DBMS Mini-Project

UE21CS351A: Database Management System

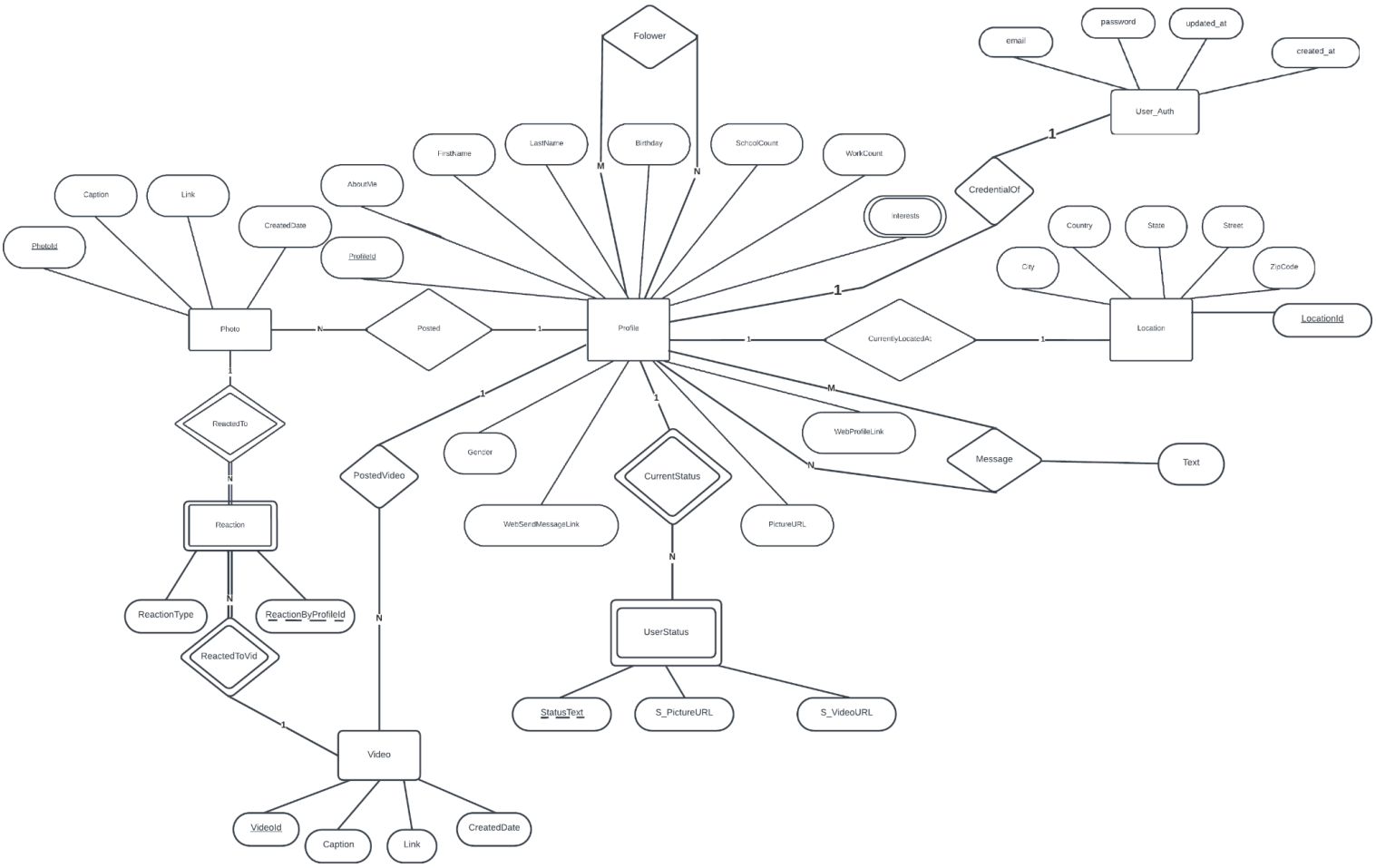
Project Title: Social Media Application

Team:

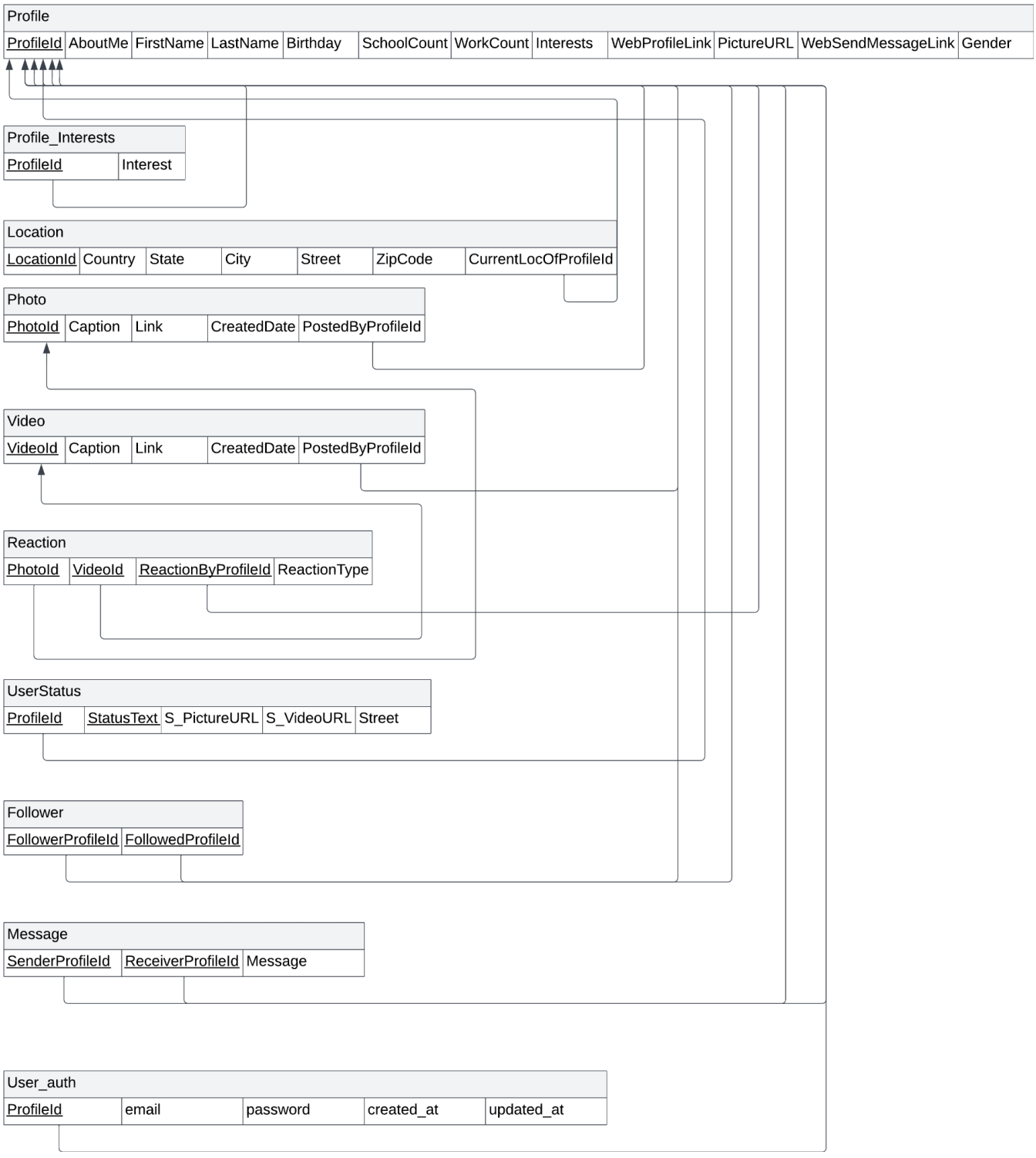
Name: PRANAVA S SRN: PES1UG21CS435

Name: R HEMA BHUSHAN SRN: PES1UG21CS462

ER Diagram



Relational Schema



MINI PROJECT USER REQUIREMENT SPECIFICATION

Introduction

Purpose:

The mini-project, social media app serves as a platform where people create and post content, share conversations, and reflect on others' content through comments and reactions. The app would allow many people to connect on a common platform. People across different regions could share their thoughts, hobbies, and practices and discuss ideas with other people.The app would provide a platform to people to improve their networking

Scope:

The project aims to showcase the usage of different entities and data types in a relational database. RDBMS is mostly used for storing metadata and textual data in the case of applications like social networking sites. Large complex inter-relationships can exist between various tables within a given database, the project would demonstrate the usage of these relationships and their corresponding attributes if applicable

Project Description

Project Overview:

The project is a social media app that allows a user to hold an account on the platform through which the user can curate and post content such as photos, videos, and text. Each user can have associated peers with whom his/her content is shared, with this the other users would be able to react and comment on the content posted. Users would also be able to chat with each other in a given section of the platform.

