastly, they will also be able to see how long they have been a member on the app.