Final Report

Team Details:

Kanad Pardeshi	: 190050056
Manan Agarwal	: 190050065
Tanu Goyal	: 190050123
Dhruva Dhingra	: 190070020

What we Implemented:

We were able to develop our Insti-Gram website with the following features:

Login: Any existing user can log in using their credentials (email and password)

Signup: Any new user can make an account on Instigram, by clicking on the Sign Up button and filling up their personal details

Once logged in, a user can use the Navigation Bar to access the following features:

Encrypted Password: The password is never stored in plaintext form in the database and nor is it transmitted in plaintext form from the frontend to the backend. Only the encrypted form is stored in the database.

Homepage: On the Homepage, users can see the posts from their friends. Each post shows the content, the user that posted it, and the number of likes. Users can use the Like and Dislike button to react to the post.

Timeline: On a friend's Timeline, users can see their details, like Residence and Birthday, and their posts, sorted by the time of posting. Again, users can use the Like and Unlike button to react to the post. Users can send friend requests to different users through Send Button on the Timeline

Both Timeline and Homepage have Next and Back buttons to scroll through the content.
The like and unlike buttons, on clicking, update the total-likes counter in real-time.

Search: Users can use the Search Button to search the social network and make new friends. He/She can put in any number of space-separated keywords, and any name with the substring matching the given set is returned from the dataset. Users can click on the results to visit these timelines.

Create Post: Users can click on create a post, to open up a form. Here he/she can type the content to share a post that would be seen on the Homepages of his/her friends. Timeline of the User creating the post will get updated.

Messenger: Users can click on Messenger to open up the messaging interface. On the Base Page, the user can see all his friends with their last message, sorted according to the time they last interacted.

Personal Chat: On clicking on a user in the Messenger App, the user is taken to the Personal Chat with that friend. Here all messages have been sorted, and displayed in different colours, depending on whether it was sent or received. Users can also use the Send Button, to share a new message over the Messenger.

Messages shared across the personal chat are also updated in real-time.

Friend Requests: Users can click on the Friend Requests tab to open up another page, showing all the Friend requests that he/she has received over the Instigram. Here, they can decide whether to accept the invite or decline it. The page also shows Friendship Recommendation, using the neo4j server. This is done on the basis of the following algorithm:

Recommendation Algorithm: For a given user, we find the nodes that are separated by two degrees from the user, that is Friend of Friend, with undirected edges, and calculate their total number of In and Outgoing edges, to gauge their popularity. Out of these, we bring the top 15 nodes as recommendations, as well as 5 more nodes that are randomly picked, to avoid making the graph stagnant.

Analytics: Users can click on the Analytics tab, to see various analytical graphs. This includes a graph showing the frequency of users with respect to the number of friends, from which one can see that it indeed seems to follow a power-law distribution. There are also graphs showing the relationship between the number of average likes on a post with respect to the number of friends, and the Hourly/ Weekday/ Daily frequency of posts. Daily frequency seems to increase as more and more users join Instigram.

Edit Profile: Here the user can edit their profile information such as Batch, Branch, Residence, Birthday and Profile Picture.

About: Here the user can see their own personal details and their Profile Picture.

Logout: Finally the user can click on the Logout button to log out from the Instigram Website.

Dataset Generation:

For the purpose of testing and implementation, we prepared the following datasets:

Users: We picked up a dataset of authentic Indian names, and using a python script selected other fields randomly from the set of plausible values. For example, the branch was assigned a value randomly from the set {EE, CSE, ME…}.

Posts: For posts, we picked up a Twitter dataset, and timestamps were picked randomly, taking the signup date of the user who posted, into account.

Friends: The dataset for Friends was created using the network edges in a Facebook dataset, and mapping them to our existing users. Due to this, our data mimics the real-life properties of such social networks like the Power Law.

Message: For generating the messaging dataset, we referred to the Human-to-Human Conversation dataset from Alexa, and picked up random conversations to be mapped to pair of friends.

Reactions: For generating reactions, we assumed a Bernoulli distribution for each friend to like the posts. This, therefore, gives us data having a correlation between the number of likes and the number of friends.

Profile Picture: This data was generated by assigning our own set of photos, to different users.

Load Testing Results

The load testing was done with the approximate size of datasets as follows:

1) Users	: 4,000
2) Posts	: 15,000
3) Friends	: 90,000
4) Messages	: 80,000
5) Reactions	: 80,000
6) Profile Picture	: 20

Test Results

The following testing on Negative and Positive Test Cases was carried out.