Major Project Functionalities:

1. To provide a simplified interface to share and post content
2. To provide an interface which allows users to chat on a one-to-one basis
3. Showcase usage of RDBMS on a social networking site

System Features and Function Requirements

1. User Profiles : Ability to create user profiles and edit them
2. Posts: Users will be able to create,edit and delete posts which can be a photo or a video
3. Following: Users will be able to follow each other
4. Reactions: Users will be able to react to the posts by their followers

DDL SQL Commands

Creating follower table:

```
CREATE TABLE `follower` (
  `followerprofileid` bigint NOT NULL,
  `followedprofileid` bigint NOT NULL,
  PRIMARY KEY (`followerprofileid`,`followedprofileid`),
  KEY `followedprofileid` (`followedprofileid`),
```

```
  CONSTRAINT `follower_ibfk_1` FOREIGN KEY
(`followerprofileid`) REFERENCES `profile`
(`profileid`),
  CONSTRAINT `follower_ibfk_2` FOREIGN KEY
(`followedprofileid`) REFERENCES `profile` (`profileid`)
)
```

Creating location table:

```
CREATE TABLE `location` (
  `LocationId` bigint NOT NULL AUTO_INCREMENT,
  `Country` varchar(15) DEFAULT NULL,
  `State` varchar(15) DEFAULT NULL,
  `City` varchar(15) DEFAULT NULL,
  `Street` varchar(20) DEFAULT NULL,
  `ZipCode` varchar(10) DEFAULT NULL,
  `currentlocofprofileid` bigint DEFAULT NULL,
  PRIMARY KEY (`LocationId`),
  KEY `currentlocofprofileid` (`currentlocofprofileid`),
  CONSTRAINT `location_ibfk_1` FOREIGN KEY
(`currentlocofprofileid`) REFERENCES `profile`
(`profileid`)
)
```

Creating message table:

```
CREATE TABLE `message` (
  `senderprofileid` bigint NOT NULL,
  `receiverprofileid` bigint NOT NULL,
  `Message` varchar(400) DEFAULT NULL,
  `messageid` bigint NOT NULL AUTO_INCREMENT,
  `mes_timestamp` timestamp NULL DEFAULT
CURRENT_TIMESTAMP,
  PRIMARY KEY (`messageid`),
```

```
  KEY `senderprofileid` (`senderprofileid`),
  KEY `receiverprofileid` (`receiverprofileid`),
  CONSTRAINT `message_ibfk_1` FOREIGN KEY
(`senderprofileid`) REFERENCES `profile` (`profileid`),
  CONSTRAINT `message_ibfk_2` FOREIGN KEY
(`receiverprofileid`) REFERENCES `profile` (`profileid`)
)
```

Creating photo table:

```
CREATE TABLE `photo` (
  `PhotoId` bigint NOT NULL AUTO_INCREMENT,
  `Caption` varchar(500) DEFAULT NULL,
  `Link` varchar(200) DEFAULT NULL,
  `CreatedTimestamp` timestamp NULL DEFAULT
CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  `postedbyprofileid` bigint DEFAULT NULL,
  PRIMARY KEY (`PhotoId`),
  KEY `postedbyprofileid` (`postedbyprofileid`),
  CONSTRAINT `photo_ibfk_1` FOREIGN KEY
(`postedbyprofileid`) REFERENCES `profile` (`profileid`)
)
```

Creating profile table:

```
CREATE TABLE `profile` (
  `profileid` bigint NOT NULL,
  `AboutMe` varchar(200) DEFAULT NULL,
  `FirstName` varchar(20) DEFAULT NULL,
  `LastName` varchar(20) DEFAULT NULL,
  `Birthday` date DEFAULT NULL,
  `SchoolCount` int DEFAULT NULL,
  `WorkCount` int DEFAULT NULL,
  `WebProfileLink` varchar(50) DEFAULT NULL,
  `PictureURL` varchar(50) DEFAULT NULL,
```

```
  `WebSendMessageLink` varchar(50) DEFAULT NULL,
  `Gender` enum('Male','Female','Other') DEFAULT NULL,
  PRIMARY KEY (`profileid`),
  CONSTRAINT `profile_ibfk_1` FOREIGN KEY (`profileid`)
REFERENCES `user_auth` (`profileid`)
)
```

Creating table profile_interests:

```
CREATE TABLE `profile_interests` (
  `profileid` bigint NOT NULL,
  `Interest` varchar(15) NOT NULL,
  PRIMARY KEY (`profileid`,`Interest`),
  CONSTRAINT `profile_interests_ibfk_1` FOREIGN KEY
(`profileid`) REFERENCES `profile` (`profileid`)
)
```

Creating table reactions:

```
CREATE TABLE `reactions` (
  `ReactionId` bigint NOT NULL AUTO_INCREMENT,
  `PhotoId` bigint DEFAULT NULL,
  `VideoId` bigint DEFAULT NULL,
  `reactionbyprofileid` bigint DEFAULT NULL,
  `ReactionType`
enum('Like','Laugh','Cry','Wow','Angry','Sad') DEFAULT
NULL,
  PRIMARY KEY (`ReactionId`),
  KEY `reactionbyprofileid` (`reactionbyprofileid`),
  KEY `PhotoId` (`PhotoId`),
  KEY `VideoId` (`VideoId`),
  CONSTRAINT `reactions_ibfk_3` FOREIGN KEY
(`reactionbyprofileid`) REFERENCES `profile`
(`profileid`),
```

```
  CONSTRAINT `reactions_ibfk_4` FOREIGN KEY (`PhotoId`)
REFERENCES `photo` (`PhotoId`),
  CONSTRAINT `reactions_ibfk_5` FOREIGN KEY (`VideoId`)
REFERENCES `video` (`VideoId`)
)
```

Creating table user_auth:

```
CREATE TABLE `user_auth` (
  `profileid` bigint NOT NULL AUTO_INCREMENT,
  `email` varchar(100) DEFAULT NULL,
  `password` varchar(200) DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  `updated_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP
ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`profileid`)
)
```

Creating tables userstatus:

```
    CREATE TABLE `userstatus` (
      `profileid` bigint NOT NULL,
      `StatusText` varchar(100) NOT NULL,
      `S_PictureURL` varchar(50) DEFAULT NULL,
      `S_VideoURL` varchar(50) DEFAULT NULL,
      PRIMARY KEY (`profileid`,`StatusText`),
      CONSTRAINT `userstatus_ibfk_1` FOREIGN KEY (`profileid`)
REFERENCES `profile` (`profileid`)
    )
```

Creating table video:

```
    CREATE TABLE `video` (
      `VideoId` bigint NOT NULL AUTO_INCREMENT,
```

```
      `Caption` varchar(500) DEFAULT NULL,
      `Link` varchar(50) DEFAULT NULL,
      `CreatedTimestamp` timestamp NULL DEFAULT CURRENT_TIMESTAMP
ON UPDATE CURRENT_TIMESTAMP,
      `postedbyprofileid` bigint DEFAULT NULL,
      PRIMARY KEY (`VideoId`),
      KEY `postedbyprofileid` (`postedbyprofileid`),
      CONSTRAINT `video_ibfk_1` FOREIGN KEY (`postedbyprofileid`)
REFERENCES `profile` (`profileid`)
    )
```

CRUD Operations Screenshots

User Operations:

Sign In:

```
        const queryRes = await query("SELECT * FROM user_auth
WHERE email=?", [
          email,
        ]);
```

