Lucas Damian Chindemi - 232399
Cristian Fernandez - 229611
Mònica Grau - 230461
Helena Pruna Craus - 231616

Report Unitter

Our project

Our 'Twitter' is themed around university students. The name we have chosen for it is: Unitter

Types of users

We contemplate three types of users:

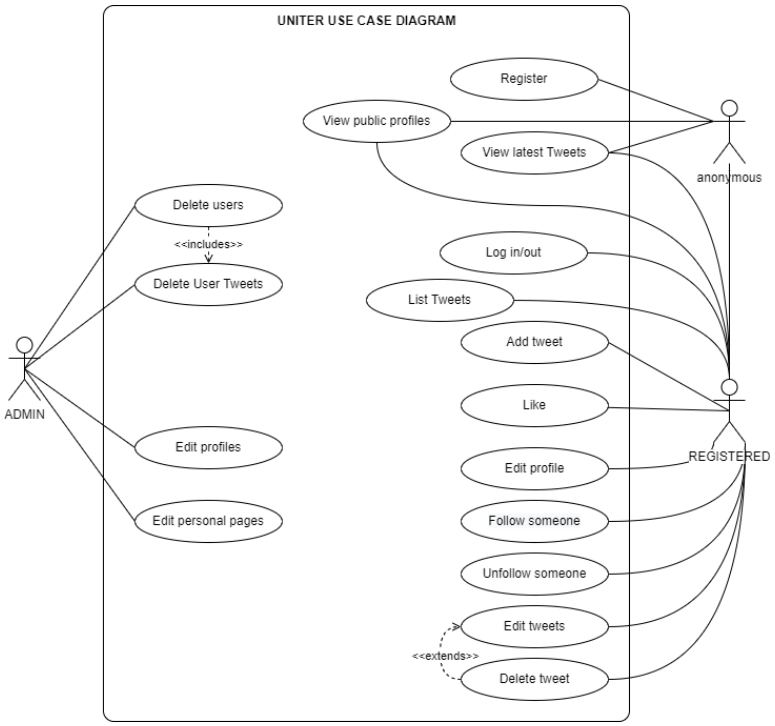
- Anonymous users (users that do not have an account): who can explore all the tweets from the users and see their profile but they won't be able to follow them, nor publish, retweet or like tweets.
- Registered users: the 'normal' users, who can publish, retweet, and like tweets, follow users, see their followers, edit or delete their tweets, and more.
- Administrators: which are similar to registered users but with added privileges like deleting and modifying users and their tweets.

User specifications

Field	Required	Format	Logical requirements	Technical requirements	Input type	Implementation details
Name	Yes	string	check format	-	text	-
User name	Yes	string	check format check availability	Check on server and client; check restrictions	text	no offensive names will be accepted
E-mail	Yes	string	check format check availability	Check on server and client	e-mail	check that it is from a university
Password	Yes	string	check format	masked check on server and client	password	store it hashed on the database
Password check	Yes	string	check format	masked check on client	password	-
Gender	No	string	-	-	dropdown	-
University	Yes	string	check format	-	dropdown	a list of universities from all the world will be displayed
Degree	No	string	-	-	text	-
Country	No	string	-	-	dropdown	all countries will be listed

Birthday date	No	date	check date format	-	dropdown	standard DD/MM/YYYY will be used
Student/ Teacher	No	string	-	-	dropdown	-

Users and functionalities



As mentioned above, in our twitter website (unitter), we have 3 different kinds of users: admins, anonymous users (not registered) and registered users. Each of them has different functionalities and privileges inside our webpage.

The anonymous user is sent to the explore tab and can see the most recent post and can click on the profiles and view them.

They can register/login themselves at any time by pressing the registration/login button.

