HSC Major Project – Web Technologies

Students find a client for whom they can develop a web-based software solution. Students unable to find a client may use Scenario 1 or Scenario 2 in the Student support material section of this assessment task.

The Task Description

A: Identifying and Defining

Interview

Interview Questions:

Client 1

Client 2

Features guaranteed to be included:

- Accounts
- Following
- Blocking people
- Private messaging
- Group chats (in some form)

1. What are your favourite features from other social media sites?

   The ability to make chats with each other

   Friends, following people, likes and dislikes, comments, comment chains/replies, pings/mentions, personal page/story, topics, DMs, blocking.

2. What are your least favourite features from other social media sites?

   Ads

   Lack of moderation, advertisements, scrollable short-form videos.

3. If you were to pick another social media site style (functionality wise) that is your favourite (discord, twitter, Instagram, facebook, teams, whatsapp) what would you want this to be most like?

   Discord

   Twitter.

4. Are there any unique features you would like this site to have?

   Privacy, I'd like the site to focus on user privacy

   Topics like in a forum site, but more simplified.

5. What colours would you like the website to predominantly feature?

   Any colour that suits the “feel” of the app, purple maybe???

   Black, lilac, purple.

6. What type of device would this app be predominantly used on?

   Desktop

   Laptop or desktop.

7. What would you like the name of the website to be?

   I dont know yet

   N/A

8. What vibe would you like the website to have? Casual, formal, fun, etc.

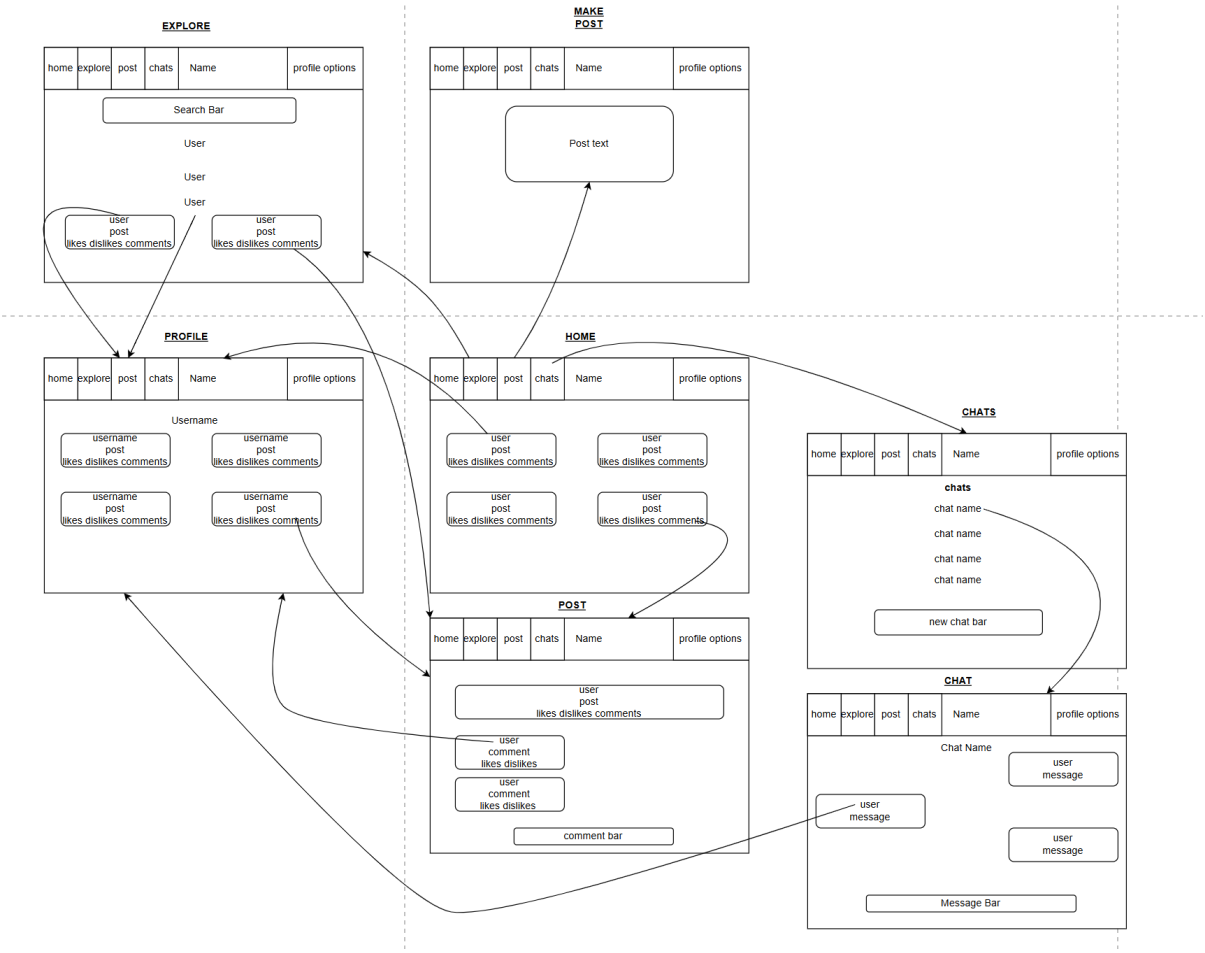
   Formal / casual

   Casual, fun, whimsical.

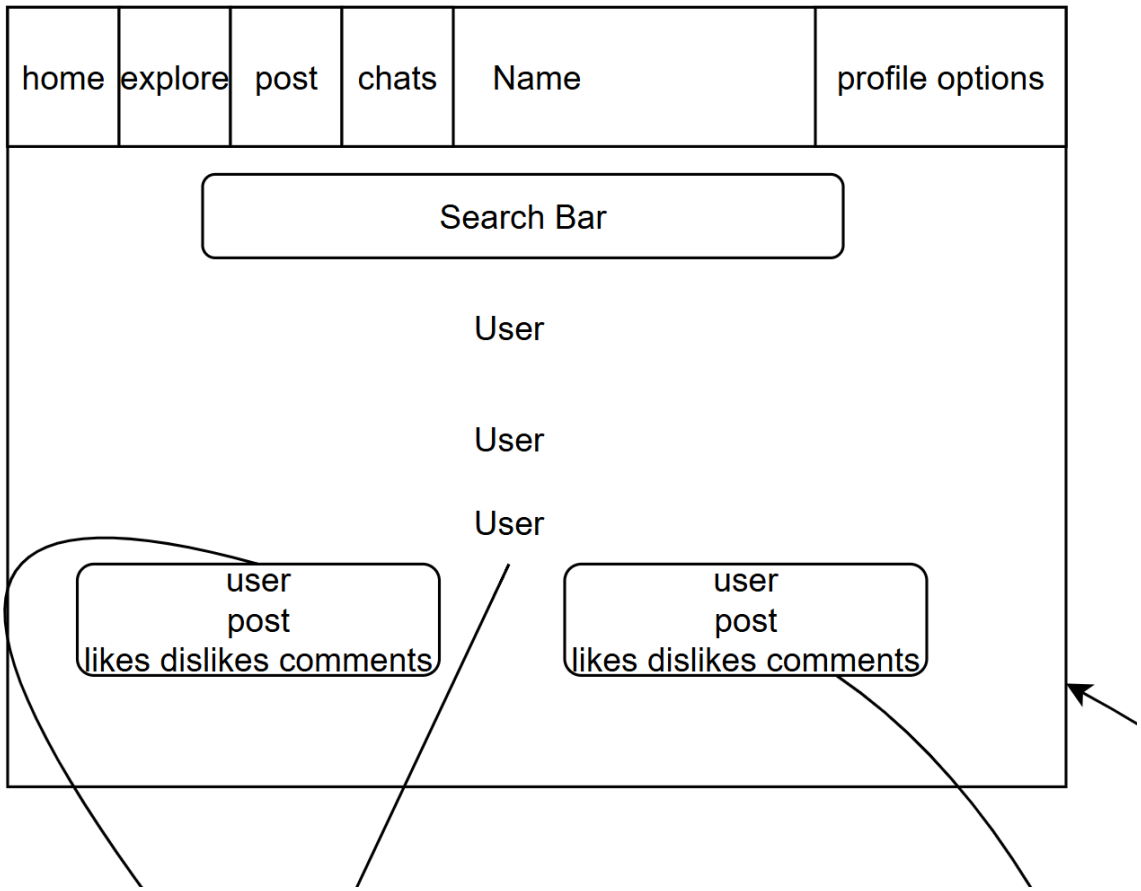
Requirements Definition:

- Must be able to make group chats
- Must be able to follow people
- Must be able to block people
- Must have likes and dislikes
- Must not have ads
- Must have hashed passwords (privacy concern)
- Must have purples/lavenders/lilac colour scheme
- UI must be primarily designed for laptop
- Must have a casual ‘vibe’
- Must be able to post
- Must be able to comment