Sign Up:

```
          const existingUser = await query("SELECT * FROM
user_auth WHERE email=?", [email,
        ]);
         const createdUser = await query(
          "INSERT INTO user_auth (email,password) VALUES (?,?)",
          [email, hashedPassword]
        );
```

```
    const profileUser = await query(
          "UPDATE profile SET firstname = ?, lastname = ?,
birthday =  ?, gender =  ? WHERE profileid = ?",
          [firstname, lastname, dob, gender,
createdUser["insertId"]]
        );
```

View Current Profile:

```
     const profileRes = await query(
          "SELECT * FROM profile WHERE profileid=(SELECT
profileid FROM user_auth WHERE email = ?)",
          [email]
        );
```

Edit Profile:

```
    const profileUpdate = await query(
          "UPDATE profile SET aboutme = ?, firstname = ?,
lastname = ?,birthday = ?,schoolcount = ?,workcount = ?,gender =
? WHERE profileid = (SELECT profileid FROM user_auth WHERE email
= ?)",
          [
            aboutme,
            firstname,
            lastname,
            birthday,
            schoolcount,
            workcount,
            gender,
            email,
          ]
        );
```

View Current Profile Feed:

```
    const photoRes = await query(
          "SELECT * FROM photo WHERE postedbyprofileid IN (SELECT
followedprofileid FROM follower WHERE followerprofileid=(SELECT
profileid FROM user_auth WHERE email = ?))",
          [email]
        );
        const videoRes = await query(
          "SELECT * FROM video WHERE postedbyprofileid IN (SELECT
followedprofileid FROM follower WHERE followerprofileid=(SELECT
profileid FROM user_auth WHERE email = ?))",
          [email]
        );
```

View own posts:

```
    const photoRes = await query(
          "SELECT * FROM photo WHERE postedbyprofileid=(SELECT
profileid FROM user_auth WHERE email = ?)",
          [email]
        );
        const videoRes = await query(
          "SELECT * FROM video WHERE postedbyprofileid=(SELECT
profileid FROM user_auth WHERE email = ?)",
          [email]
        );
```

Search People:

```
    const searchRes = await query(
          "SELECT * FROM profile WHERE (firstname like ?) OR
(lastname like ?)",
```

```
            [search, search]
        );
```

Get unfollowed Profiles:

```
    const profileRes = await query(
            "SELECT * FROM profile WHERE profileid NOT IN (SELECT
followedprofileid FROM follower WHERE followerprofileid=(SELECT
profileid FROM user_auth WHERE email=?)) AND profileid <> (SELECT
profileid FROM user_auth WHERE email=?)",
            [email, email]
        );
```

Delete Account:

```
     const deleteRes = await query(
            "call delete_account((SELECT profileid FROM user_auth
WHERE email=?))",
            [email]
        );
```

Social Networking operations:

Follow Profile:

```
     const addFollowRes = await query(
            "INSERT INTO follower
(followerprofileid,followedprofileid) VALUES (?,?)",
            [followerprofileid, followedprofileid]
        );
```

Unfollow Profile:

```
    const removeFollowRes = await query(
            "DELETE FROM follower WHERE followerprofileid = ? and
followedprofileid = ?",
```

```
            [followerprofileid, followedprofileid]
        );
```

Total follower Count function:

```
    const followerNum = await query(
        "SELECT total_followers((SELECT profileid FROM
user_auth WHERE email = ?)) as followercount",
        [email]
    );
```

Total followed Count function:

```
    const followedNum = await query(
        "SELECT total_followed((SELECT profileid FROM user_auth
WHERE email = ?)) as followedcount",
        [email]
    );
```

Post operations:

Create Photo Post:

```
    const insertPostRes = await query(
        "INSERT INTO photo (caption,link,postedbyprofileid)
VALUES (?,?,(SELECT profileid FROM user_auth WHERE email = ?))",
        [caption, link, email]
    );
```

Update Photo Post:

```
    const updatePostRes = await query(
        "UPDATE photo SET caption = ?, Link = ? WHERE PhotoId =
?",
```

```
          [caption, link, postid]
        );
```

Delete Photo Post:

```
      const deleteRes = await query("DELETE FROM photo WHERE
PhotoId = ?", [
          postid,
        ]);
```

Create Video Post:

```
     const insertPostRes = await query(
          "INSERT INTO video (caption,link,postedbyprofileid)
VALUES (?,?,(SELECT profileid FROM user_auth WHERE email = ?))",
          [caption, link, email]
        );
```

Update Video Post:

```
    const updatePostRes = await query(
          "UPDATE photo SET caption = ? , Link = ? WHERE VideoId
= ?",
          [caption, link, postid]
        );
```

Delete Video Post:

```
     const deleteRes = await query("DELETE FROM video WHERE
VideoId = ?", [
          postid,
        ]);
```

React To Photo:

```
      const reactInsertRes = await query(
          "INSERT INTO reactions
(photoid,reactionbyprofileid,reactiontype) VALUES (?,(SELECT
profileid FROM user_auth WHERE email = ?),?)",
          [postid, email, reactiontype]
        );
```

React to Video:

```
      const reactInsertRes = await query(
          "INSERT INTO reactions
(videoid,reactionbyprofileid,reactiontype) VALUES (?,(SELECT
profileid FROM user_auth WHERE email = ?),?)",
          [postid, email, reactiontype]
        );
```

Update Reaction:

```
     const reactDelRes = await query(
          "UPDATE reactions SET reactiontype = ? WHERE reactionid
= ?",
          [reactiontype, reactionid]
        );
```

Delete Reaction:

```
     const reactDelRes = await query(
          "DELETE FROM reactions WHERE reactionid = ?",
          [reactionid]
        );
```

Reactions Count:

```
    if (isPhoto) {
          queryStr = "SELECT total_photo_reactions(?) as
reactioncount";
        } else {
          queryStr = "SELECT total_video_reactions(?) as
reactioncount";
        }
```

Message Operations:

Add New Message:

```
     const insertMessageRes = await query(
          "INSERT INTO message
(senderprofileid,receiverprofileid,Message) VALUES ((SELECT
profileid FROM user_auth WHERE email = ?),?,?)",
          [email, receiverprofileid, message]
        );
```

Remove Sent Message:

```
       const delRes = await query("DELETE FROM message WHERE
messageid = ?", [
          messageid,
        ]);
```

Retrieve chats with followers:

```
      const messageRes = await query(
          "SELECT distinct
m.receiverprofileid,m.Message,m.mes_timestamp,p.FirstName,p.LastN
ame FROM message m RIGHT JOIN profile p ON
p.profileid=m.receiverprofileid WHERE m.senderprofileid = (SELECT
u.profileid FROM user_auth u WHERE u.email = ?) ORDER BY
m.mes_timestamp DESC",
```

