Social Content Sharing Platform

Sayan Ghosh — 20CS1A3141
Aasif Khan — 20CS1A3121

Synopsis:

Most modern social media platforms are usually owned and operated by big corporations. These platforms often employ algorithms to rank content favourable for their advertisement partners but not their users. This kind of model is called an advertising-centric product model. These kinds of products are not palatable for normal users as they are not in control over the content they are seeing. Some platforms provide a false sense of control by employing features like ‘add friend’ or ‘subscribe’, but they still don't stick purely to the subscription model. This leads to customer dissatisfaction. In an audience-centric product, the user should have total control over the content they see. This not only includes which posts are shown but also their order of appearance. Thus to determine the order of the posts, some metric needs to be maintained, which is audience-centric. Making this unique for each user would be harder as the user would have to rate multitudes of posts. The most practical solution for this becomes a community-centric algorithm. This uses other users’ explicit expression of their liking/disliking of a post to construct a sortable metric. The algorithm also needs to take into account the age of the post, so that newer posts are given a chance to get exposure instead of being buried under popular but old posts. This can be done using the ‘hot ranking algorithm’ for the posts so that newer posts are given a higher ranking than older posts, all the while also using the like/dislike counts to assign the score. The comment section of the posts however needs to have a different algorithm so that it does not depend on the age of the comment, since better comments should always be present on the top. This can be achieved by using the ‘confidence sort’ using the lower bound of the ‘Wilson score interval’. This is similar to social platform ‘Reddit’ but differs from it as it uses a subscription-based model of users following other users instead of sourcing through the entire pool of users. Our project deals with these shortcomings and provides a rich, user-friendly, and audience-centric platform that lets users be in-charge of the posts and comments they see.

System Study:

Existing System:

Currently multitudes of social media platforms exist and listing all of them would not be practically possible. However, most of the products like Facebook, Instagram, Linkedin, etc focus on proprietary algorithms that favour advertisers over users. These algorithms are made to keep the users on the app for longer periods of time and/or to make them see advertisements and click on them. These platforms do not respect the user’s time and energy, hence the need

of a audience-centric platform. Platforms like Reddit, 4chan, etc are more user-centric but are not social platforms per-se. These platforms either show posts from all users across the globe, or certain sandboxed community pages, which only pertain to a specific topic/thing/person/place. This does not allow the user to follow other users and see only posts made by them, losing the social aspect. These shortcomings are addressed by combining the follower-followee model and the merit-based ranking model.

Problems in Existing System:

1. The algorithm for ranking is not audience-centric.
2. Users cannot subscribe to individual users.
3. Unpredictable behaviour of the platform.

Proposed System:

The proposed system combines the advantages of existing solutions while not incorporating its shortcomings. The project uses a merit-based ranking system with/without considering the age of the item. This does not incorporate any external variables in the ranking of posts, such as advertiser interest, as the platform does not run on ads. The solution utilises the ‘hot ranking algorithm’ for ranking posts using their submission date and time as well as their like and dislike counts as metrics. This causes posts to ‘age’ naturally and slowly rank lower than newer posts. Comments on the platform utilise the ‘confidence sort’ which utilises the ‘Wilson score interval’ to statistically test the hypothesis of how good the comment is, based on the like-dislike data available. This is better than using an absolute score or absolute average score directly from the like-dislike counts. This prevents conflicting situations like posts with one like and zero dislikes being ranked higher than posts with hundreds of likes and one dislike. This causes a more organic flow of comments which are worthy of being shown at the top. The system utilises the ‘subscribe’ model to filter contents to show. This lets the user be in complete control of what they see and what they don’t. Users only see content from people they explicitly ‘follow’.

Advantages over Existing System:

1. Ranks content systematically based on their merit and age.
2. Lets users choose their circle and author of the posts they see.
3. Incentivises content creation and engaging in liking/disliking of other’s content to improve the community.
4. Incorporates transparency in content ranking algorithm.