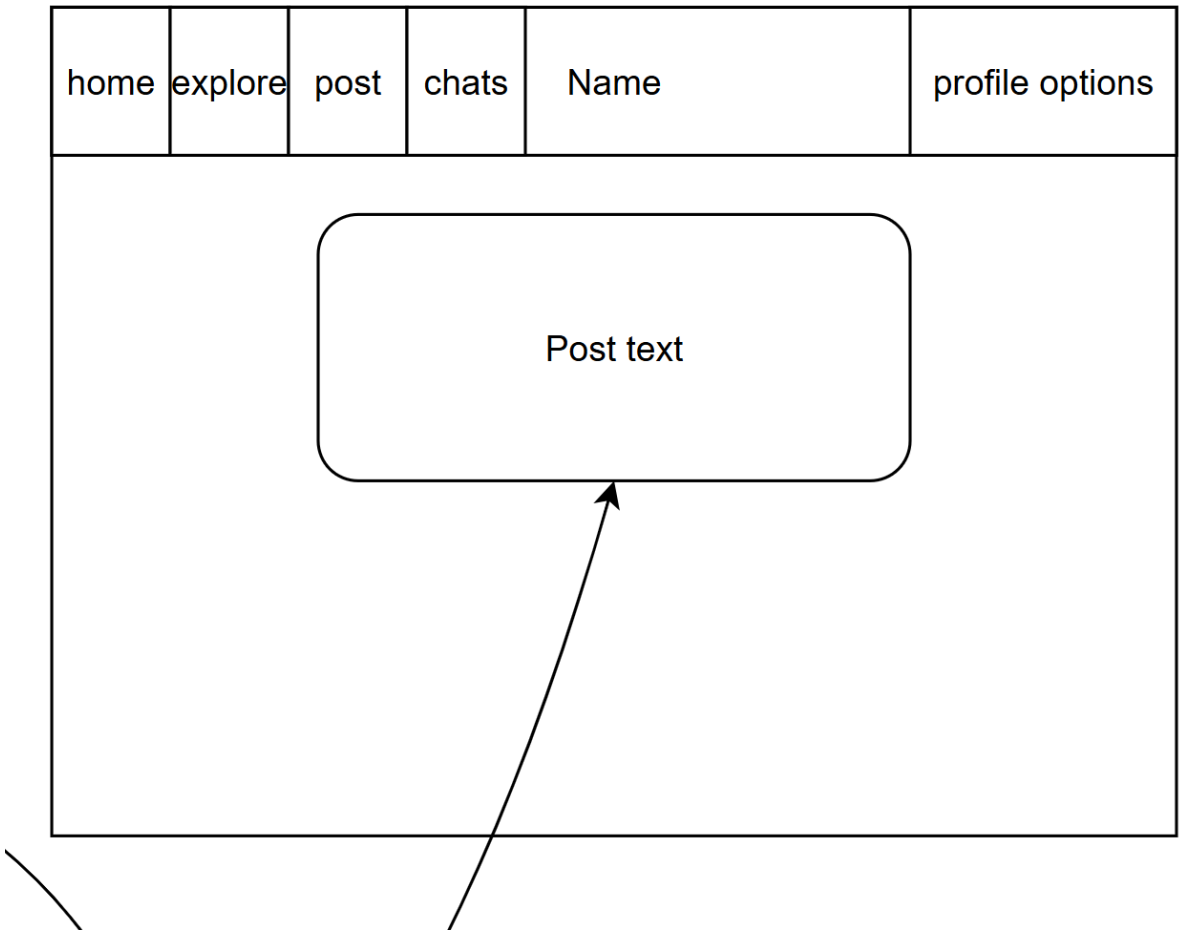
Storyboard

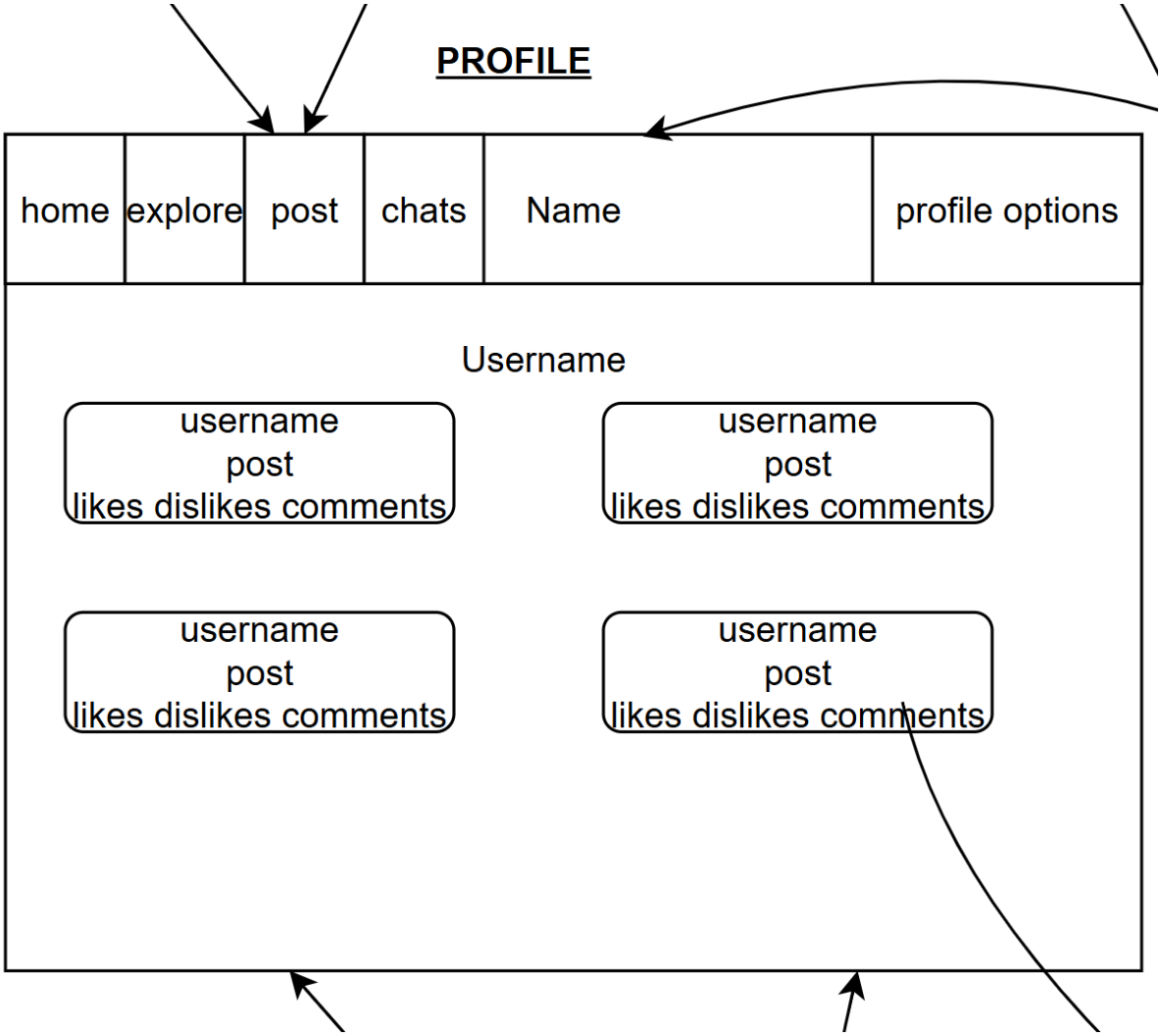

EXPLORE
home
explore
post
chats
Name
profile options
Search Bar
User
User
User
user
post
likes dislikes comments
user
post
likes dislikes comments
MAKE POST
home
explore
post
chats
Name
profile options
Post text
PROFILE
home
explore
post
chats
Name
profile options
Username
username
post
likes dislikes comments
username
post
likes dislikes comments
username
post
likes dislikes comments
username
post
likes dislikes comments
HOME
home
explore
post
chats
Name
profile options
user
post
likes dislikes comments
user
post
likes dislikes comments
user
post
likes dislikes comments
user
post
likes dislikes comments
POST
home
explore
post
chats
Name
profile options
user
post
likes dislikes comments
user
comment
likes dislikes
user
comment
likes dislikes
comment bar
CHATS
home
explore
post
chats
Name
profile options
chats
chat name
chat name
chat name
chat name
new chat bar
CHAT
home
explore
post
chats
Name
profile options
Chat Name
user
message
user
message
user
message
Message Bar