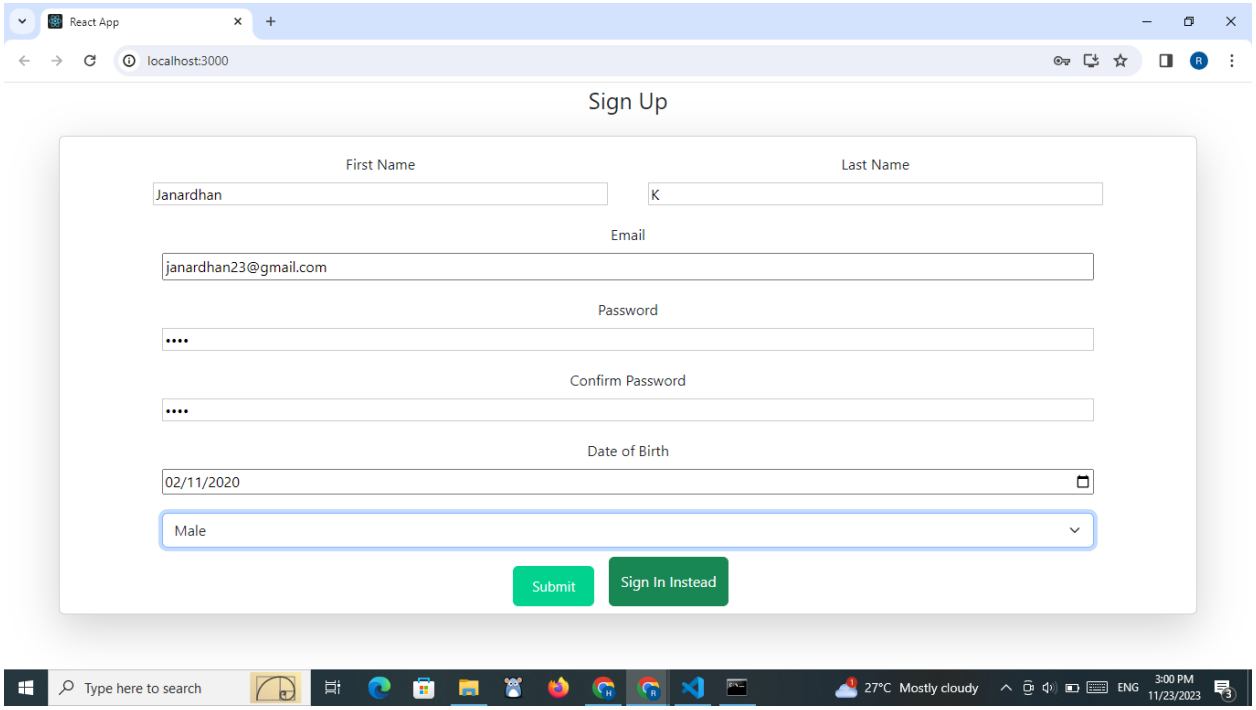
```
            [email]
        );
```

Retrieve messages with a specific profile:

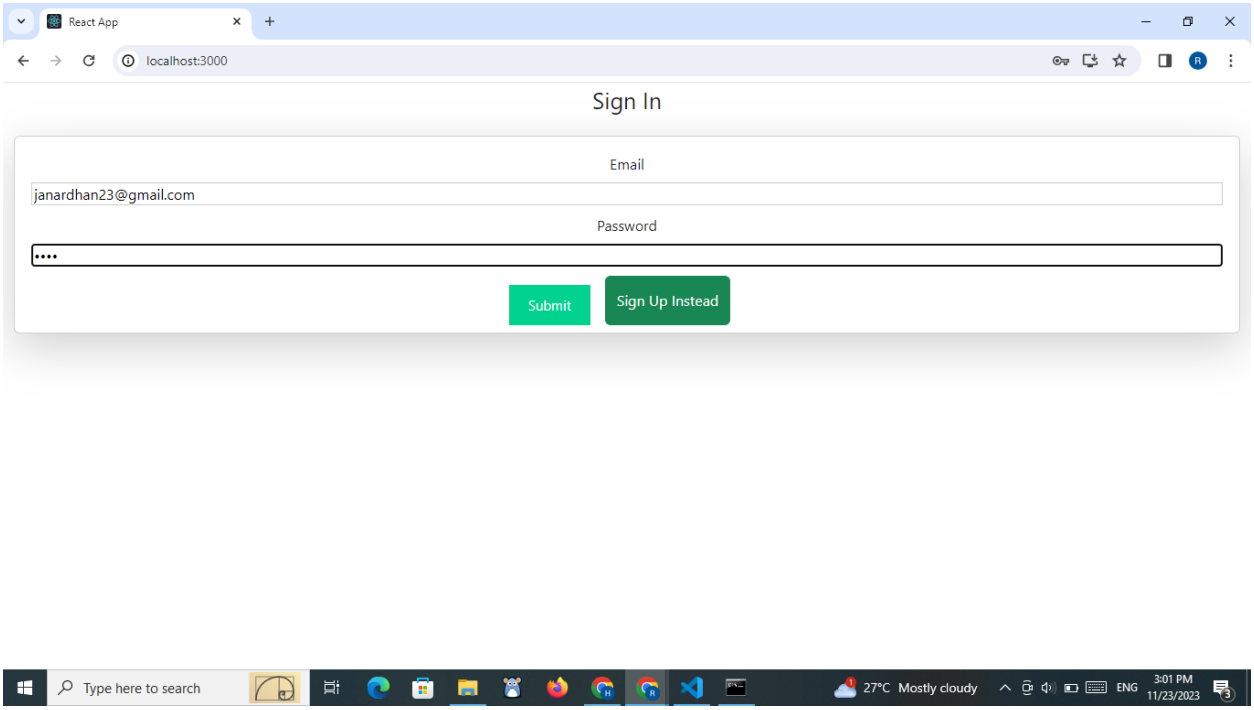
```
      const messageRes = await query(
          "SELECT
m.senderprofileid,m.messageid,m.Message,m.mes_timestamp,p.FirstNa
me,p.LastName FROM message m INNER JOIN profile p ON
p.profileid=m.senderprofileid WHERE (m.senderprofileid = ? AND
m.receiverprofileid = (SELECT profileid FROM user_auth u WHERE
u.email = ?)) OR (m.senderprofileid = (SELECT profileid FROM
user_auth u WHERE u.email = ?) AND m.receiverprofileid = ?) GROUP
BY
m.senderprofileid,p.FirstName,m.messageid,m.Message,m.mes_timesta
mp ORDER BY m.mes_timestamp",
          [chatwithprofileid, email, email, chatwithprofileid]
        );
```

Functionalities and associated frontend screenshots

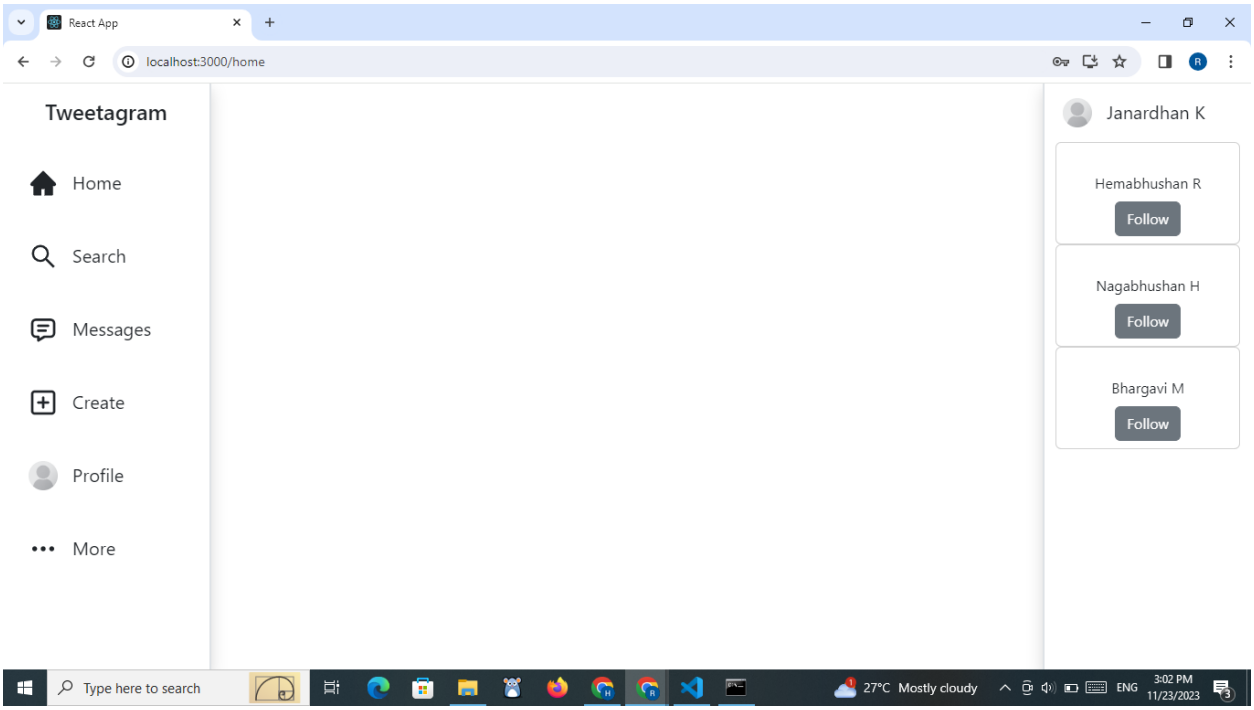
Sign Up (Creates a new entry in user_auth and profile table)