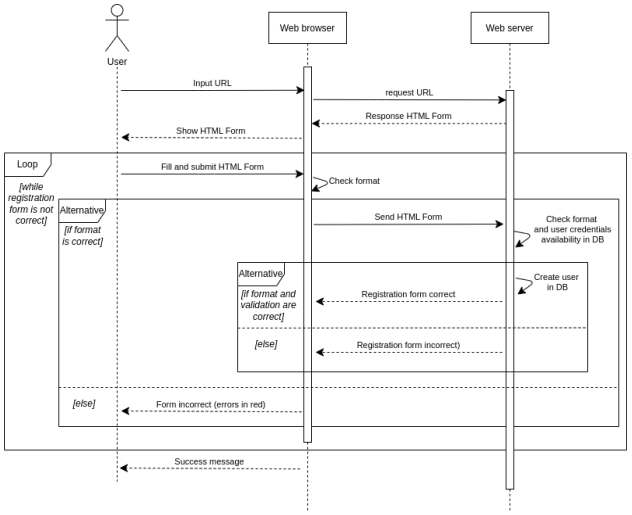
The registered user after having logged in can add tweets (maximum length of 200 characters) at the home and explore tab and edit and delete them whenever they want with the buttons at the right of each tweet. Furthermore, they can follow any user from “who to follow” or go directly to their profile where they can see the tweets and retweets of that user and follow or unfollow them. To get to the profile the user can click on the profile image/name/username of a tweet (if it’s a retweet they can press the name of the retweeter and then get the profile of them), also they can get to the profile from the “who to follow” and from the tab following. The registered user can like and retweet any tweet and remove those too.

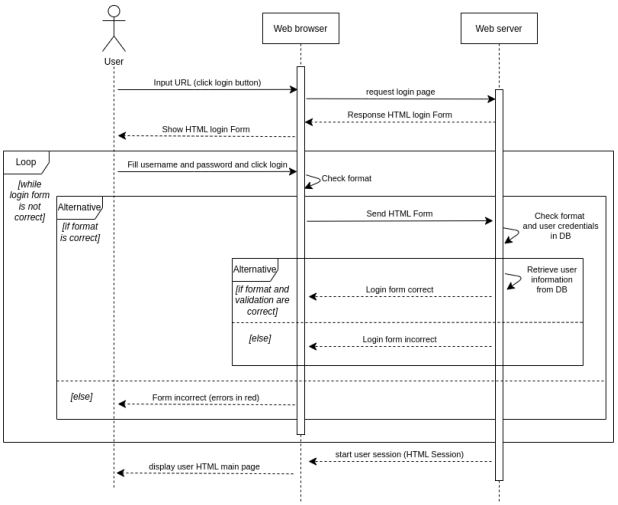
The administrator (assigned by us) can delete any user from the user’s profile and edit them. They can edit the tweets and delete them at the explore and home tab. Furthermore, they get all the functionalities of the regular user.

Lucas Damian Chindemi - 232399
Cristian Fernandez - 229611
Mònica Grau - 230461
Helena Pruna Craus - 231616

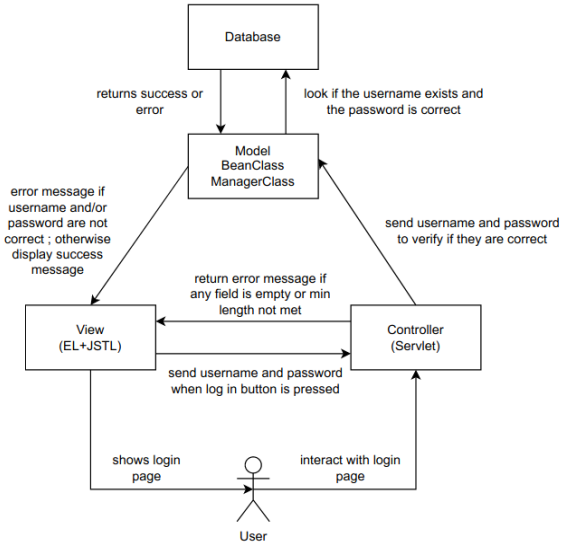
Universitat Pompeu Fabra Barcelona

Register and login page sequence diagram





MVC diagram



The different components we have on the MVC diagram are:

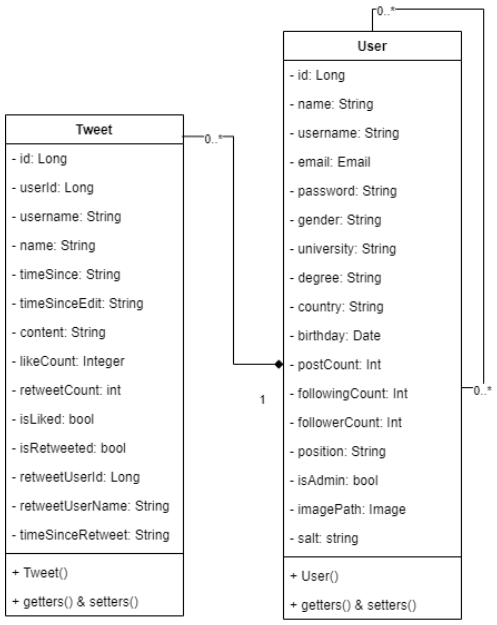
- the user: who sees and interacts with the login page, filling in the requested information and clicking the login button
- controller: which receives updates on the view and is in charge of doing a quick verification if the username and password fields are inputted correctly (not empty, password length, etc). If it is correct, it sends the credentials to the model to verify whether it is registered.
- model: holds the data structure and communicates with the database to verify if the username received exists and if its password is correct. Either way, it sends a message to update the view (display success/error message)
- view: shows the login page, specifically displaying the username and password fields for the user to input, and a submit (login) button. It is in charge of showing all the relevant information to the user and sending the data inputted to the controller.

Lucas Damian Chindemi - 232399
Cristian Fernandez - 229611
Mònica Grau - 230461
Helena Pruna Craus - 231616

Universitat Pompeu Fabra Barcelona

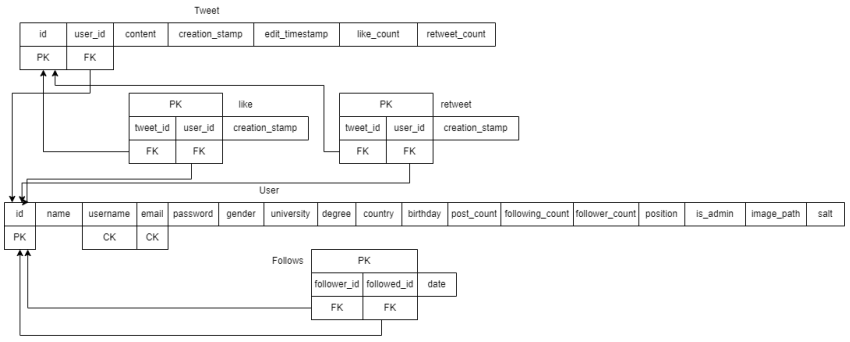
Twitter Database design

UML diagram:



As we can see in this diagram we have two classes, Tweet and User. Regarding the relations, we have two. Tweet - User: Each user can write 0 to many tweets, and the tweet has the id of the user who has published it. We store the number of posts and retweets done in the postCount variable.
User - User: each user can follow other users, this relation is many to many since each user can follow and be followed by many. We store the number of followers and following in the corresponding counters.
In the Tweet class, we will store its id, the user id, the username, and name to show who wrote the tweet, the timeSince to show when it was first written and timeSinceEdit in case the tweet has been edited it will also show when it was done. The tweet has an option to retweet and like and each Twitter stores the id of the user doing the retweet and the timeSinceRetweet to be able to show the *"{user.name} reposted {timeSinceRetweet} ago"*.