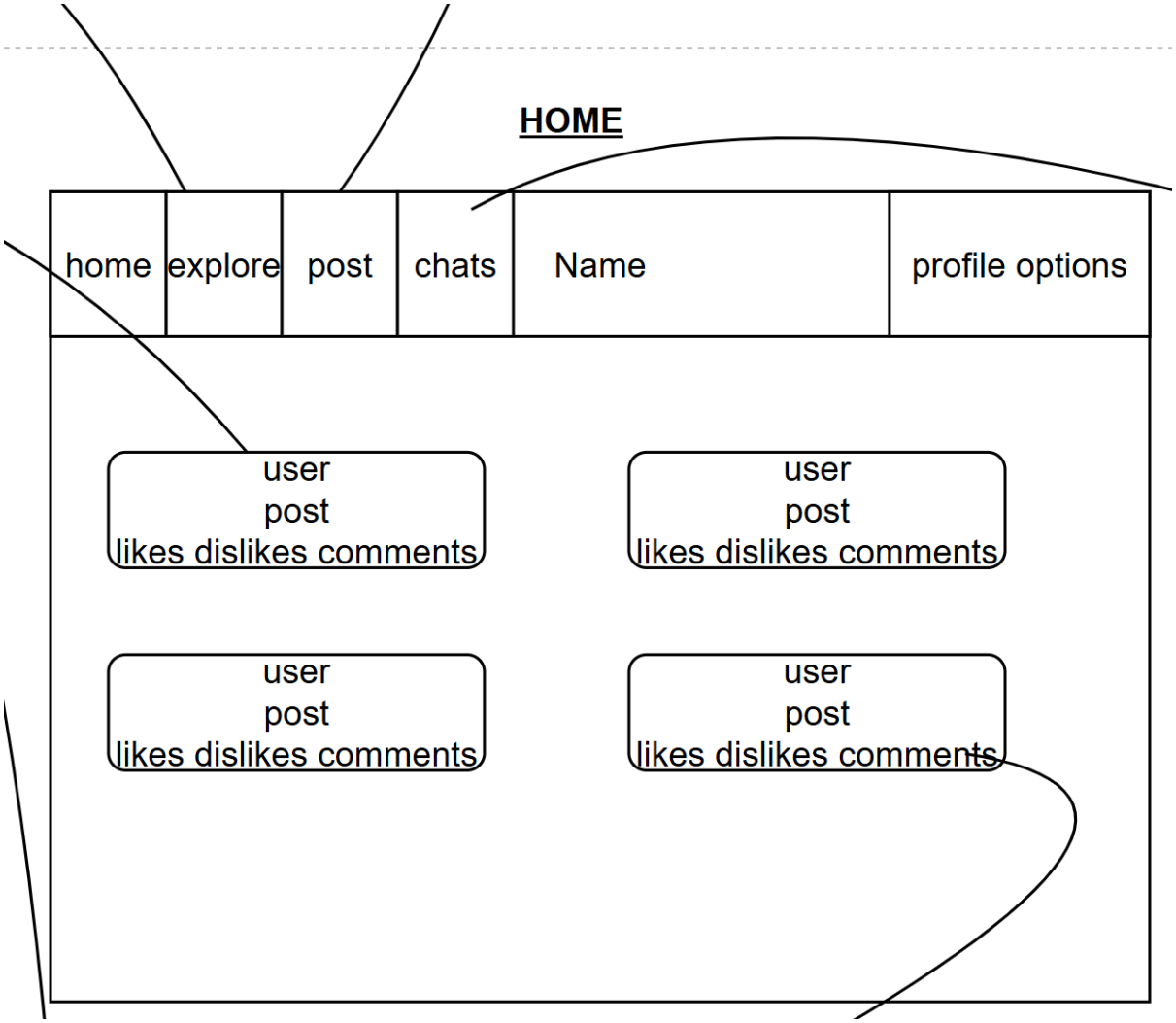
EXPLORE

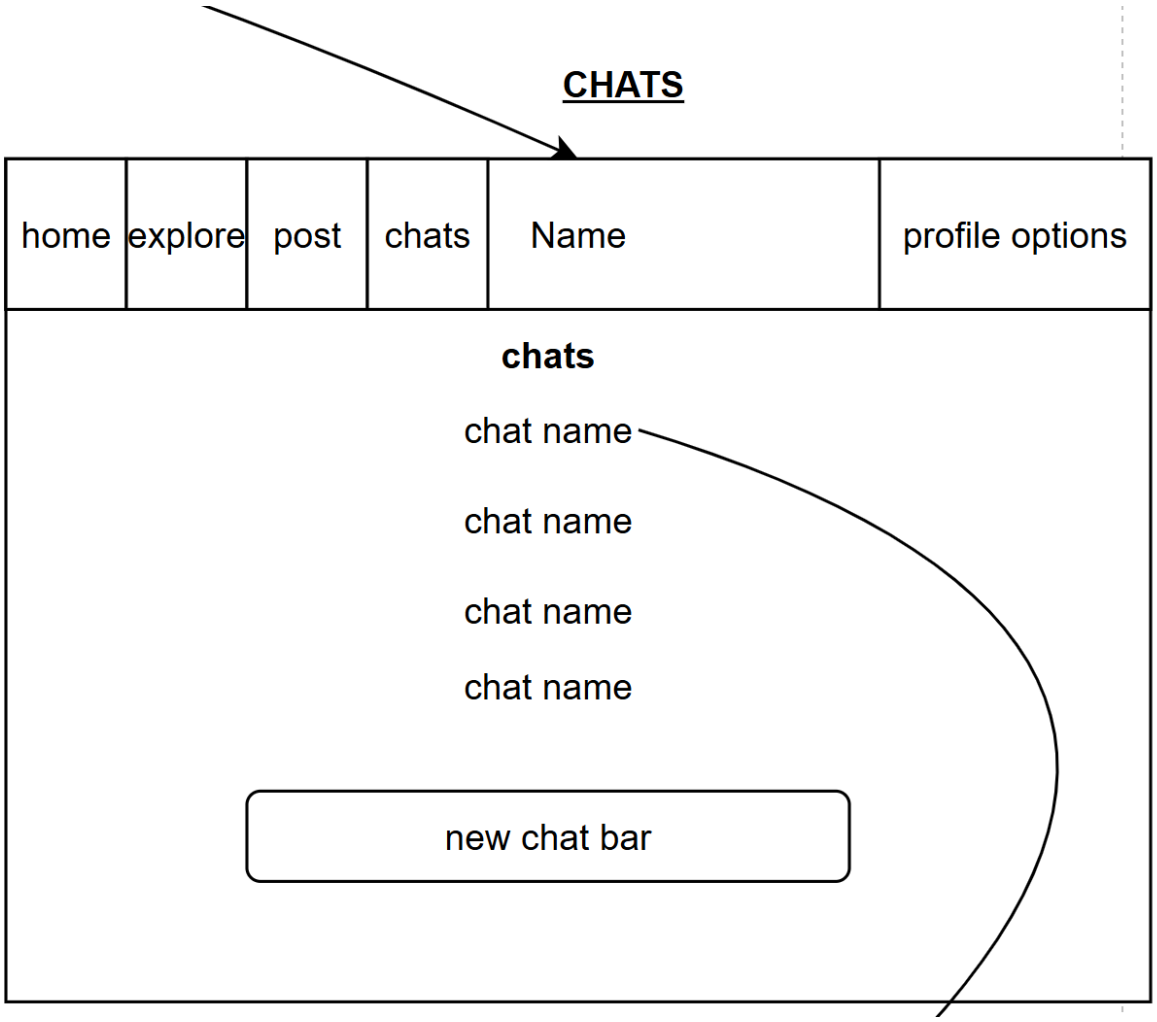

home
explore
post
chats
Name
profile options
Search Bar
User
User
User
user
post
likes dislikes comments
user
post
likes dislikes comments

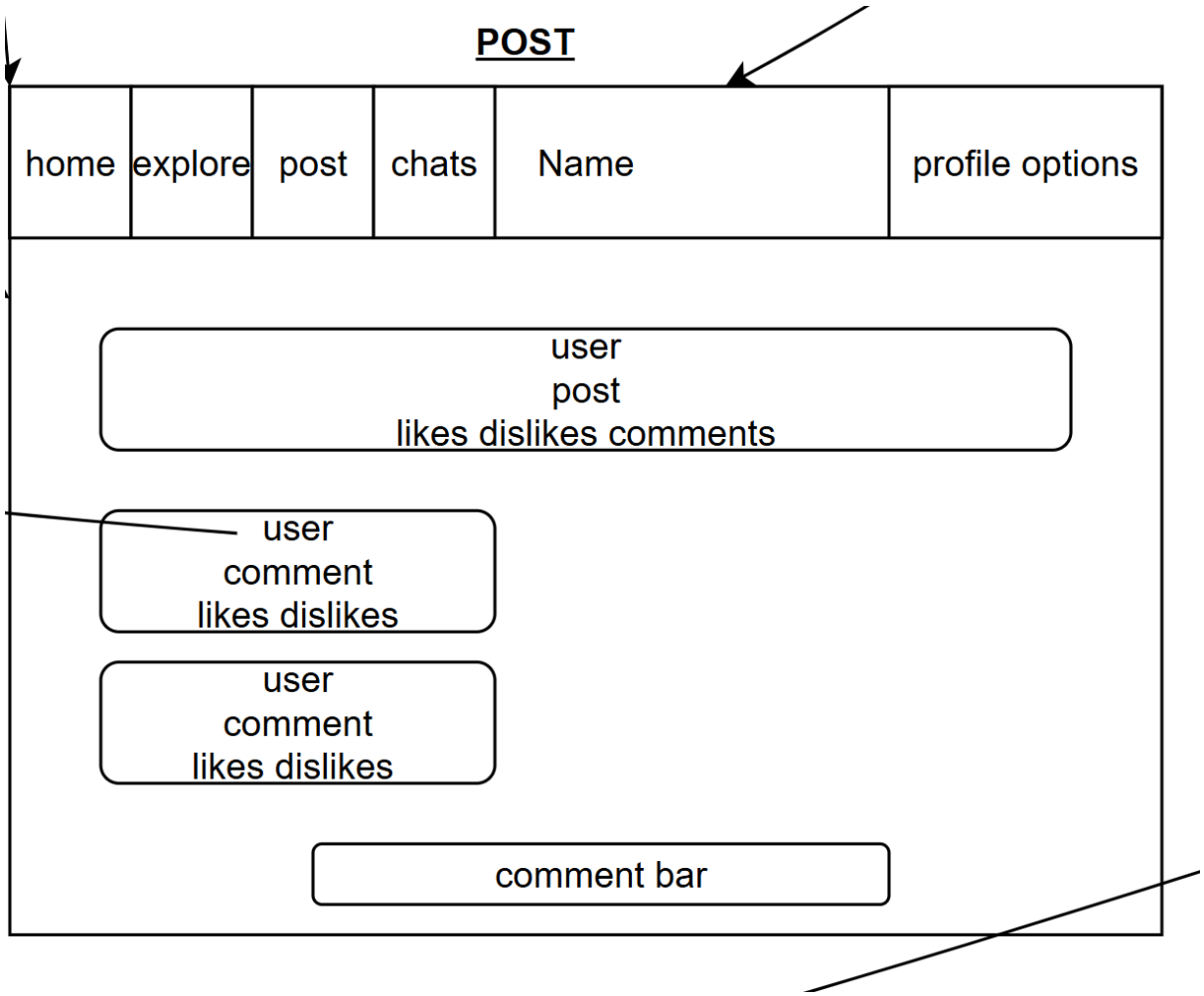
MAKE POST

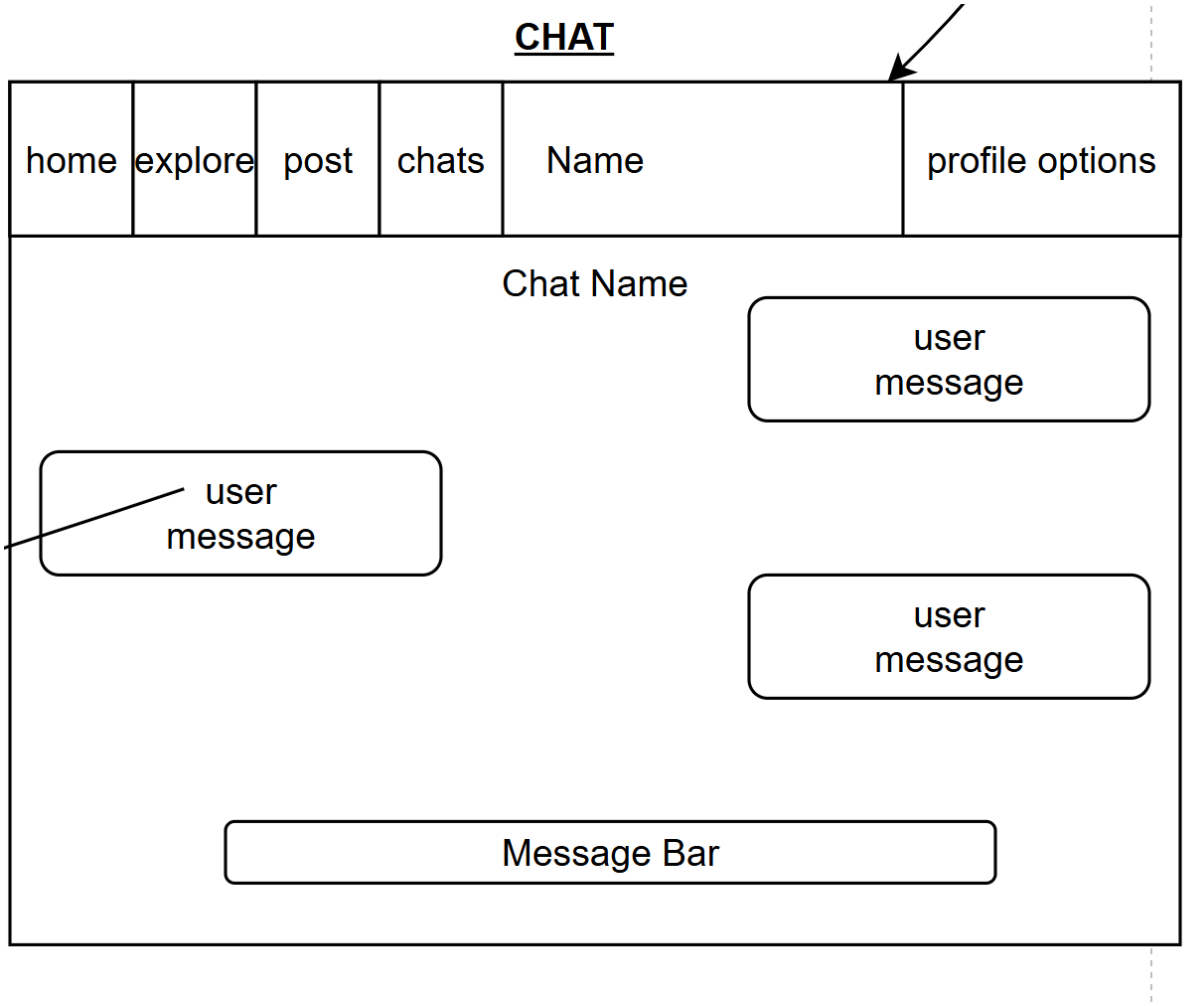



PROFILE
home
explore
post
chats
Name
profile options
Username
username
post
likes dislikes comments
username
post
likes dislikes comments
username
post
likes dislikes comments
username
post
likes dislikes comments

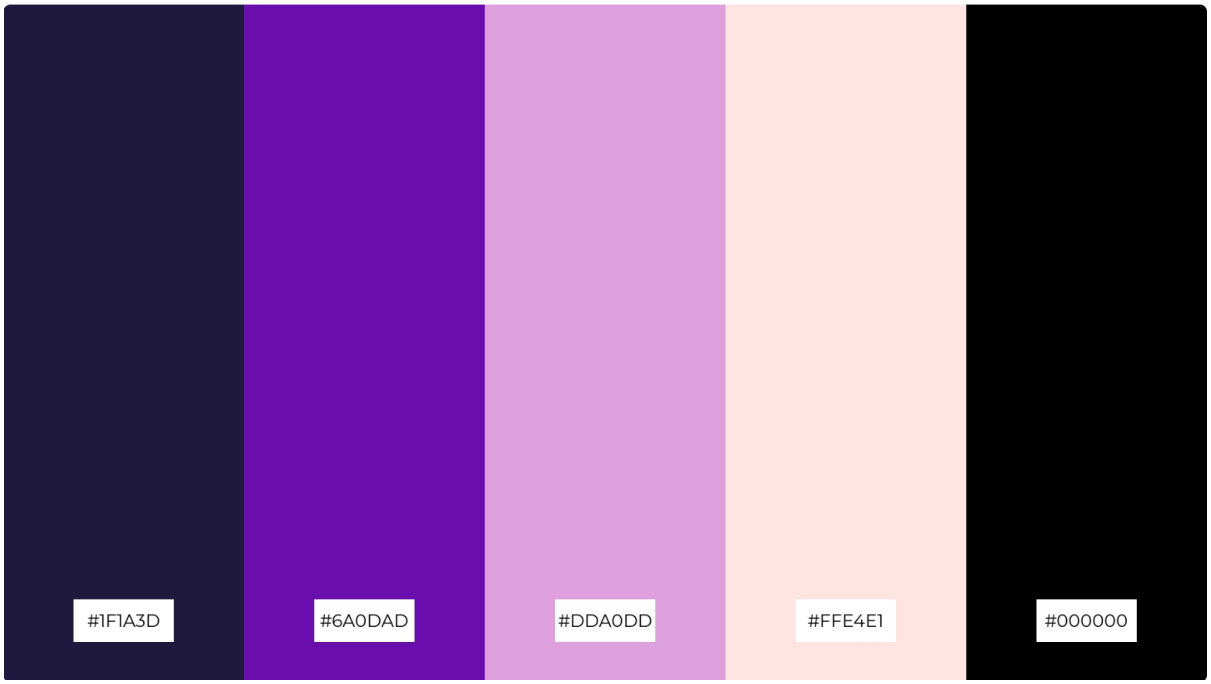

HOME
home
explore
post
chats
Name
profile options
user
post
likes dislikes comments
user
post
likes dislikes comments
user
post
likes dislikes comments
user
post
likes dislikes comments