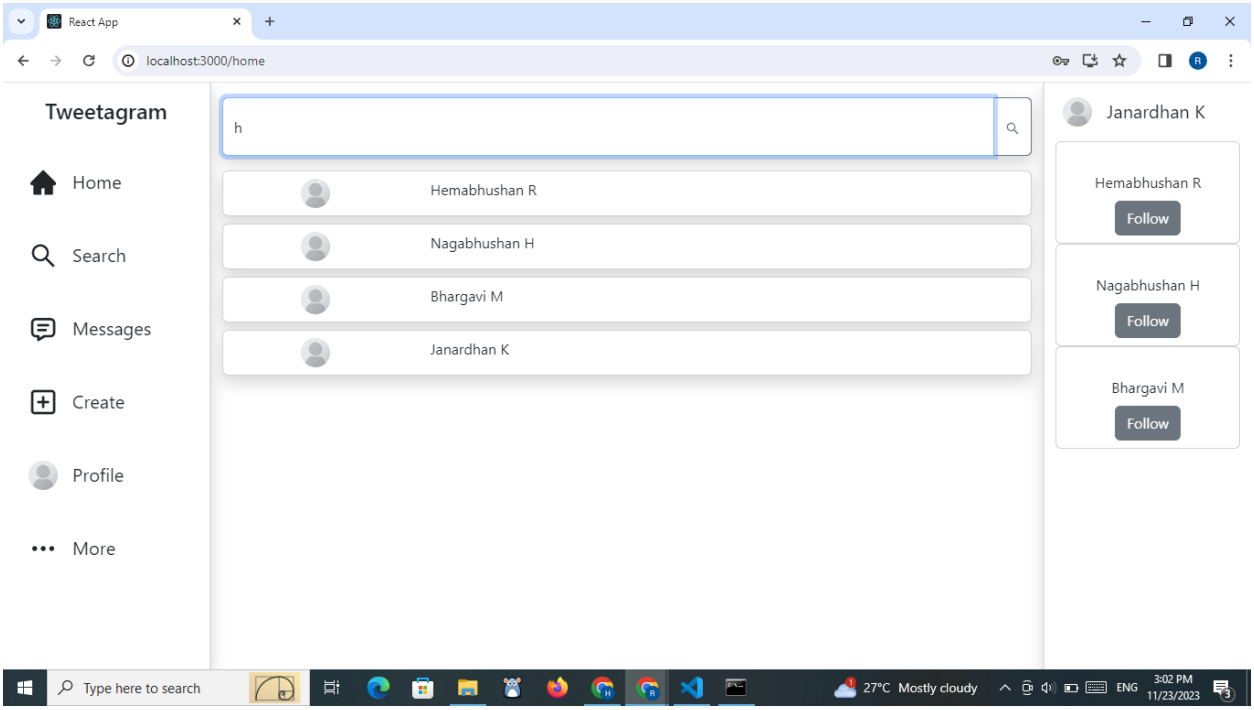
Sign In (Verifies entered credentials against corresponding entry in user_auth table)



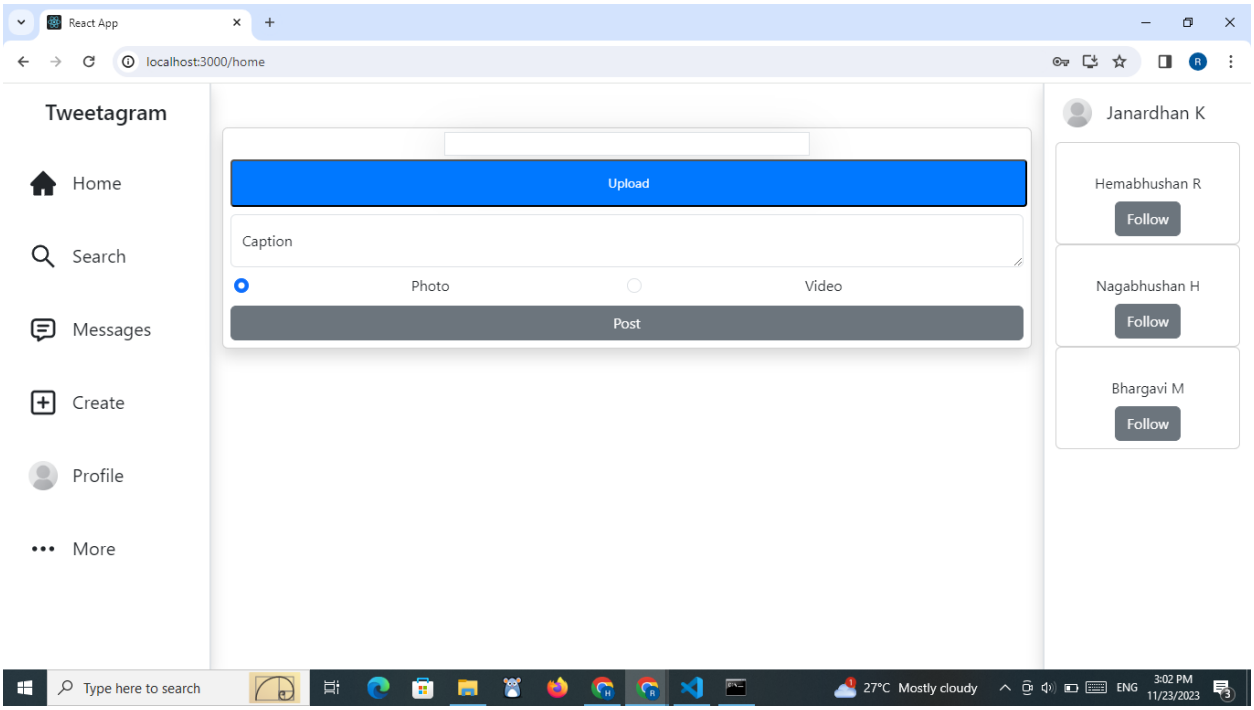
Home Page which reads and shows all available users from profile table allowing current to follow them



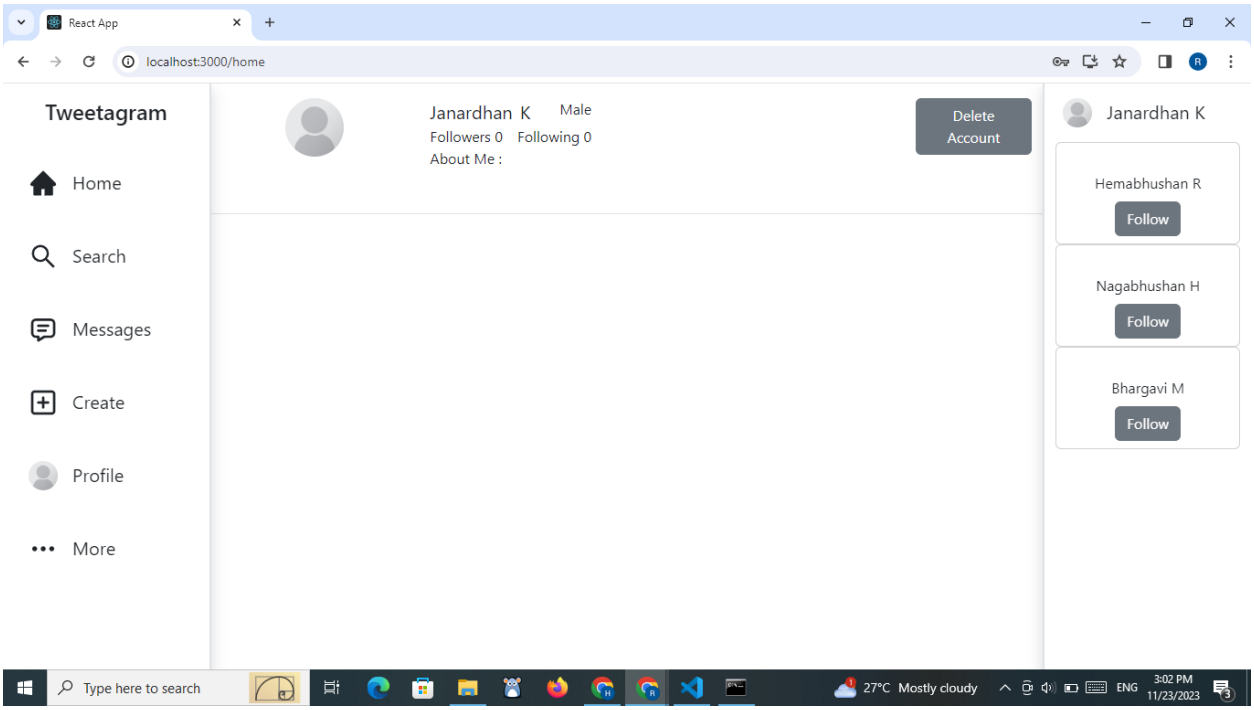
Search box allowing to search and retrieve names of accounts from profile table