Relation Schema diagram:



In this relation schema diagram, we have the tables User, Tweet, Follows, like, and retweet.
User has a primary key called "id" and two candidate keys username and email as both must be unique.
The Tweet table has an id as a primary key and has two foreign keys and user_id which stores the id of the user that has written the tweet that cannot be null.
Lastly, we have three junction tables: Follows contains the id of the follower and followed users; Like has the id of the tweet and the user that likes the post; Retweet also the tweet_id and the user_id. In each junction, these elements are the primary key of the table and we have an added variable to store the date.
The values of User's table post, following and follower count, and of the Tweet's table the retweet and like count is computed with the help of triggers in the SQL code.

Lucas Damian Chindemi - 232399
Cristian Fernandez - 229611
Mònica Grau - 230461
Helena Pruna Craus - 231616

Visual design/branding decisions

Application name

As we have stated in previous seminars, the name of our application is Unitter.
The reason behind this is that our application will be focused mainly on university students and professors. Also, the mail functionalities will resemble those of the Twitter application. So: *Uni* → *uni*versity and *tter* → twi*tter*.

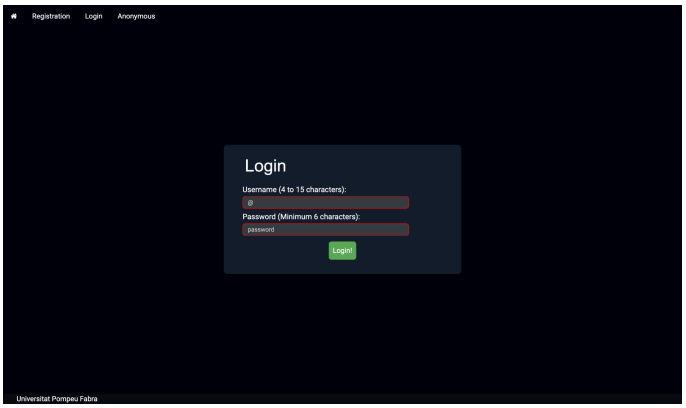
Logotypes

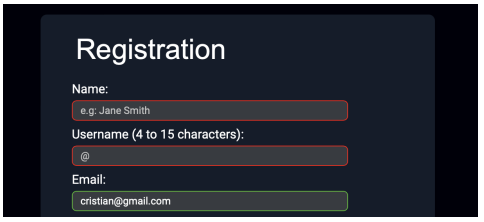
U Unitter

One of the logos (used as the favicon of the website) we have created is a U that represents the first letter U of Unitter. The other logo is the name of the website itself. This logo is shown inside our application, once you log in or you enter in anonymous, in the top-left corner below the top navigation bar and is always visible.
For this logo, we have chosen the 'Anek Malayalam' font which is available through the Google Fonts project. We use this font because it has a modern artistic style that we quite like and it fits the aesthetic of the rest of the website, furthermore it is licensed under the Open Font License and it is free to use.

Color palette

Regarding the color palette, we have chosen to go with a 'dark mode' palette for our website, which is quite trending nowadays. We have based our style on Twitter and other websites of similar thematic (since why would we reinvent the wheel given that these companies invest millions of dollars in researching the best possible design). We use plain black (#000) as our background color, with accents of blue.





This can be noticed from the first moment users enter our website, to the login page. We also color code the validation of the input fields in green or red to give users an idea of what they are doing. We use a brighter blue for the Post button since it's one of the most important aspects of the page and we want to highlight it clearly and give it a different style from the rest. We also use white and different tonalities of the blue color for highlights, hovers, and focus.