CHATS
home
explore
post
chats
Name
profile options
chats
chat name
chat name
chat name
chat name
new chat bar


POST
home
explore
post
chats
Name
profile options
user
post
likes dislikes comments
user
comment
likes dislikes
user
comment
likes dislikes
comment bar



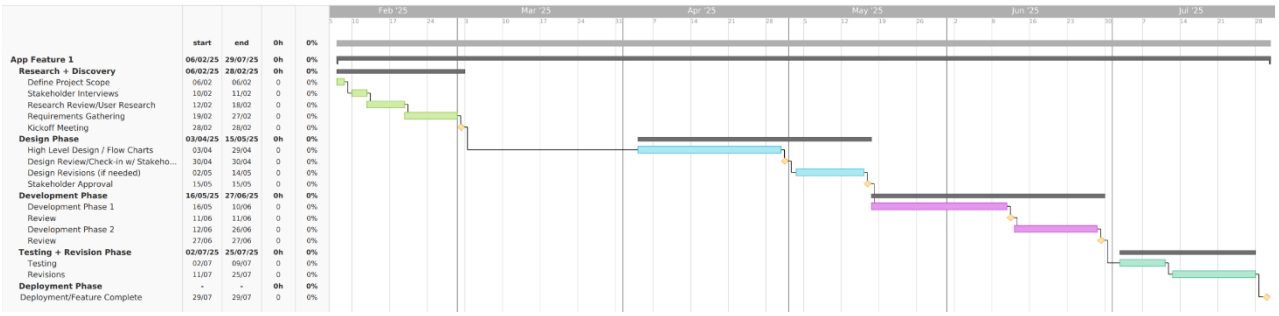
Colour Scheme



B: Research and planning

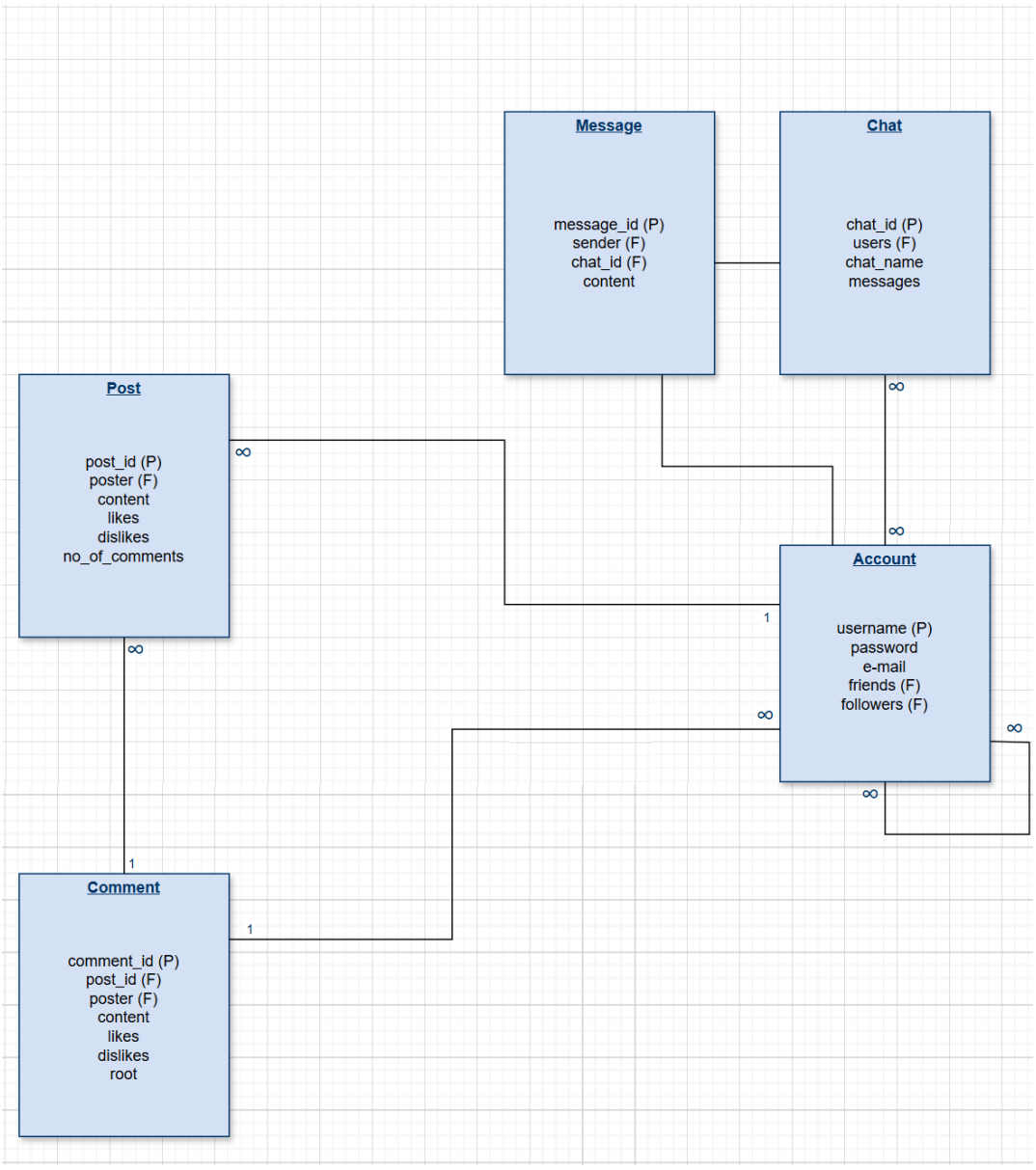
Gantt Chart

[gantt chart.pdf]



	start	end	0h	0%
App Feature 1	**06/02/25**	**29/07/25**	**0h**	**0%**
Research + Discovery	**06/02/25**	**28/02/25**	**0h**	**0%**
Define Project Scope	06/02	06/02	0	0%
Stakeholder Interviews	10/02	11/02	0	0%
Research Review/User Research	12/02	18/02	0	0%
Requirements Gathering	19/02	27/02	0	0%
Kickoff Meeting	28/02	28/02	0	0%
Design Phase	**03/04/25**	**15/05/25**	**0h**	**0%**
High Level Design / Flow Charts	03/04	29/04	0	0%
Design Review/Check-in w/ Stakeho...	30/04	30/04	0	0%
Design Revisions (if needed)	02/05	14/05	0	0%
Stakeholder Approval	15/05	15/05	0	0%
Development Phase	**16/05/25**	**27/06/25**	**0h**	**0%**
Development Phase 1	16/05	10/06	0	0%
Review	11/06	11/06	0	0%
Development Phase 2	12/06	26/06	0	0%
Review	27/06	27/06	0	0%
Testing + Revision Phase	**02/07/25**	**25/07/25**	**0h**	**0%**
Testing	02/07	09/07	0	0%
Revisions	11/07	25/07	0	0%
Deployment Phase	-	-	**0h**	**0%**
Deployment/Feature Complete	29/07	29/07	0	0%

ERD



Data Dictionary

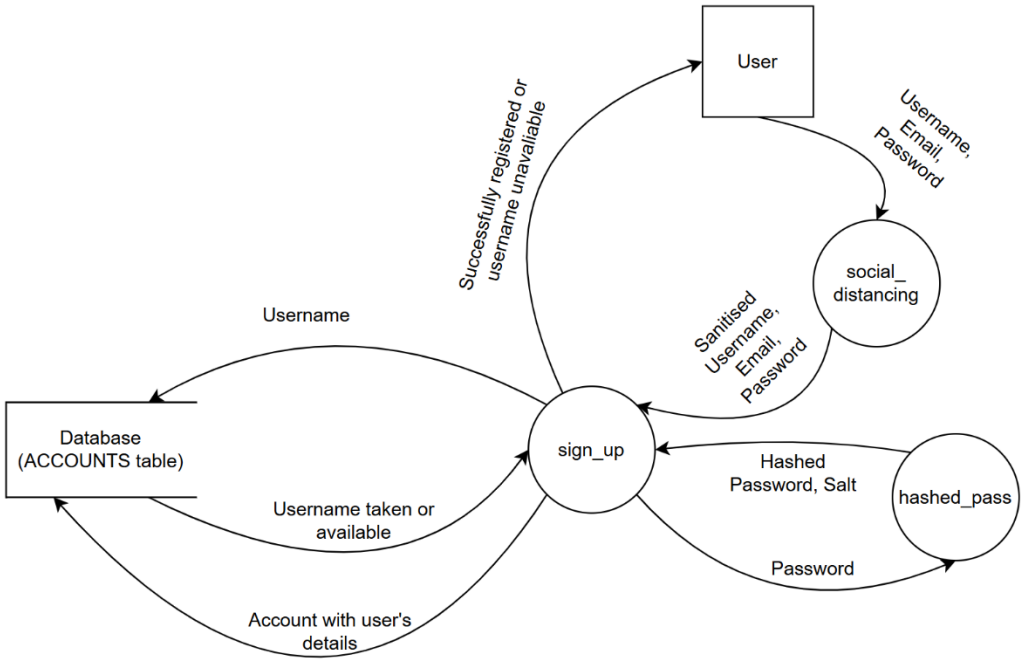
Name	Table	Description	Data Type	Accepted values	Example
chat_id	CHATS	Primary key, used to identify and keep track of each chat.	Integer	Unique integer values where chat_id > 0	1

users	CHATS	A string containing all of the users with access to the chat separated by slashes	Text	All strings	CharlieM/test/admin
chat_name	CHATS	Keeps track of the name of a chat, defaults to listing users	Text	All strings	CharlieM, test, admin
message_id	MESSAGES	Primary key, used to identify and keep track of each message.	Integer	Unique integer values where message_id > 0	1
content	MESSAGES	Contains the content of a message	Text	All strings	Hello my name is jim
sender	MESSAGES	Contains the username who sent a message	text	All strings	CharlieM
chat_id	MESSAGES	Foreign key, contains the chat in which the message was sent	Integer	integer values where chat_id > 0	1
username	ACCOUNTS	Primary key, used to identify and keep track of each account.	text	Unique strings	CharlieM
password	ACCOUNTS	Contains the hashed (password+salt) to the account	text	All strings	61bba07db45f3e121b75828b2e8abdbec24d6fd2085958cb2d54f011d96dc33c
email	ACCOUNTS	Contains the email associated with the account	text	All strings	Test@test.com
salt	ACCOUNTS	Contains a randomly generated used to salt the password	text	All strings	UOgsNh5j4EFvnbVIFQBzeOPOqW0Caths
followers	ACCOUNTS	A string containing all of the other users who have followed the user separated by slashes	text	All strings	CharlieM/test/admin
blocked	ACCOUNTS	A string containing all of the users that the user has blocked separated by slashes	text	All strings	CharlieM/test/admin