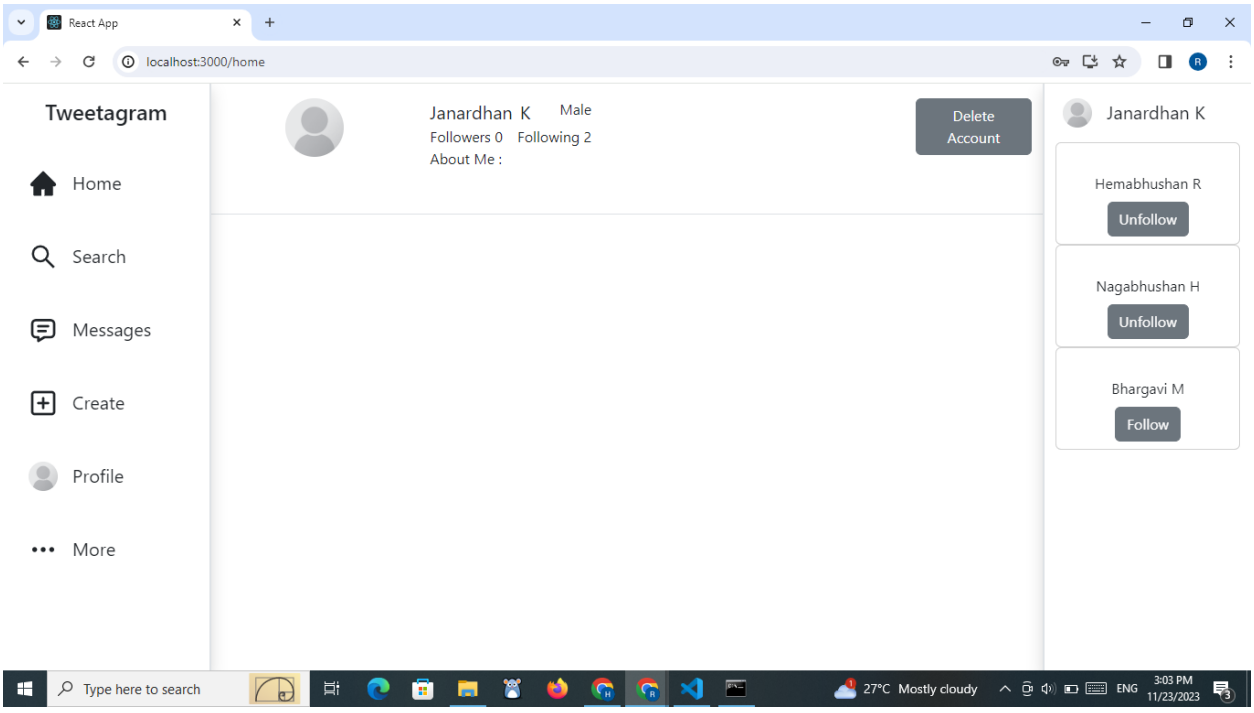
Page to upload video or photo and retrieve its url allowing user to post Video or Photo(Makes an entry in Video or Photo table)



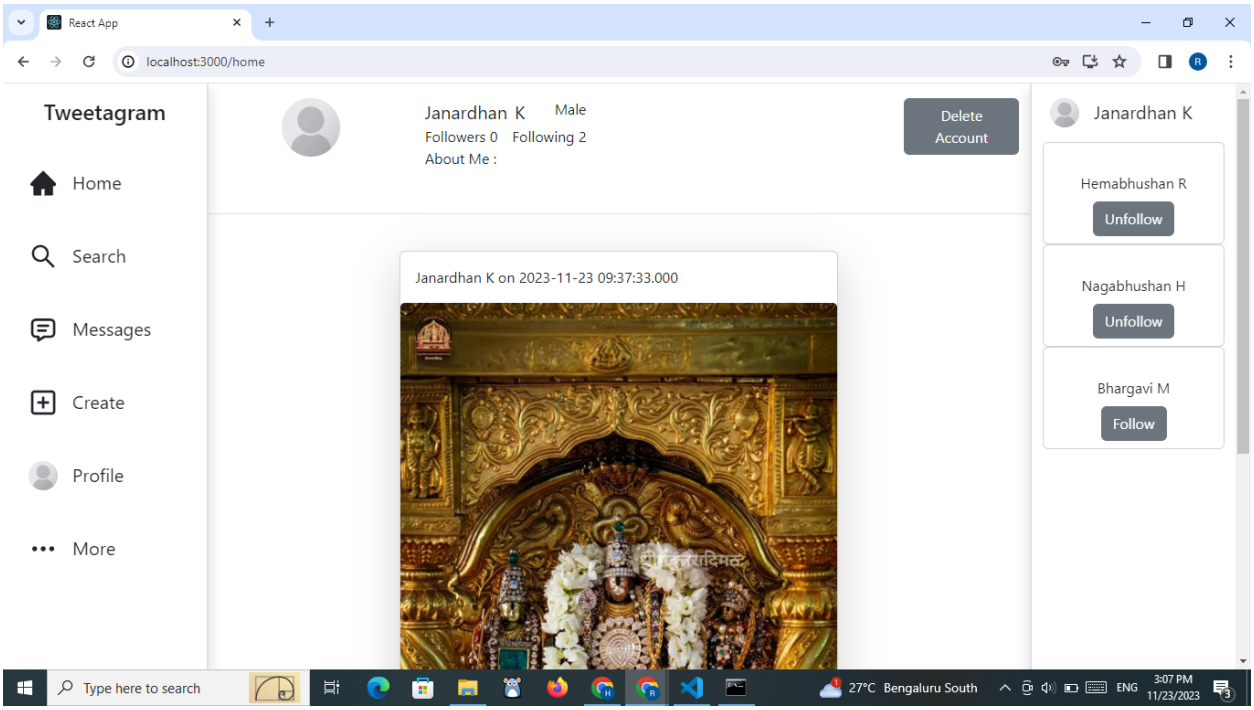
Current user Profile view which allows user to view his/her followers or following count, posts posted by the current user , also allows user to delete account (Makes two function calls total_followers(profileid) total_followed(profileid) and on deleting account calls procedure delete_account(profileid))



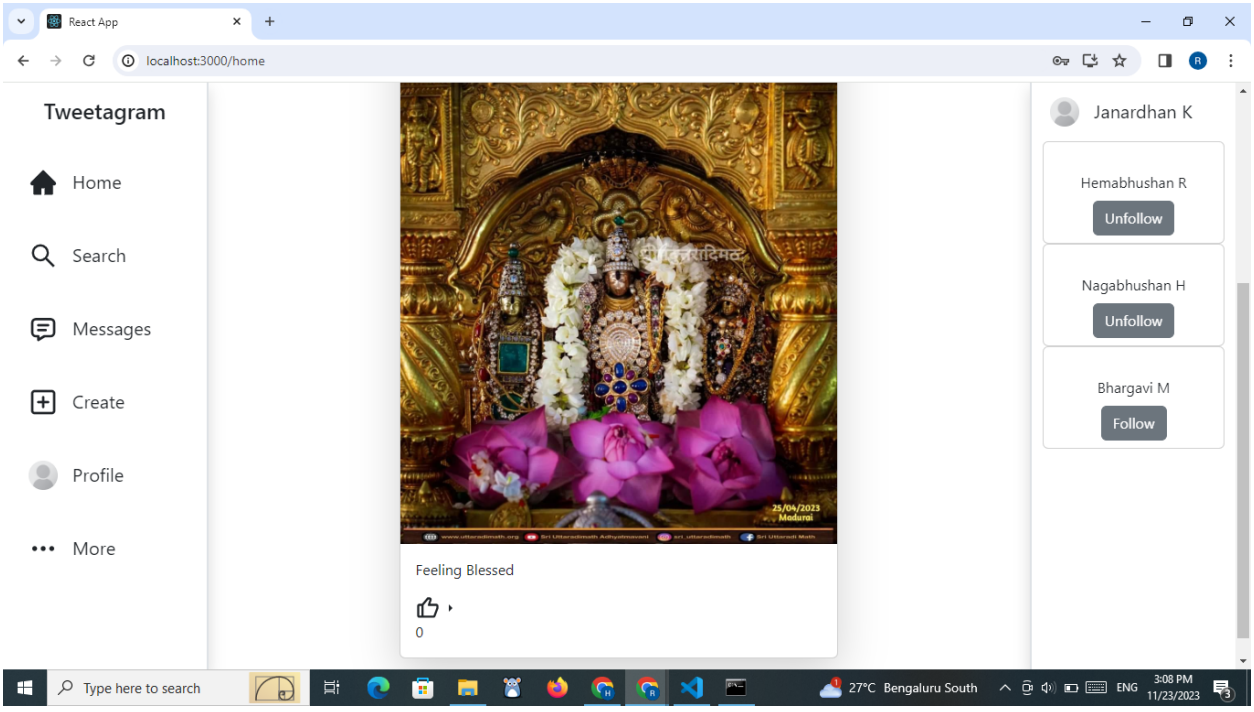
After following two people shown on side bar (Makes an entry in follower table)