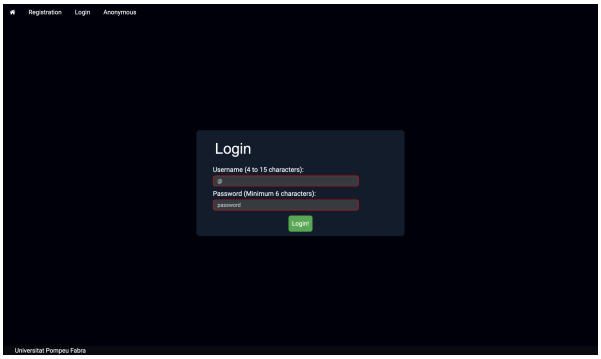
We also use blue to define the “grid lines” of our layout, and a light gray for the text that is not as important. To sum up, here you can see a summary of the main colors in our color palette:

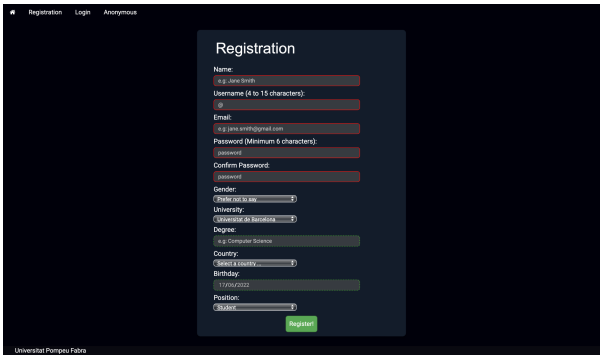
Typography

Regarding the typography, as we have already mentioned we use Anek Malayalam for the logo. However, in the rest of the website we use Roboto, which is a widely used font, for example by Google in Android. It is very readable and elegant, and it comes with many font weights and other attributes, so it is a good choice for the website. It is also available on Google Fonts.

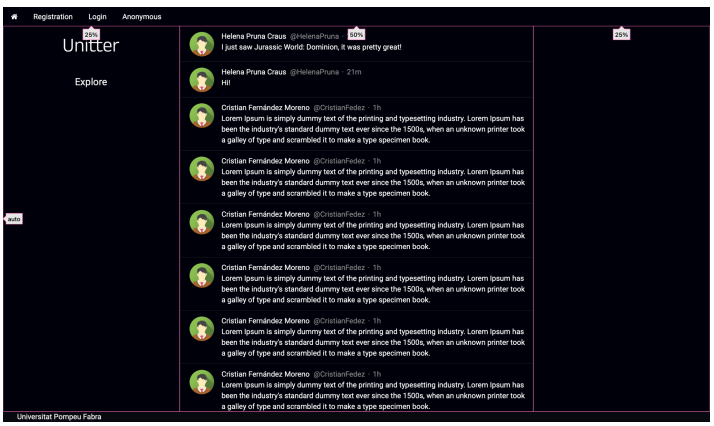
Layout / Functionality

Before logging in





When the user enters our web for the first time he sees the login page. Here as you can see we have a top navigation bar where he can go to the Registration, Login, or Anonymous pages. The registration form follows the table user specifications.
The container for the login and register pages is centered in the middle of the page, in which we display the form for the user to fill in and log in to our application.

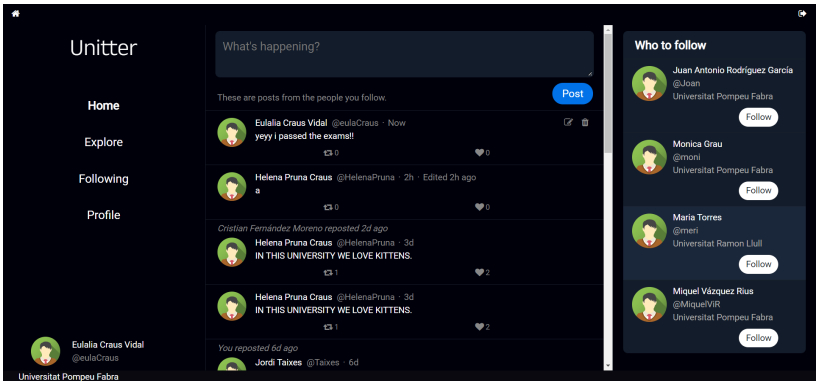


If the user decides to go anonymous browsing, they will be redirected to a view where we can already see the main layout of the website.
Here even though the actions of an anonymous user are limited (only global timeline and view profiles), we can already see the main layout of our website, which consists of three columns, organized using CSS Grid, which is organized in width as