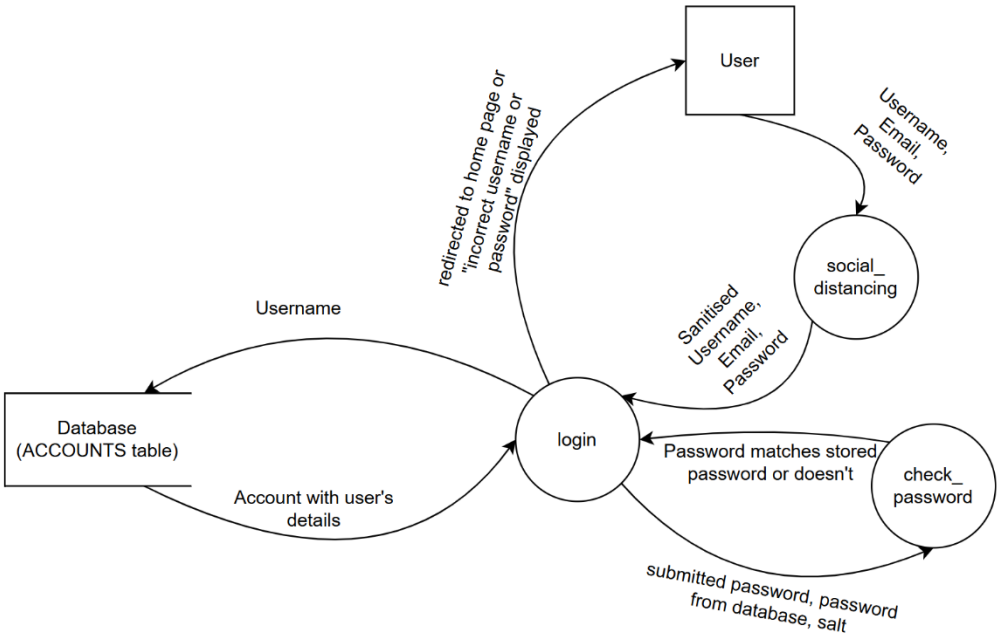
post_id	POSTS	Primary key, used to identify and keep track of each post.	Integer	Unique integer values where post_id > 0	1
poster	POSTS	Contains the username who posted a post	text	All strings	CharlieM
content	POSTS	Contains the text content of a post	Text	All strings	Hello my name is bob
dislikes	POSTS	Tracks the users who have disliked the post in a string separated by slashes	text	All strings	CharlieM/test/admin
likes	POSTS	Tracks the users who have liked the post in a string separated by slashes	text	All strings	CharlieM/test/admin
comment_id	COMMENTS	Primary key, used to identify and keep track of each comment.	Integer	Unique integer values where comment_id > 0	1
post_id	COMMENTS	Foreign key, used to identify and keep track of the post which the comment was made on.	Integer	integer values where post_id > 0	1
poster	COMMENTS	Contains the username who posted a comment	text	All strings	CharlieM
content	COMMENTS	Contains the content of a comment	Text	All strings	Hello my name is jim
likes	COMMENTS	Tracks the users who have liked the comment in a string separated by slashes	text	All strings	CharlieM/test/admin
dislikes	COMMENTS	Tracks the users who have disliked the comment in a string separated by slashes	text	All strings	CharlieM/test/admin
root_comment	COMMENTS	Contains the comment_id of the comment this was a reply to. Null if not a reply.	Integer	integer values where comment_id > 0, NULL	1