Sr No	Use Case	Positive Test	Negative Test
1	Signup/Login	1. Valid signup 2. Valid login	1. Signup with pre-existing info 2. Invalid login a. Wrong username b. Wrong password
2.	Homepage	1. Basic test of homepage with posts	1. Checking homepage without posts
3.	Friend Requests / Recommendations	1. Checking display recommendations	None
4	View User Page (Includes sending requests)	1. Checking if appropriate user/page (with their posts) is being displayed 2. Sending request to user/page	1. Checking page of user/page which is already a friend/being followed (Sending request to an already existing friend) 2. Trying to view your own user page, should redirect to the user timeline.
5	Create Post as a User	1. Creating valid post	1. Trying to create a post with some missing data. This should not be created, and a prompt should show up instead
6	React on post	1. Choosing reaction among those available 2. Checking reversibility of reactions	1. User reacting on their own post shouldn't be possible
7	View Messages	1. Checking if latest messages are visible in chronological order	None
8	Send Message	1. Checking if sent message is visible to receiver, for both individual and group messages	1. Check that empty message is not sent

Requirements

Following libraries/ tools were used for implementing InstiGram:

1) NodeJS
2) React
3) Neo4j
4) Python 3

Conclusions

We were successfully able to implement our own social network website- Instigram, with many essential features, such as Messenger, Creating and Liking Posts, Visiting Homepage and Timelines.

What you were not able to do and why:

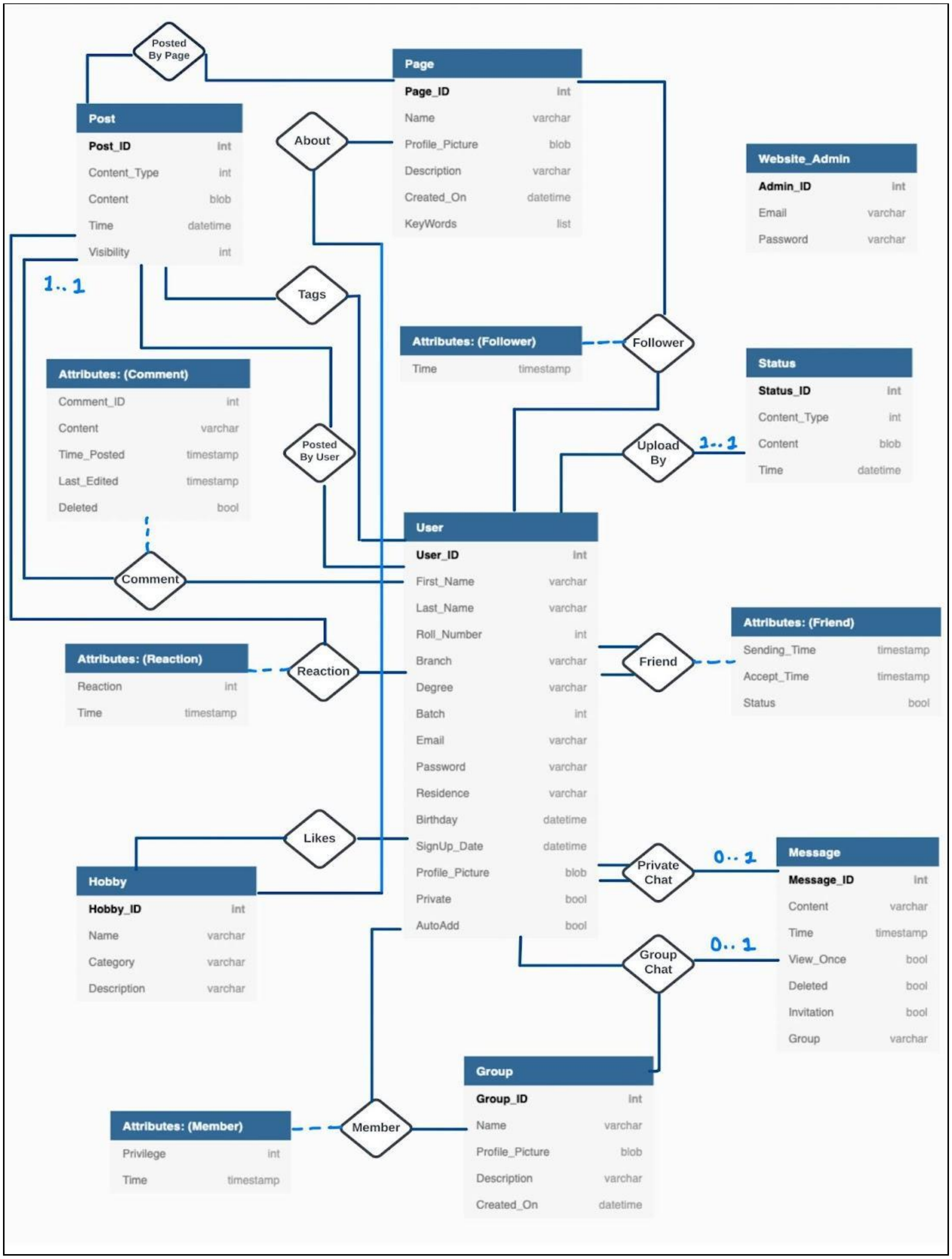
We were not able to develop the following features on our Instigram website:

1) Groups and Pages
2) Website Admin Privileges: We decided that the Analytics could be made available to each user, and therefore, Website Admins were not implemented.
3) Status: We were able to sort out how to handle communication of images between backend and frontend at the later stages, and not sufficient time was available for the status.

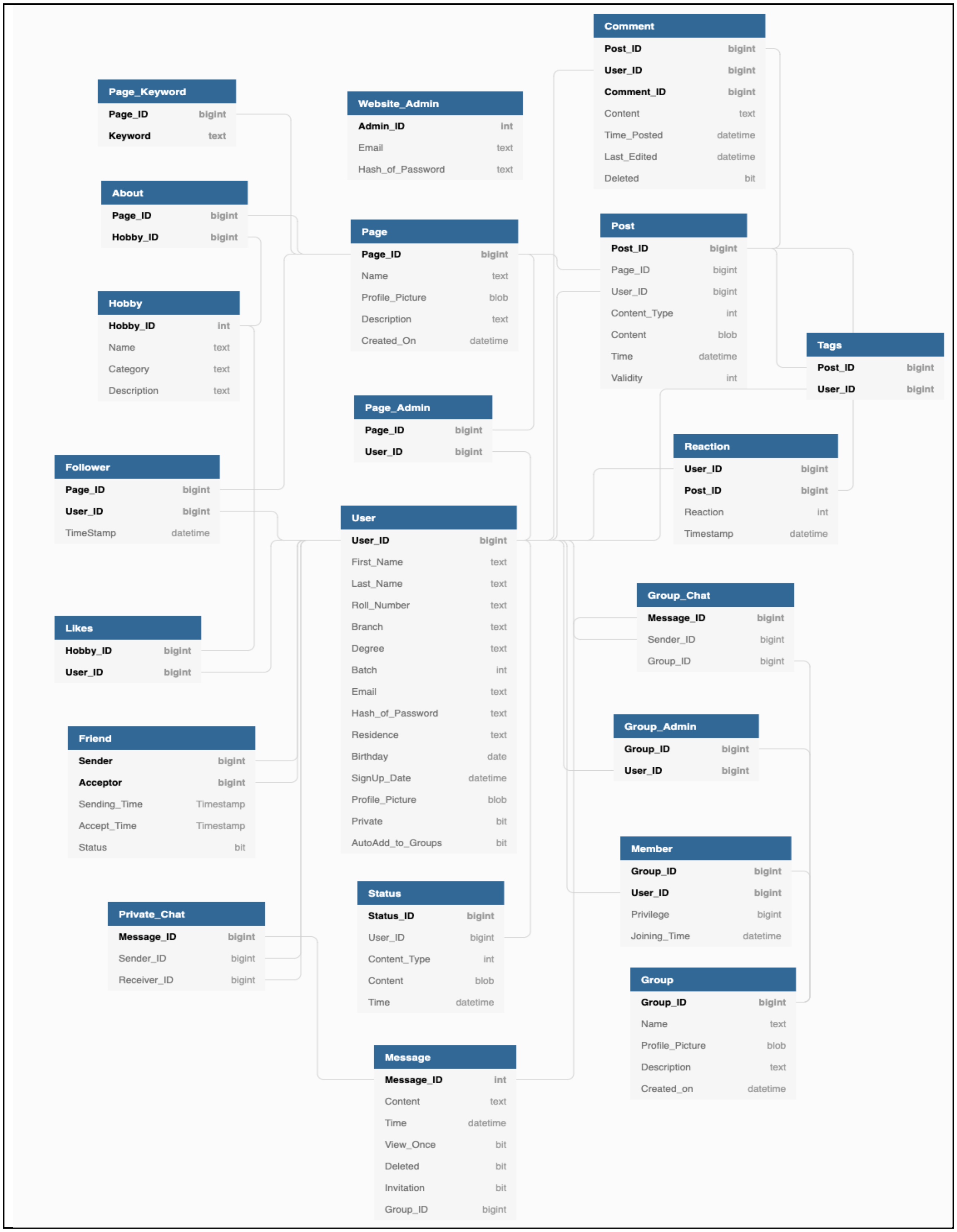
Github Link

https://github.com/Dhruva-Dhingra/CS387_Project.git

ER Diagram


Posted By Page
Page
Page_ID int
Name varchar
Profile_Picture blob
Description varchar
Created_On datetime
KeyWords list
Post
Post_ID int
Content_Type int
Content blob
Time datetime
Visibility int
About
Website_Admin
Admin_ID int
Email varchar
Password varchar
1..1
Tags
Attributes: (Follower)
Time timestamp
Follower
Status
Status_ID int
Content_Type int
Content blob
Time datetime
Attributes: (Comment)
Comment_ID int
Content varchar
Time_Posted timestamp
Last_Edited timestamp
Deleted bool
Posted By User
Upload By
1..1
User
User_ID int
First_Name varchar
Last_Name varchar
Roll_Number int
Branch varchar
Degree varchar
Batch int
Email varchar
Password varchar
Residence varchar
Birthday datetime
SignUp_Date datetime
Profile_Picture blob
Private bool
AutoAdd bool
Comment
Attributes: (Friend)
Sending_Time timestamp
Accept_Time timestamp
Status bool
Friend
Attributes: (Reaction)
Reaction int
Time timestamp
Reaction
Likes
Private Chat
0..1
Message
Message_ID int
Content varchar
Time timestamp
View_Once bool
Deleted bool
Invitation bool
Group varchar
Hobby
Hobby_ID int
Name varchar
Category varchar
Description varchar
Group Chat
0..1
Group
Group_ID int
Name varchar
Profile_Picture blob
Description varchar
Created_On datetime