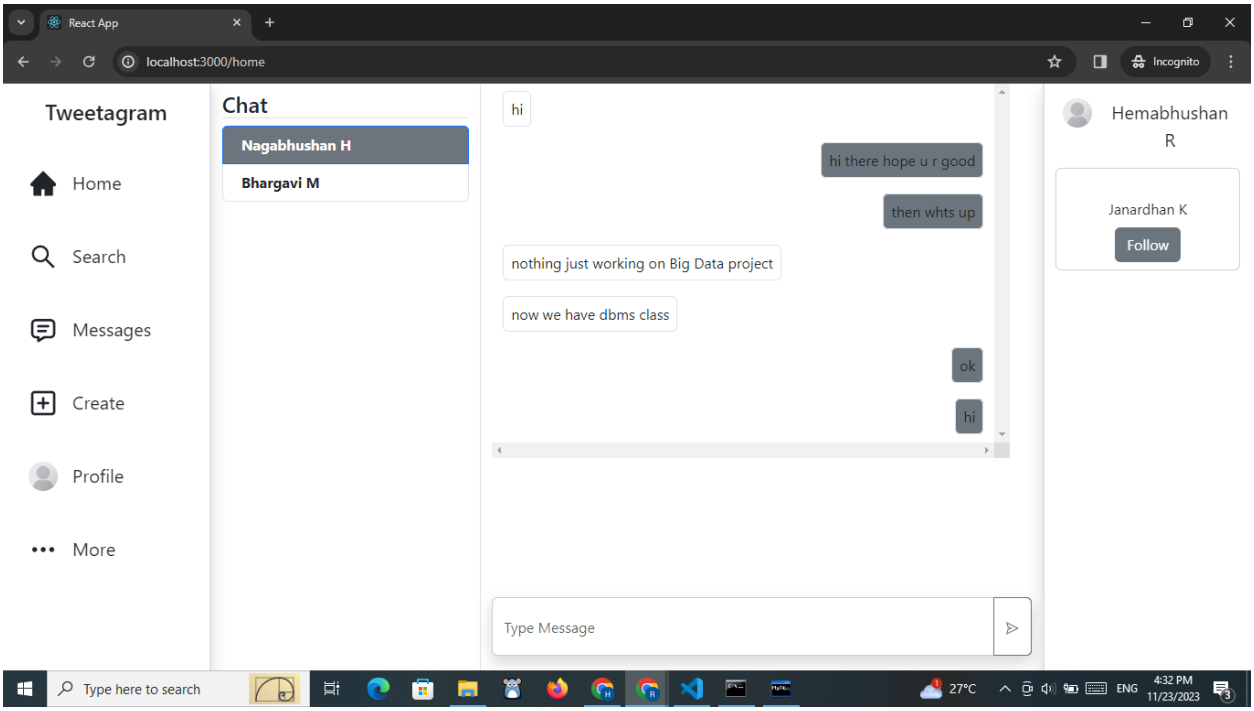
After posting a photo



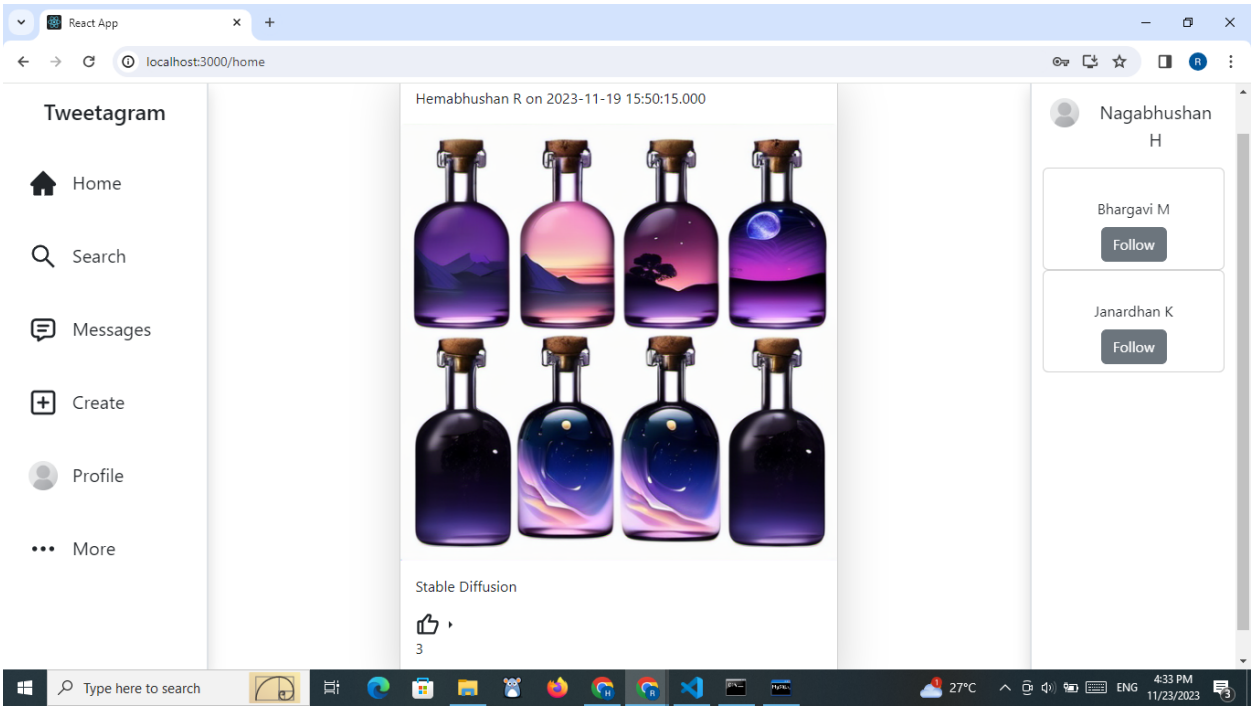
Post showing reaction count and option to react to the photo post (Calls a function total_reactions(reactid) and on a reaction makes an entry in reactions table)