Module Description:

1. **Discover** - View content shared by peers whom you follow. Content is sorted so that newer and better content is shown first.
2. **Messaging** - Chat with your peers or anyone else in the world. Chats are asynchronous and stored on a centralised server.

3. **User Account** - View / Change account details like name, bio, dp. Users can only change their own details post authentication.
4. **Content Upload and Post** - Let users upload and share content. Store the content in the database and let it be accessible to all other users who follow him.
5. **Notification Centre** - View and open notifications received by a user pertaining to his account and posts. Get notified about peers commenting on your posts.
6. **Settings** - Change site settings on a per-user basis. Toggle between light and dark mode. Clear cache and stored cookies.
7. **Donate -** Support the application with donation. Donations are optional and do not affect any functionality of the application.
8. **Admin Dashboard** - Show statistics and data about users, registrations, posts, comments, and other similar insights about the platform.

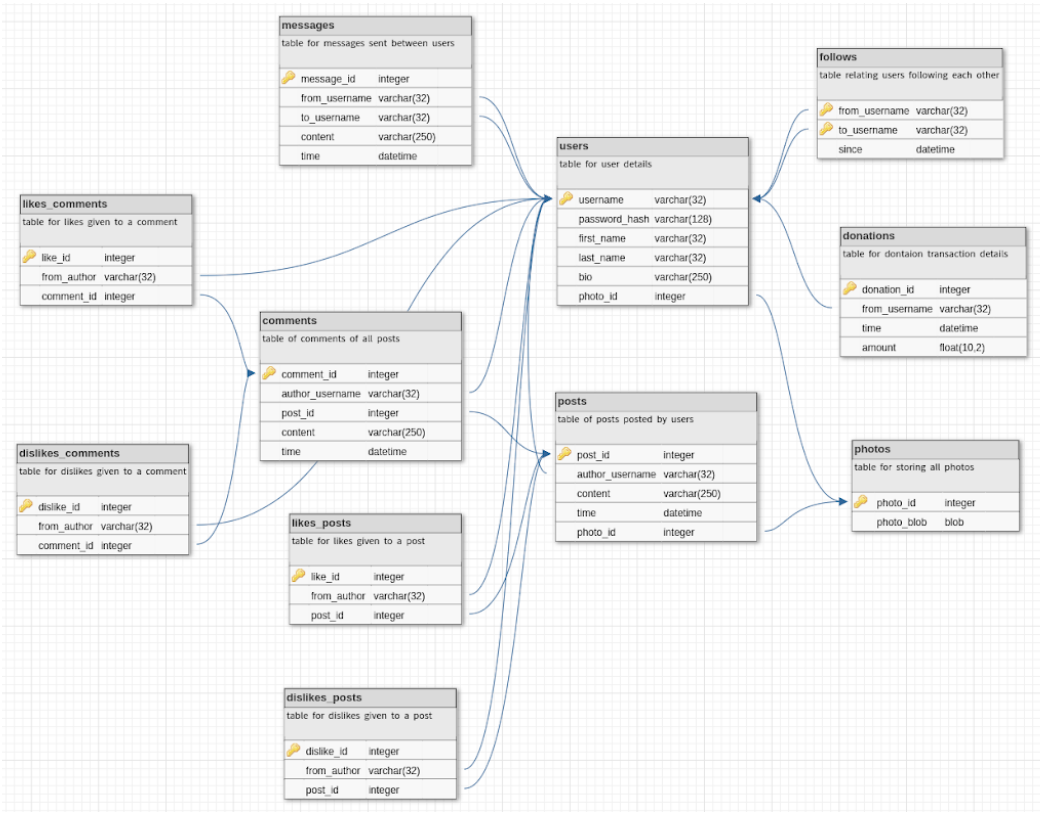
Database Design:

Table Name	Attribute	Data type	Constraints	Description
users	username	varchar(32)	primary key, unique, not null	unique username of each user
	password_hash	varchar(128)	not null	hash of username+password
	first_name	varchar(32)	not null	first name of user
	last_name	varchar(32)		last name of user
	bio	varchar(250)		user's short biography
	photo_id	int	references photos.photo_id, unique	reference id of photo of user (display picture)
posts	post_id	int	primary key, unique, not null, autoincrement	unique id of each post
	author_username	varchar(32)	references users.username	username of user who posted the post
	content	varchar(250)	not null	text part of post
	time	datetime	not null	time of posting
	photo_id	int	references photos.photo_id	photo of post (if any)
follows	from_username	varchar(32)	primary key, not null, unique, references user.username	username of person who follows
	to_username	varchar(32)		username of person whom he follows
	since	date	not null	time from when he follows

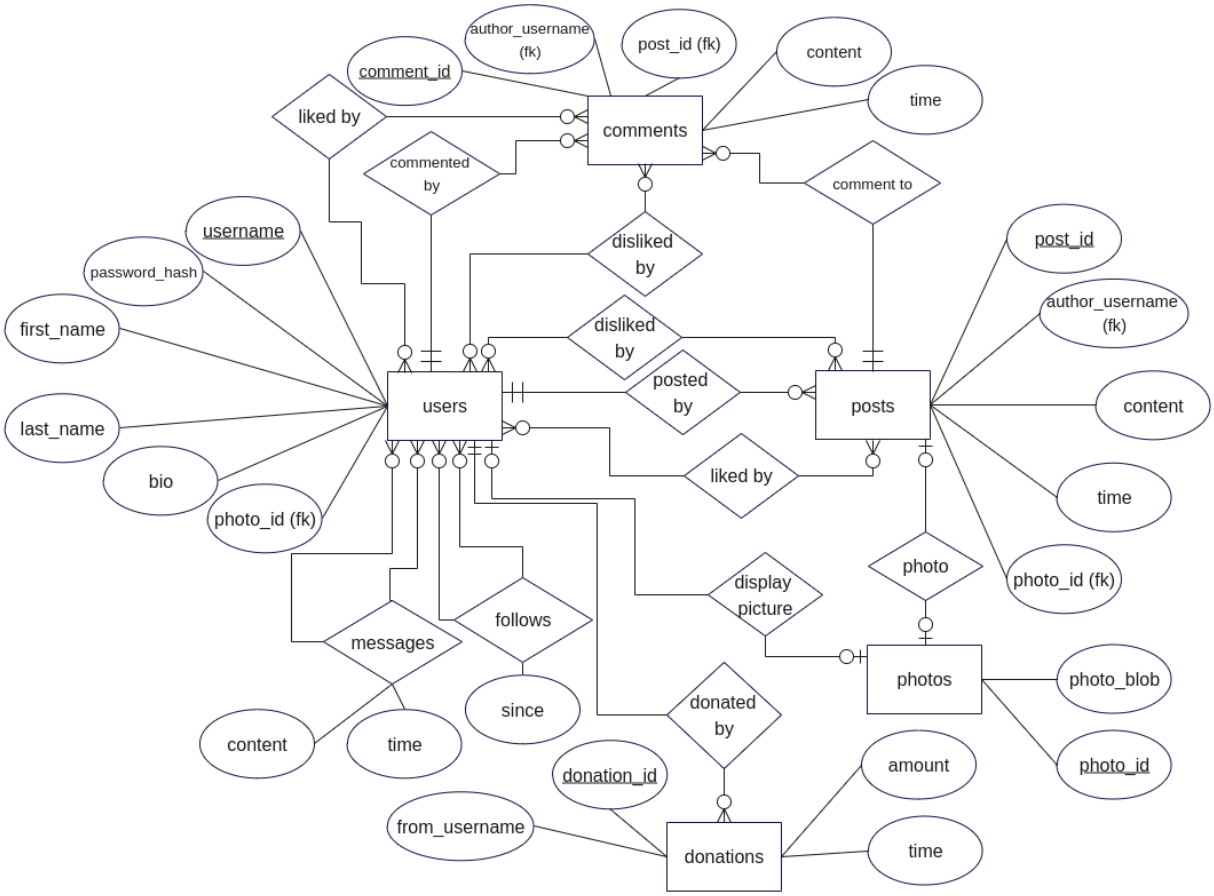
comments	comment_id	int	primary key, unique, not null, autoincrement	unique id of each comment
	author_username	varchar(32)	references users.username, not null	username of user who posted this comment
	post_id	int	references posts.post_id, not null	unique id of post to which comment belongs
	content	varchar(250)	not null	text content of comment
	time	datetime	not null	time of posting of comment
messages	message_id	int	primary key, unique, not null, autoincrement	unique id of each chat
	from_username	varchar(32)	references users.username, not null	username of user who sends message
	to_username	varchar(32)	references users.username, not null	username of user who recieve message
	content	varchar(250)	not null	text content of message
	time	datetime	not null	date of message sent
likes_posts	like_id	int	primary key, unique, not null, autoincrement	unique id of the like
	from_author	varchar(32)	references users.username	username of user who liked the post
	post_id	int	references posts.posts_id	unique id of the post liked
dislikes_posts	dislike_id	int	primary key, unique, not null, autoincrement	unique id of the dislike
	from_author	varchar(32)	references users.username	username of user who disliked the post
	post_id	int	references posts.posts_id	unique id of the post disliked
likes_comments	like_id	int	primary key, unique, not null, autoincrement	unique id of the like
	from_author	varchar(32)	references users.username	username of user who liked the comment
	comment_id	int	references comments.comment_id	unique id of the comment liked
dislikes_comments	dislike_id	int	primary key, unique, not null, autoincrement	unique id of the dislike
	from_author	varchar(32)	references users.username	username of user who disliked the comment