DFD and IPOs

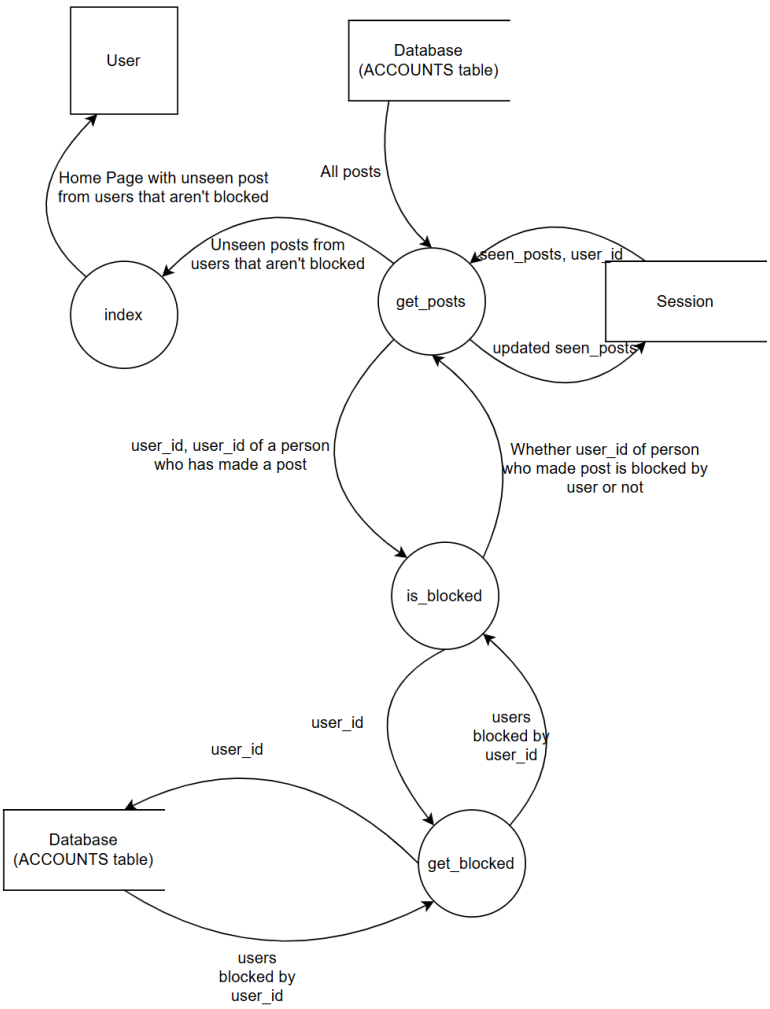
SIGN_UP DFD



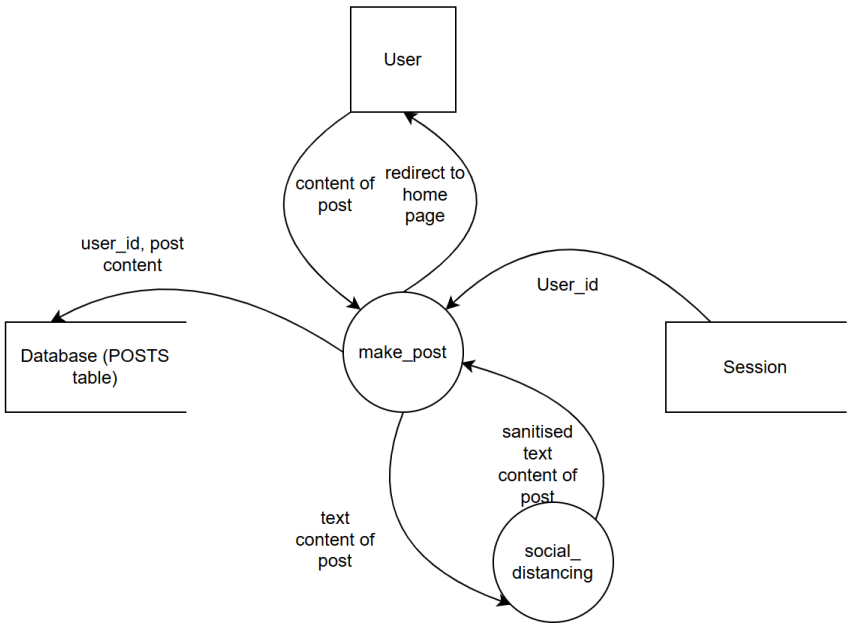
LOGIN DFD

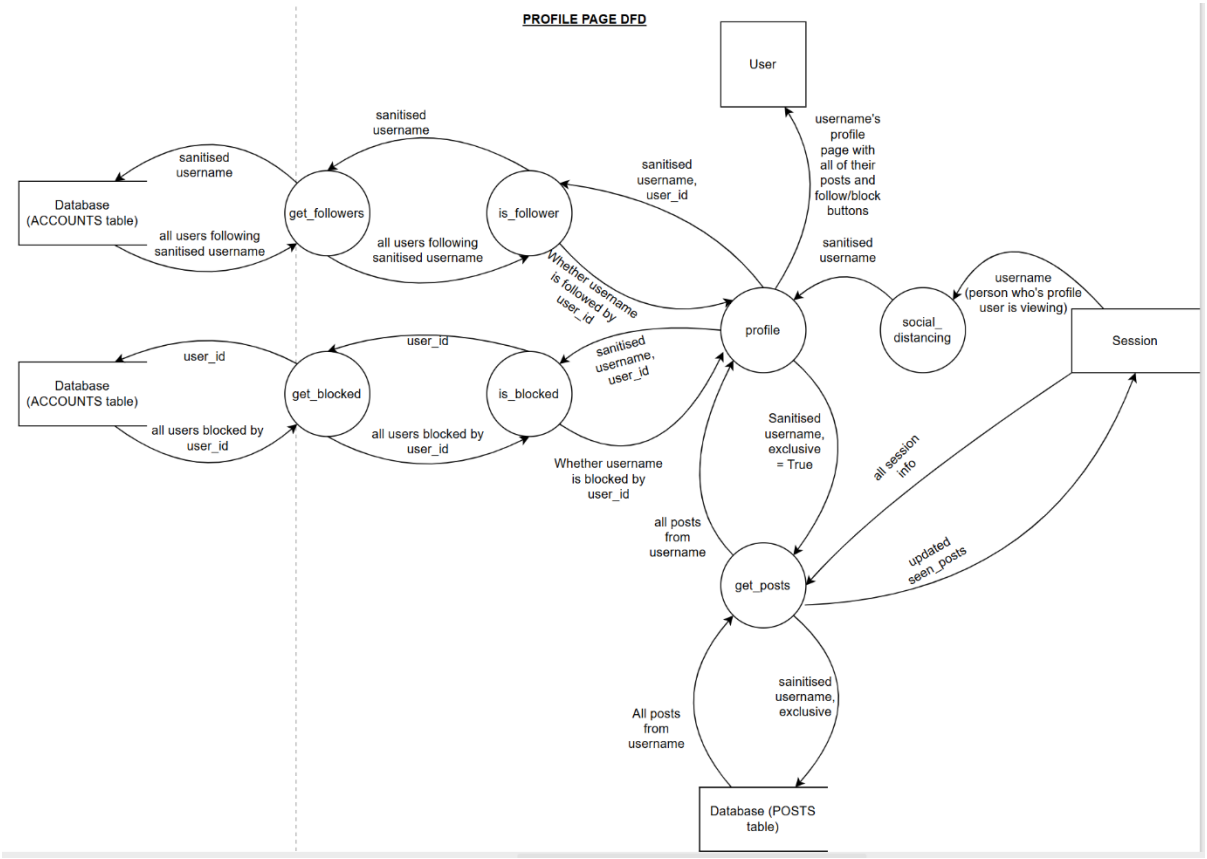


HOME PAGE DFD

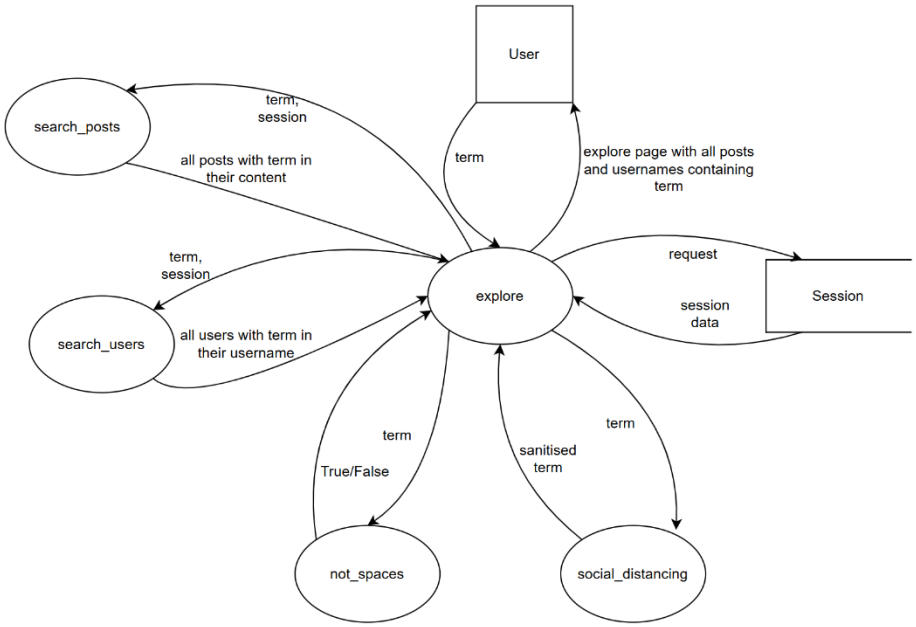


MAKE_POST DFD

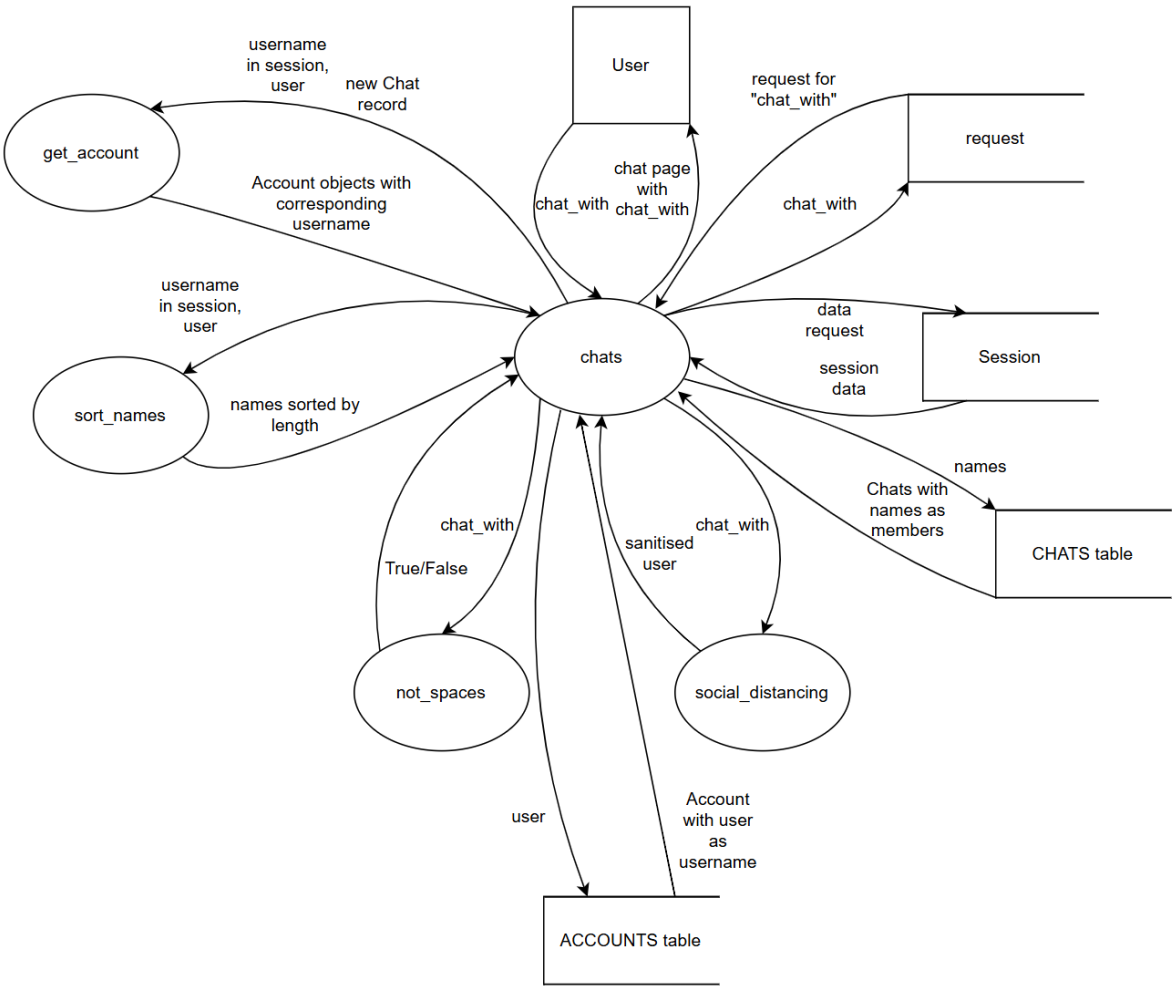


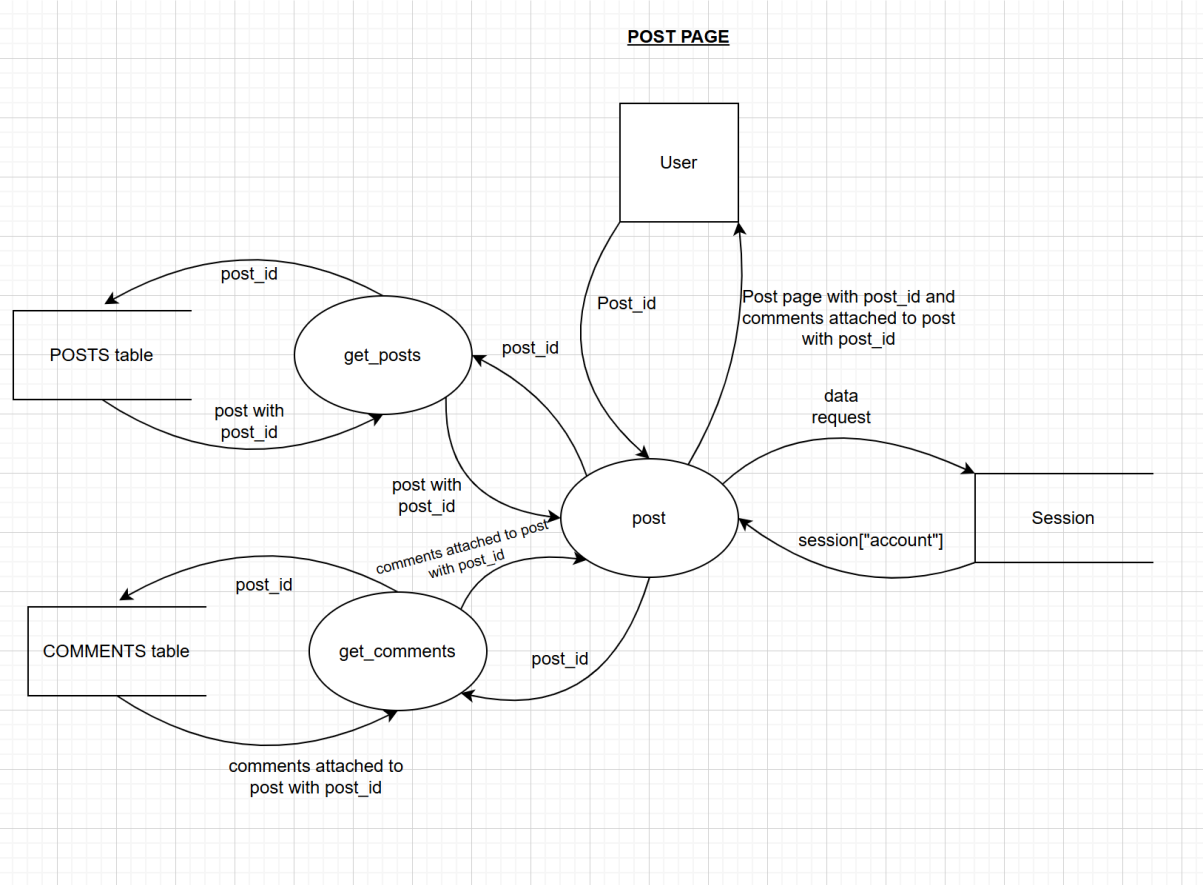

PROFILE PAGE DFD
User
username's profile page with all of their posts and follow/block buttons
sanitised username
sanitised username
Database (ACCOUNTS table)
get_followers
is_follower
sanitised username, user_id
all users following sanitised username
all users following sanitised username
Whether username is followed by user_id
sanitised username
profile
social_ distancing
username (person who's profile user is viewing)
Session
user_id
user_id
Database (ACCOUNTS table)
get_blocked
is_blocked
sanitised username, user_id
all users blocked by user_id
all users blocked by user_id
Whether username is blocked by user_id
Sanitised username, exclusive = True
all session info
all posts from username
get_posts
updated seen_posts
sainitised username, exclusive
All posts from username
Database (POSTS table)

EXPLORE PAGE DFD



CHATS PAGE DFD





IPO chart

Name	Inputs	Processes	Outputs
Is_blocked	Account object	Checks whether a user is blocked by the account executing the method	Boolean
In_chat	Account object	Checks whether an account object has access to the Chat object executing the method	Boolean
Is_follower	Account object	Checks whether a user is blocked by the account executing the method	Boolean
Sort_names	Array of names	Sorts names by length Joins items in sorted list with a '/' delineator	'/' separated string if names
Not_spaces	string	Checks if each letter is a space or not Not_empty flag set to true if non space character found	Boolean

Get_chat	Chat_id (int)	Selects record of chat with corresponding chat_id from table Creates a chat object using the record	Chat Object
Get_account	Username (string)	Selects record of account with corresponding username from table Creates an Account object using the record	Account Object
Remove_errors	Array	Removes all items with length <2 from array	array
Get_posts	Session, post_id set to ‘’	Checks if post_id is entered Selects record from posts table with post_id if so Otherwise gets all posts where the poster hasn’t blocked/been blocked by the user Converts sql records into lists Appends like count, dislike count, and comment count to each post	Posts (array of arrays)
Interaction_count	Post or comment (array), index (int)	Checks if array has any items in the provided index If so, appends liked/dislike count to end of array Otherwise appends ‘0’	Post or comment (array)
Get_comments	Post_id (int)	Selects all records from comments where post_id = post_id Converts records into lists Appends like/dislike/comment count to each list	Comments (array of arrays)
Search_posts	Term, session	Selects all records from posts with the term in the content of the post	Posts (array of arrays) or empty list