25%, 50%, 25%. And the footer will be always at the bottom of the page. This CSS grid is the 'wrapper' that contains all the rest of the content, which will be organized in each of the columns. The middle column contains the tweets and can be scrolled vertically, without the rest of the page moving.

After logging in

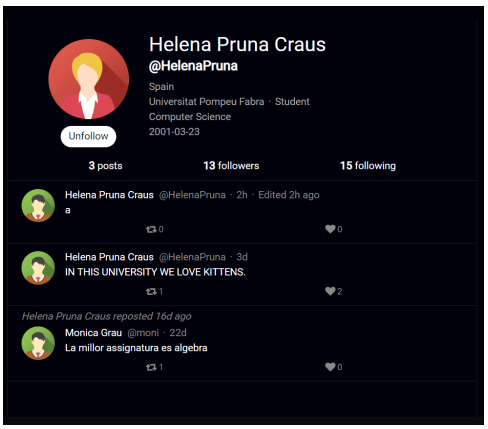
Once we are logged in, we can see that on the top navigation bar we don't have the login, registration, and anonymous options anymore, and instead, only the icons remain, instead we want the users to navigate using the menu on the left column, which has 4 different options: Home, Explore, Following and Profile. Above the menu, we have our website name such that it always remains visible. This menu will always be visible as we move through the different pages. The left column also has a container with the information of the user that has logged in at the bottom. In the right column, we have a container with the people that we may want to follow and from there we can follow them by clicking the button. This container will also always be visible.



Tweet timeline

In the 'Home' and 'Explore' options we can see the custom and global timelines, respectively, they are the same in layout, which we can see in the above pic, the only thing that changes is the message below the input which says 'These are posts from the people you follow' or 'These are posted from everyone' and the content itself. The layout is very similar to that of the anonymous timeline, with the scrolling list of tweets, but above it now we have the input area with a post button.

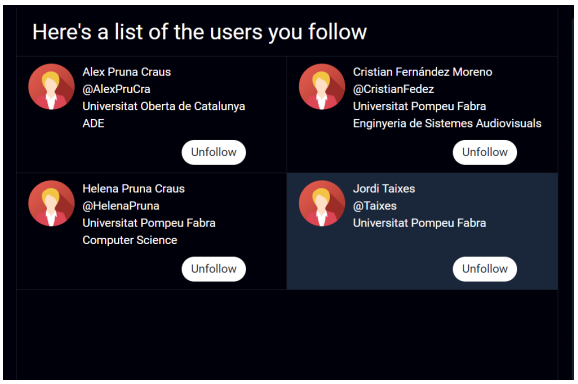
Profile page



By clickin in any of the users of the tweets, or in any of the 'who to follow' users, we go to a profile page that loads in the middle column of our layout.
At the top is the profile picture and information of the user. Below the profile, we have a Follow/Unfollow button that changes when clicked and below that some stats regarding the post, follower, and following counts of the user. Finally, below the information, we have the tweets of the specific user.