Attributes: (Member)
Privilege int
Time timestamp
Member

Logical Schema: 3-NF Normalised



Page_Keyword
Page_ID bigint
Keyword text
About
Page_ID bigint
Hobby_ID bigint
Hobby
Hobby_ID int
Name text
Category text
Description text
Follower
Page_ID bigint
User_ID bigint
TimeStamp datetime
Likes
Hobby_ID bigint
User_ID bigint
Friend
Sender bigint
Acceptor bigint
Sending_Time Timestamp
Accept_Time Timestamp
Status bit
Private_Chat
Message_ID bigint
Sender_ID bigint
Receiver_ID bigint
Website_Admin
Admin_ID int
Email text
Hash_of_Password text
Page
Page_ID bigint
Name text
Profile_Picture blob
Description text
Created_On datetime
Page_Admin
Page_ID bigint
User_ID bigint
User
User_ID bigint
First_Name text
Last_Name text
Roll_Number text
Branch text
Degree text
Batch int
Email text
Hash_of_Password text
Residence text
Birthday date
SignUp_Date datetime
Profile_Picture blob
Private bit
AutoAdd_to_Groups bit
Status
Status_ID bigint
User_ID bigint
Content_Type int
Content blob
Time datetime
Message
Message_ID int
Content text
Time datetime
View_Once bit
Deleted bit
Invitation bit
Group_ID bigint
Comment
Post_ID bigint
User_ID bigint
Comment_ID bigint
Content text
Time_Posted datetime
Last_Edited datetime
Deleted bit
Post
Post_ID bigint
Page_ID bigint
User_ID bigint
Content_Type int
Content blob
Time datetime
Validity int
Tags
Post_ID bigint
User_ID bigint
Reaction
User_ID bigint
Post_ID bigint
Reaction int
Timestamp datetime
Group_Chat
Message_ID bigint
Sender_ID bigint
Group_ID bigint
Group_Admin
Group_ID bigint
User_ID bigint
Member
Group_ID bigint
User_ID bigint
Privilege bigint
Joining_Time datetime
Group
Group_ID bigint
Name text
Profile_Picture blob
Description text
Created_on datetime