		Checks if there are any results	
Search_users	Term, session	Selects all records from users with the term in the username Checks if there are any results	Accounts (array of arrays) or empty list
Has_interacted	Thing_id, user, action, table	Gets all users who have interacted with thing Checks if user is in that list	Boolean
Convert_breaks	array	Checks if each item in array is a '\n' or '\r' If \n, replace with If \r, replace with empty string, Joins all items in list	String
Remove_from_list	Array, username	Removes given username from array. Checks how long remaining list is. If over 1 item long, returns '/'separated string. If one item, returns item. If 0 items, returns empty string.	string
Seasoned_hash_brown	Password, salt	Combines password and salt Hashes them	Hashed password+salt
Hashed_pass	password	Generates random salt Hashes combined password+salt	Hashed password+salt, salt
Check_pass	Password, salt, hashed_pass	Checks if the combined password+salt, when hashed, are the same string as hashed_pass	Boolean
Get_interactions	Thing_id, action, table	Selects record of action from table with corresponding thing_id Checks if there's a result Splits result and removes errors if so	Array of users who've interacted with thing or empty list

chat	chat_id	Attempts to make Chat object given chat_id Checks if method was GET, if the user is in the chat, and there was a chat object generated Gets content from URL Checks if there was any content Adds new message to database if so Checks if any messages are in the chat I	Displays chat page with all messages Displays chat page with no messages Redirects to chats page
Sign_up	Email, username, password	Gets record of account with username Checks email, username, and password were all filled in If so, Checks to see if username already taken If not, adds new account to database	Login page with success message Sign up page with message saying username taken Sign up page asking user to fill all fields
login	Username, password	Gets record of account with username Checks if record exists If so, make account object given username Checks password against database If match, adds username to session	Home Page Login page with message saying incorrect info Login page with message asking user to fill all fields
chats	Session, chat_with	Gets record of account with username Checks if record exists If so, make account object given username Checks if user has been blocked by account and that chat_with doesn't	New chat page Page of chat with same two members Chats page with message saying other user has blocked you

		match user's username If not blocked and names dont match, sorts users name and chat_with before searching for records of chats with those two users If no result is found, make a new chat If result is found, get info of old chat	Chats page with message saying other user doesn't exist
Change_name	Name, chat_id	Checks if name exists and isn't only spaces If so, updates name in database	Chat page with new chat name Chat page with message saying please enter new chat name
Add_member	New_username, chat_id	Gets record of chat with chat_id Checks if record of account with username exists, if so sorts names of members in chat and new_username, updates database to have new_username as a member of chat	Chat page Chat page with message saying user doesn't exist or has blocked you
Leave_chat	Chat_id	Removes you from list of members with access to chat and updates database	Chats page
New_msg_check	Msg_count, chat_id	Gets record of chat from chat_id Checks if number of messages in chat matches msg_count	JSON response (refresh) JSON response (all good)
Make_post	T_content	Sanitises t_content Converts newlines in t_content to html break tags Checks if t_content has any content If so, adds new post to database	Home page Make_post page
profile	username	Gets record of account with username Gets all posts made by account	Username's Profile page

Profile_connections	Action, recipient	Checks if 'block' is i action If so, checks if 'un' is in action If so, removes recipient from list of people who user has blocked If not, add recipient to list of people user has blocked If not, checks if follow in action If so, checks if un is in action If so, removes user from list of people following recipient If not, adds user to list of people following recipient Updates database with info	Recipient's Profile page
interactions	Action, sign, value, table, thing_id, username	Gets list of people who've interacted with thing Checks if sign == 1 If so, adds user to list Checks if sign == -1 If so, removes user from list Updates database	Confirmation message
post	Post_id	Gets record of post Gets comment on post_id	Post page
comment	Content, post_id, root_comment	Converts newline characters in content to break tags Checks if content has content that isn't just spaces If so, adds new comment to database	Post page
explore	term	Checks if term exists and isn't just spaces If so, searches for posts with term in their content and accounts with term in their username	Explore page Explore page with message "there doesn't seem to be anything here..."

refresh	Page, thing_id	Redirects user to new page	Page/thing_id

Algorithms

//get_account takes a username and returns an Account object with all data corresponding to that username.

//social_distancing sanitises input

//request.form.get is a method that takes a key and gets the value from a form submitted with the HTTP request to the Flask server

//.get_chats returns a list of all Chat objects the user has access to.

//.get_followers returns a list of all usernames who are following the Account object.

//.get_blocked returns a list of all usernames who are blocked by the Account object.

//.get_chat_id returns the chat_id of the Chat object.

//text takes a string and converts it to a textClause object which is used by sqlalchemy to access database

//.execute executes a SQL statement with the database the Connection object is linked to
//render_template takes a html page as its first input, followed by any context that the page may need to function.

//fetchone takes the first result from the output of an executed SQL statement

//.is_blocked checks if another Account object has been blocked by the Account object executing the method, returns Boolean

//.get_username returns the username of the Account object

//sort_names takes an array as input and sorts the names by length and returns them as a '/' separated string

//.get_name returns the name of a Chat object

//redirect redirects the user to a provided webpage

//url_for dynamically prepares urls given specific parameters

//Chat class takes a sql row as input and then splits it up in the __init__ to determine attributes.

```
BEGIN chats
GET session
LET chat_list = get_account(session value at key "account").get_chats()
LET follower_list = get_account(session value at key "account ").get_followers()
IF request.method = "GET" THEN
    IF chat_list THEN
        DISPLAY PAGE chats.html WITH CONTEXT chat_list, follower_list
    ELSE
        DISPLAY PAGE chats.html WITH CONTEXT follower_list
    END IF
ELSE
    IF request.method = "POST" THEN
        GET username FROM form IN request
        LET user = social_distancing(username)
        LET query = text("SELECT * FROM ACCOUNTS WHERE username='" + user + "';")
        LET result = GET first_result FROM database.db WITH query
        IF result THEN
            LET user = get_account(result)
            IF NOT user.is_blocked(get_account(session value at key "account") AND user.get_username() != session value at key "account" THEN
                LET users = sort_names([session value at key "account", user.get_username()])
                LET query = text("SELECT * FROM CHATS WHERE users='" + users + "';")
                LET result = GET data FROM database.db WITH query
                IF NOT result THEN
                    LET chat_name = session value at key "account" + ", " + user.get_username()
                    LET query = text("INSERT INTO CHATS (users, chat_name) values('" + users + "', '" + chat_name + "');")
                    EXECUTE query IN database.db
                    LET query = text("SELECT * FROM CHATS WHERE users='" + users + "';")
                    GET first_result FROM database.db WITH query
                    LET chat_id = (CREATE New Chat object with details (first_result)).get_chat_id()
                ELSE
                    LET result = CREATE New Chat object with details (result)
                    LET chat_id = result.get_chat_id()
                    LET chat_name = result.get_name()
                END IF
                DISPLAY PAGE chat/chat_id WITH CONTEXT chat_name=chat_name
            ELSE
                LET error = "This user has blocked you."
                DISPLAY PAGE chats.html WITH CONTEXT chat_list, follower_list, error
            END IF
        ELSE
            LET error = "This user doesn't exist."
            DISPLAY PAGE chats.html WITH CONTEXT chat_list, follower_list, error
        END IF
END IF
END chats
```

```
BEGIN Account.get_chats(self)
LET query = text("SELECT * FROM CHATS;")
LET chats = []
GET all_results FROM database.db WITH query
FOR i = 0 TO (length of all_results)-1
    LET new_chat = CREATE New Chat object with details (all_results[i])
    ADD new_chat TO the end of chats
    NEXT i
END FOR
LET accessible_chats = []
FOR i = 0 (length of chats)-1
    IF self.get_username() IN chats[i].get_members() THEN
        ADD chats[i] TO the end of accessible_chats
    END IF
    NEXT i
END FOR
RETURN accessible_chats
END Account.get chats
```

```
BEGIN get_comments(post_id):
LET query = text("SELECT * FROM COMMENTS WHERE post_id = '" + post_id + "';")
GET all_results FROM database.db WITH query
IF all_results AND all_results[0] THEN
    LET updated_results = []
    FOR i = 0 TO (length of all_results)-1
        LET item_array = []
        FOR j = 0 TO (length of all_results[j])-1
            ADD all_results[i][j] TO end of item_array
            NEXT j
            END FOR
        ADD item_array TO end of updated_results
        NEXT i
        END FOR
    FOR i = 0 TO (length of updated_results)-1
        LET post = updated_results[i]
        ADD number of likes TO end of post
        ADD number of dislikes TO end of post
        NEXT i
    END FOR
    RETURN updated_results
ELSE
    RETURN []
END IF
END get_comments
```

BEGIN main

IF file name = ‘main’ THEN

RUN app

END IF

END main

C: Producing and Implementing

Logbook

2/6/25:

Had meeting with teacher today. Settled on making social media app, most likely similar to Instagram. Took most of the holidays top come up with an idea so glad it got accepted. Will need to make an MVP (minimum value project?) so I don’t try and do something too elaborate and end up not completing something that I can submit before the due date. Also need to find a client but I don’t think that should be too difficult.

2/14/25:

Wrote interview questions today. Had a bit of trouble coming up with a good set of questions but ended up on a few aesthetic related but mainly function focused. Also found 2 clients. Will interview them soon.

2/20/25:

Sent interview questions to clients and got responses. Picked a color theme (black and purple) and got the styles of social media clients want: twitter/discord/Instagram. Not sure how I will combine them at the moment but will hopefully figure that out soon.

10/3/25:

Alright just had a bit of a moment of inspiration: With all the purple colors in the scheme I have currently, I want to make the app space themed. Communities will be called “galaxies”, but mostly I’m using this as a way to make choosing icons for the UI easier: discovering new galaxies will be a rocket ship, viewing your old ones will be a little solar system or galaxy icon. Not super intuitive when you pick up the app for the first time but I like it thematically so I will stick with it. Also will probably help come up with a name for it at some point in the future.

3/4:

Made gantt chart. Will help with planning stuff in the future (hopefully). Also did ERD to help when I make the database. Not much else to report on since I’ve had both assessment week and a full schedule of procrastination. Hopefully will get some work done before term is up and during

11/4:

Ok, so, finished home page storyboard, made some edits after clients reviewed, got the database working on the project (I think, schema is working at the very least). Name idea: Galaxi.

1/5:

Kind of forgot to do work over the holidays but back on the grind now frfr ong 🔥 🔥 🔥 🔥 🔥 🔥 💯 💯 💯 💯

Anyway, made a sanitization function today as well as a login wrapper which I struggled with previously, but stack overflow is simply goated (*with the sauce*) and I

managed to figure it out. Am I procrastinating? Maybe, but its still working on the project so it’s a win for me. Also made a register page extended from the login page.

5/5:

Alright forgot to do this when I made the database but heres the schema for the database in case future me needs it:

```
CREATE TABLE IF NOT EXISTS "Accounts" (
"username"TEXT UNIQUE,
"password"TEXT,
"email"TEXT,
“salt”TEXT,
"friends"TEXT,
"followers"TEXT,
PRIMARY KEY("username")
);
CREATE TABLE IF NOT EXISTS "Chats" (
"chat_id"INTEGER UNIQUE,
"users"TEXT,
"chat_name"TEXT,
"messages"TEXT,
PRIMARY KEY("chat_id")
);
CREATE TABLE IF NOT EXISTS "Comments" (
"comment_id"INTEGER UNIQUE,
"post_id"INTEGER,
"poster"TEXT,
"content"TEXT,
```

"Field5di"INTEGER,

"Field6"INTEGER,

"root_comment"INTEGER,

PRIMARY KEY("comment_id")

);

CREATE TABLE IF NOT EXISTS "Messages" (

"message_id"INTEGER UNIQUE,

"content"TEXT,

"sender"TEXT,

"chat_id"TEXT,

PRIMARY KEY("message_id")

);

CREATE TABLE IF NOT EXISTS "Posts" (

"post_id"INTEGER UNIQUE,

"poster"TEXT,

"content"TEXT,

"dislikes"INTEGER,

"comments"TEXT,

"likes"INTEGER,

PRIMARY KEY("post_id")

);

Note: just added salt to the database table because I forgot to do that when I originally made it

Anyway, added a hashing function using the hashlib library combined with a salt generator to hash passwords. Finished the register page functionality.