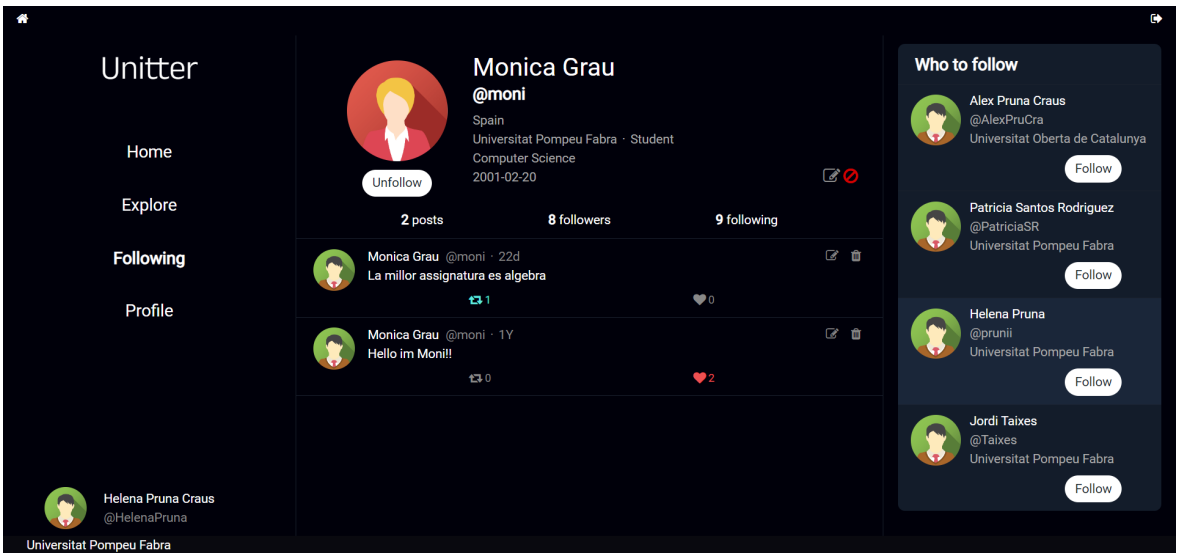
If the profile the user clicks or they press the icon at the bottom of the left colomn the profile shown will be their own, and it will share some similarities with the previous one but with the diference that there is no follow/unfollow button, and the user will be able to edit the profile.

Following page



Finally, if we choose the 'Following' option in the menu, we go to a view with a list of all the users that we are following.
In this case, the middle column is organized with a grid layout with two columns, and the 'box' of every user is organized automatically when added to the layout. You can either go to one of the following users or directly unfollow them.

Administrator



As explained before the administrator user has other functionalities and as we can see in this image the user with username "@HelenaPruna" is an administrator, and she can edit any tweet she wants, but in addition is that she can edit the profile, and if she deems necessary delete the user from the web pressing the red button.

Lucas Damian Chindemi - 232399
Cristian Fernandez - 229611
Mònica Grau - 230461
Helena Pruna Craus - 231616

CSS

File structure

To work properly and in an organized way we have decided to separate our CSS styles into different files, each for a different purpose, roughly following the idea of 'one-view-one-css'. They are:

- mainStructure.css: which holds all the styles for the body, content, navigation, and footer.
- form.css: which holds all the styles related to the forms: login and register. It has the errors, and the input styles for valid/no valid.
- profile.css: which holds all the styles related to the user profiles.
- timeline.css: which holds all the navigation styles from the tweets and users and in general the main page.
- followedUsers.css: which holds all the styles related to the 'Following' view.
- anek.css: used to retrieve the Anek Malayalam font family so we can use it on our application.

These CSS files will all be loaded into our index.jsp at startup.

Frameworks and access to the elements in the DOM

We don't use any specific frameworks for CSS, we write it ourselves using the CSS standard and mainly flex/grid. To access the elements in the DOM we try to stick to Vanilla JS as much as possible since it is a lot quicker than using jQuery or similar frameworks and it has improved a lot over the past few years, but in some exceptions where readability is more important, we still use jQuery. In general, we will try to separate content in HTML, style in CSS, and functionality in JS, although very occasionally when modifying the page actively we use JS in the client to add/delete elements in HTML or modify the style in others.

Accessibility

We think that the text on our website has good contrast to be read without problems, and the color palette (black with blue as an accent color) makes it so that people with common color blindness (such as green-red or blue-yellow) can see the website well.