	comment_id	int	references comments.comment_id	unique id of the comment disliked
photos	photo_id	int	primary key, unique, not null, autoincrement	unique id of each photo stored in database
	photo_blob	blob	not null	binary large object storing photo data
donations	donation_id	int	primary key, unique, not null, autoincrement	unique id of each donation transaction
	from_username	varchar(32)	references users.username	username of user who donated
	time	datetime	not null	time of transaction
	amount	numeric(10,2)	not null	amount of money donated

Schema:

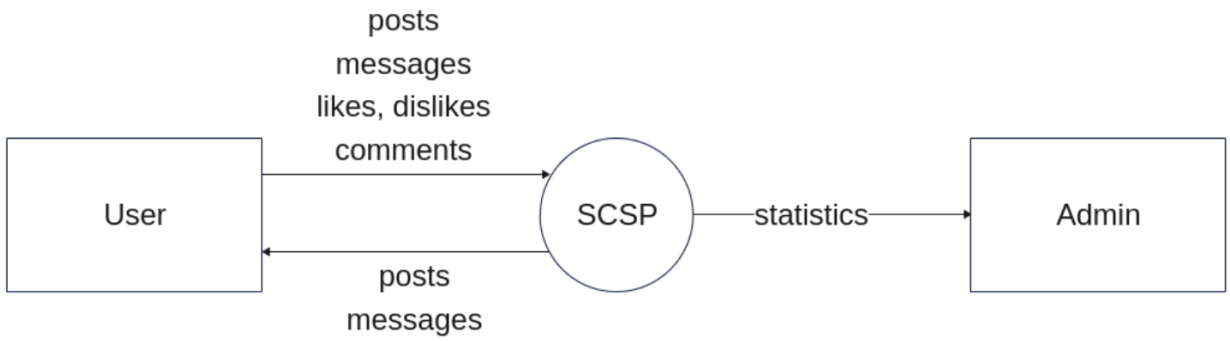


ER Diagram:

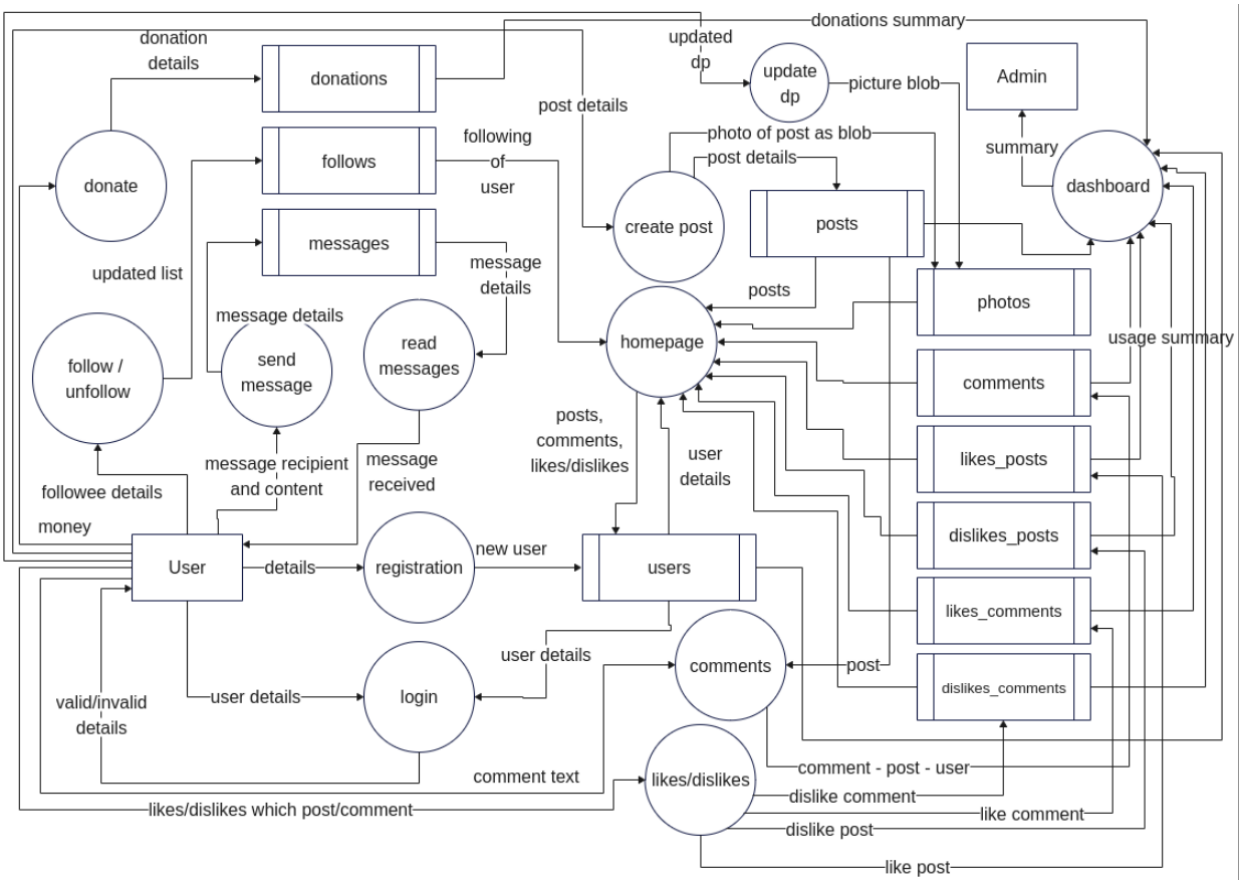

author_username (fk)
post_id (fk)
content
comment_id
time
liked by
comments
commented by
comment to
username
disliked by
post_id
password_hash
author_username (fk)
first_name
disliked by
posted by
users
posts
content
last_name
bio
liked by
time
photo_id (fk)
photo
photo_id (fk)
display picture
follows
messages
photos
photo_blob
since
donated by
content
time
donation_id
amount
photo_id
from_username
donations
time

Data Flow Diagram:

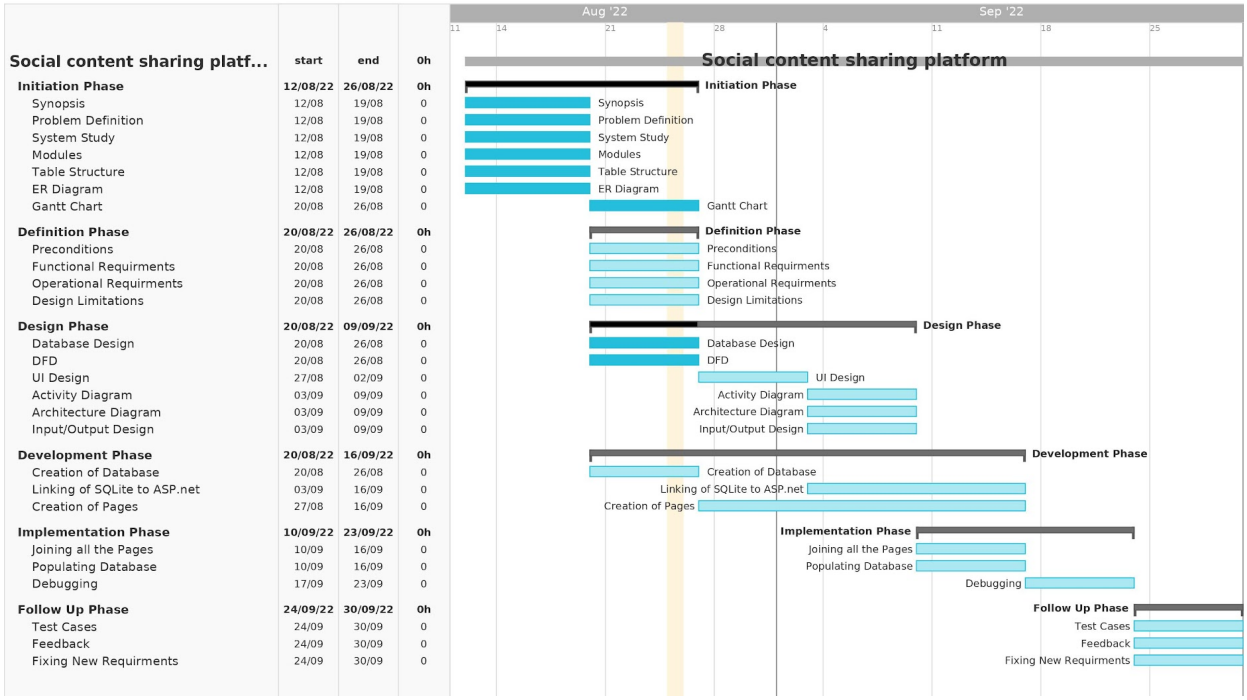
Level 0:



Level 1:



Gantt Chart:


Aug '22
Sep '22
11
14
21
28
4
11
18
25
Social content sharing platf...
start
end
0h
Social content sharing platform
Initiation Phase 12/08/22 26/08/22 0h
Synopsis 12/08 19/08 0
Problem Definition 12/08 19/08 0
System Study 12/08 19/08 0
Modules 12/08 19/08 0
Table Structure 12/08 19/08 0
ER Diagram 12/08 19/08 0
Gantt Chart 20/08 26/08 0
Definition Phase 20/08/22 26/08/22 0h
Preconditions 20/08 26/08 0
Functional Requirments 20/08 26/08 0
Operational Requirments 20/08 26/08 0
Design Limitations 20/08 26/08 0
Design Phase 20/08/22 09/09/22 0h
Database Design 20/08 26/08 0
DFD 20/08 26/08 0
UI Design 27/08 02/09 0
Activity Diagram 03/09 09/09 0
Architecture Diagram 03/09 09/09 0
Input/Output Design 03/09 09/09 0
Development Phase 20/08/22 16/09/22 0h
Creation of Database 20/08 26/08 0
Linking of SQLite to ASP.net 03/09 16/09 0
Creation of Pages 27/08 16/09 0
Implementation Phase 10/09/22 23/09/22 0h
Joining all the Pages 10/09 16/09 0
Populating Database 10/09 16/09 0
Debugging 17/09 23/09 0
Follow Up Phase 24/09/22 30/09/22 0h
Test Cases 24/09 30/09 0
Feedback 24/09 30/09 0
Fixing New Requirments 24/09 30/09 0