Add a “seasoned_hash_brown” function that returns a hashed password and salt, “check_password” function that returns True if a password and salt hashed matches a given string, and finished the login route functionality

Friend/Follow requests: Just realised I had no idea how to handle this. But gameplan! Make a friend request a special type of message with a button on it that makes the sender and recipient friends. Only allow one friend request to be sent from a specific person to another specific person at a time.

15/5:

Alr so did some work yesterday as well but forgor to write it down anyway so decided to instead of requiring being friends to start a chat you just need to be not blocked by them. Makes it more convenient for users and also is way easier for me to implement. Finished the templates for the ‘chats’ page and the ‘chat’ page (chats page allows you to make more chats and view old ones, chat page shows you messages). Had a lot of syntax errors in both python and sql but fixed those pretty quick

Friend request work idea: content = recipient 🔥 🔥 🔥 🔥 🔥 🔥

19/5:

Low motiv. Made it so can’t access chats that you are not a part of.

29/5:

Has been 10 days since last logbook entry but NOT since last time worked on. Just forgot abt this because tspmoicl. Anyway so added posts, which was surprisingly easy, got the font awesome stuff imported which is good because working with images is awful and the worst (I will have to do this later for images in posts), logout button works, started some minor css just to make the posts look a lil better and they look decent but will likely make it a grid. Literally just now discovering I have a friends.html template that I don’t have a route for? Guess I wanted to do that at some point. Had an issue with chats where the sanitised strings were not displayed properly to the browser, which I figured out was because jinja automatically sanitises strings for you which seems helpful but really isn’t since it sanitises semicolons which are used in *all* html codes, making them display incorrectly. Fixed it by looking it up and finding out you can add a ‘| safe’ to a jinja variable to make it not sanitise the string. Also, the chat page used to ask you if you were sure you wanted to resubmit the form everytime you reloaded the page, which was very annoying. Fixed this by making the form transfer info via a get request

rather than a post request (stack overflow is simply the root of all human knowledge is the main thing ive gathered from this project.)

What I'm planning on doing next:

- Get teacher to help with js (I desperately need this)
- Make profile pages to see people posts, friends, mutuals, liked posts, etc
- Get comments actually working (unsure if I will make a page dedicated to a post that opens when you click on it or just have it sort of insta-esque where it pulls up comments)
- Maybe include some machine learning somewhere?
- End to end encryption for messages (having messages stored in the database plain text isn't exactly secure because I can read them which is unoptimal)

30/5

Alright so actually did the js by myself, found some code online to help but troubleshooted it myself. Surprisingly hard because of how quotes and stuff work in html and it wasn't fun but now I can actually continue with blocking/following

3/6

Weekend was good but back on the grind

Fixed the really annoying chat problem where you needed to reload the page once to see your message appear. Still need to reload to see other people's messages but eh whatever. Fixed it by making a 'refresh' route that you redirect to that immediately just redirects you back to the original route. Also, now once you block someone, they can't make a chat with you. Peak.

5/6:

Css is humanity's worst invention by far but I managed to get the css for the nav bar working, though not sure how it will perform on displays of different scales. Hopefully still good.

12/6:

Javascript is a lot easier than I initially thought especially since a lot of it is almost like python but a teeny bit different. Got blocked people's messages now auto displaying as "blocked message" in chats with them and will soon prevent their posts from showing.

4/7:

Alright so today I decided to do some work and yesterday I did some as well on the same thing which was getting likes and dislikes working. You'd think that'd be an nice easy task to get done. 3 hours. The jinja didn't want to work, and just vented to friend abt the numerous errors I encountered today. 1. Have a '/' separated string of people who've liked each post, but not an actual count stored within database so added that in python since it was easier. 2. Since I need to split the string, and python gets real angry if you try to split an empty string, use an if check to check if the string contains anything. Forget to add an else that appends a 0 instead of jinja just deciding that 'index doesn't exist so will display nothing'. 3. When splitting the string the way my format works is that it adds a '/(username)' to the list of ppl each time a new person likes it. For some unknown reason, when using .split, python goes "HMMMMMM lets see we have the string '/user'. I think this should be a list containing ['', 'user']." WHY WOULD ANYONE EVER USE IT LIKE THIS. Realised I already encountered this problem earlier and had already built a function to get rid of the extra ones for me. Finally got it working.

25/7

SO

AS IT TURNS OUT

Classes are required for this assessment. Having written my entire code thus far in the functional paradigm instead of the object oriented one, this poses a significant issue. Or, it did. I just spent around 2 hours converting chats, accounts, and messages to classes as well as debugging. It did make my code significantly cleaner all things considered, however it was a hassle I'd rather not have gone through. Main issue wasn't any logic issues but rather the fact that you can't store objects in the session. This was particularly annoying for me after i went through my code, replacing all instances of session["user_id"] with session["account"] and storing an account object in it. Had to effectively switch back after that.

28/7

Everything's working. Explore page, dms with multiple people, leaving gcs, everything. BUT. Css is awful and the worst thing in the world. Managed to get some semblance of consistency by basically just slapping everything with a 'center' class that added some padding and put things to the center of the page. @medis took a bit to understand but i managed to get through it and so the UI actually semi-works on phones now. OR, at least, my phone.

User Interface(UI)

Programming

Database

D: Testing and Evaluating

Test Plan

Test input	Function/module	Expected output	Actual output	Why this test
“”	chat	nothing	nothing	If user were to accidentally click enter while their message bar was empty, they shouldn’t send a message.
“ “	chat	nothing	nothing	Sending nothing doesn’t make sense.
“’;--”	Chat, social_distancing	Message with content “’;--” displayed	Message with content “’;--” displayed	Sanitisation check. Need to make sure input is being sanitised.
“”	Make_post	nothing	nothing	If user were to accidentally click enter while their post was empty, they shouldn’t post nothing.
“ “	Make_post	nothing	nothing	Posting nothing doesn’t make sense.
“’;--”	Make_post, social_distancing	Post with content “’;--” displayed	Post with content “’;--” displayed	Sanitisation check. Need to make sure input is being sanitised.
“”	comment	nothing	nothing	If user were to accidentally click enter while their comment bar was empty, they shouldn’t post an empty comment.
“ “	comment	nothing	nothing	commenting nothing doesn’t make sense.
“’;--”	Comment, social_distancing	comment with content “’;--” displayed	comment with content “’;--” displayed	Sanitisation check. Need to make sure input is being sanitised.
“”	explore	nothing	nothing	searching nothing doesn’t make sense.
“ “	explore	nothing	nothing	searching nothing doesn’t make sense.

“’;--”	explore, social_distancing	Posts and usernames with with “’;--” in the content/usernames respectively will be displayed	Post with “’;--” was displayed	Sanitisation check. Need to make sure input is being sanitised.
“”	chats	Message “user doesn’t exist” should be displayed	Message “user doesn’t exist” displayed	Trying to make a chat with noone shouldn’t work
“ ”	chats	Message “user doesn’t exist” should be displayed	Message “user doesn’t exist” displayed	Trying to make a chat with someone who doesn’t exist shouldn’t work
“test”	chats	Message “User has blocked you” should be displayed	Message “User has blocked you” displayed	On the test account i blocked my main account, and then tried to make a chat with it on the main account. You shouldn’t be able to make chats with people who’ve blocked you.

User Accessibility Testing

Test	*Client 1 satisfaction (1-10)*	*Feeback*	*Client 2 satisfaction (1-10)*	*Feeback*
Registering an account	*8*	*Should start on register page, otherwise good*	*9*	
Logging in	*7*	*Why do i need to log in after registering?*	*9*	
Navbar	*9*		*8*	*Had to experiment to find out what each icon meant, but good nonetheless*
Making a post	*10*	*Easy to use.*	*10*	
Commenting	*9*		*7*	*Hard to click the small icon*
Using Explore page to search	*10*		*10*	

Viewing a profile	*8*	*Why does the name not look blue?*	*7*	*Annoying to click on names*
Following/Blocking	*10*		*10*	
Making a private chat	*9*		*6*	*Would like to add more than one person at a time*
Messaging	*9*		*9*	
Leaving a private chat	*10*		*10*	
Adding members to a private chat	*9*		*7*	*See feedbsck for making a private chat*
Liking/disliking posts/comments	*10*		*8*	*Small icons D:*

Overall, clients reported that while clearly still in its infancy, the product was useful and easy to learn, had a clean experience, and would be happy for it to be deployed.

For accessibility issues, i focused on people with colour blindness. The *Colour-Blind Awareness* organisation states that Deutanaropia is the most common form of colour blindness, affecting around 8% of men and 0.5% of women worldwide. This is a deficiency in the green cones in the eye, leading to confusion between mid-reds and greens, blue-greens and mid pinks, light blue with lilacs, and a few more. To make the website more accessible for people afflicted with this condition, I specifically avoided using colours commonly confused with each other in my web design. While pinks and purples are prominent, I made sure to not feature any blue-greens or light blues which could lead to confusion. Furthermore, to assist those with severe color blindness such as monochromacy, or simply people with significant visual impairments, I made heavy use of named divs within my HTML to make my site more parseable for screen readers. For example, my navbar is contained within a *nav* tag, and all my posts are contained within *articles.*

E: Viva Voce

On the day of submission (or the next day you are at school if you have a doctor's certificate) there will be an exhibition of Year 12 Major Works and you will have a Viva Voce with your teacher. You should be able to undergo questioning about your project and the decisions you made.

Candidates will be judged on how well they demonstrate:

- A technical focus
- An understanding of the project and their client's needs
- An understanding of the different web technologies and the relationship between them
- Relevant examples from the code that support the discussion

References:

https://stackoverflow.com/questions/32054891/python-input-sanitization https://mateam.net/html-escape-characters/
https://stackoverflow.com/questions/20503183/python-flask-working-with-wraps https://www.w3schools.com/tags/tag_datalist.asp
https://www.w3schools.com/python/ref_list_sort.asp https://webmasters.stackexchange.com/questions/131460/how-do-you-prevent-html-special-character-coding-from-displaying-on-a-website https://stackoverflow.com/questions/57169670/stop-jinja-from-converting-string-to-url-encoding
https://webmasters.stackexchange.com/questions/131460/how-do-you-prevent-html-special-character-coding-from-displaying-on-a-website
https://stackoverflow.com/questions/6833914/how-to-prevent-the-confirm-form-resubmission-dialog https://medium.com/@brodiea19/flask-sqlalchemy-how-to-upload-photos-and-render-them-to-your-webpage-84aa549ab39e ^^^^^ file upload managing for flask (uses oop and some libraries)
https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch https://www.w3schools.com/jsref/event_onclick.asp
https://stackoverflow.com/questions/7562095/redirect-on-select-option-in-select-box https://stackoverflow.com/questions/5303899/change-onclick-action-with-a-javascript-function https://developer.mozilla.org/en-US/docs/Web/API/Element/children https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Indexed_collections https://www.w3schools.com/js/js_if_else.asp https://hostman.com/tutorials/how-to-create-scrolling-on-your-website-using-css/ https://testdriven.io/courses/learn-flask/sessions/#:~:text=Sessions%20in%20Flask,-In%20Flask%2C%20you&text=How%20are%20sessions%20implemented%20in,server%2Dside%20where%20it's%20decoded.
https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Forms/Sending_and_retrieving_form_data
https://www.w3schools.com/sql/sql_count.asp https://stackoverflow.com/questions/2876789/how-can-i-search-case-insensitive-in-a-column-using-like-wildcard

https://piktochart.com/tips/black-purple-color-palette

https://www.colourblindawareness.org/colour-blindness/types-of-colour-blindness/