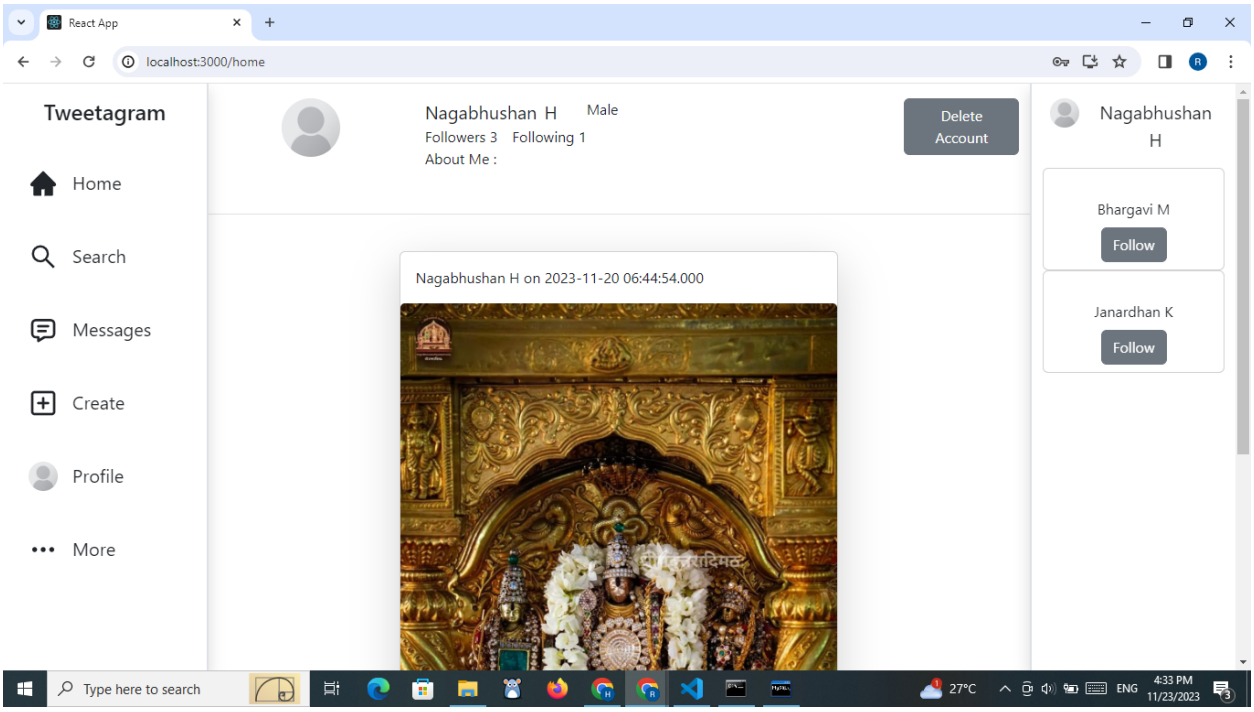
Chat Message Page which allows user to chat with his/her followers



Home Page of a user showing photo posted by follower

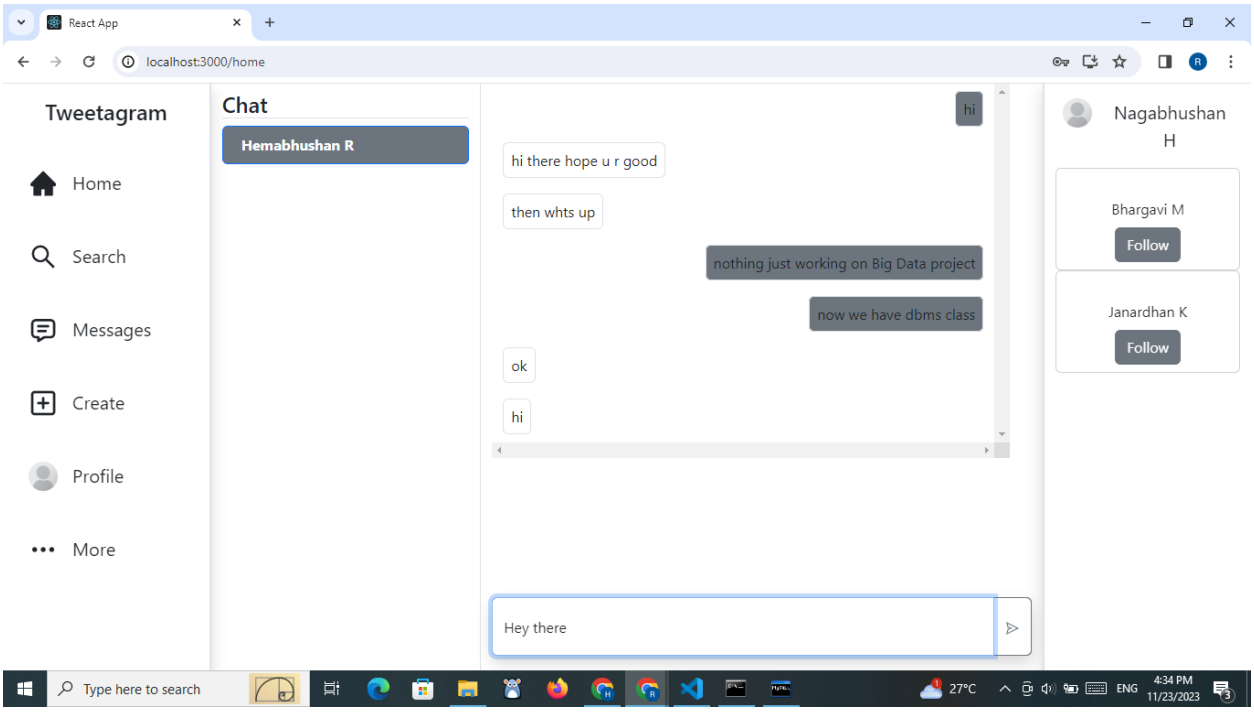


Profile view of user showing photo posted by him

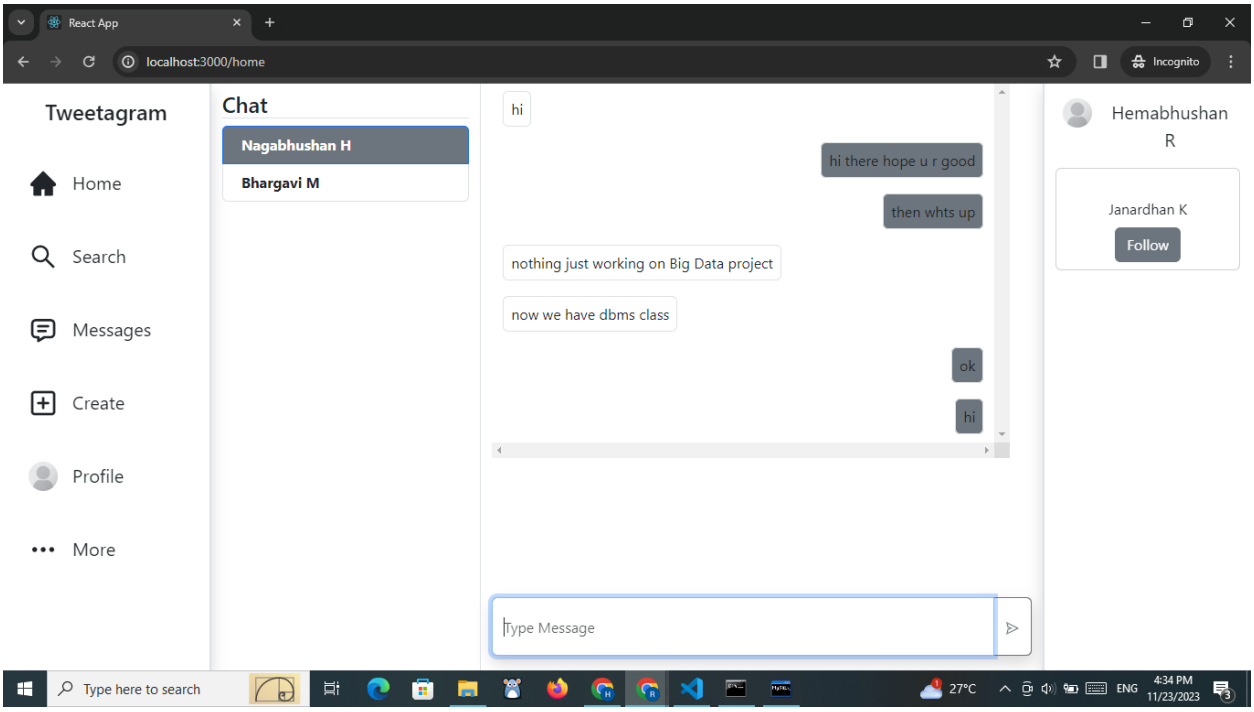


Chat Page before sending message (Retrieves messages after performing join

operation between two tables message and profile)



Chat page on other user end before receiving message



Chat page after receiving message from other user

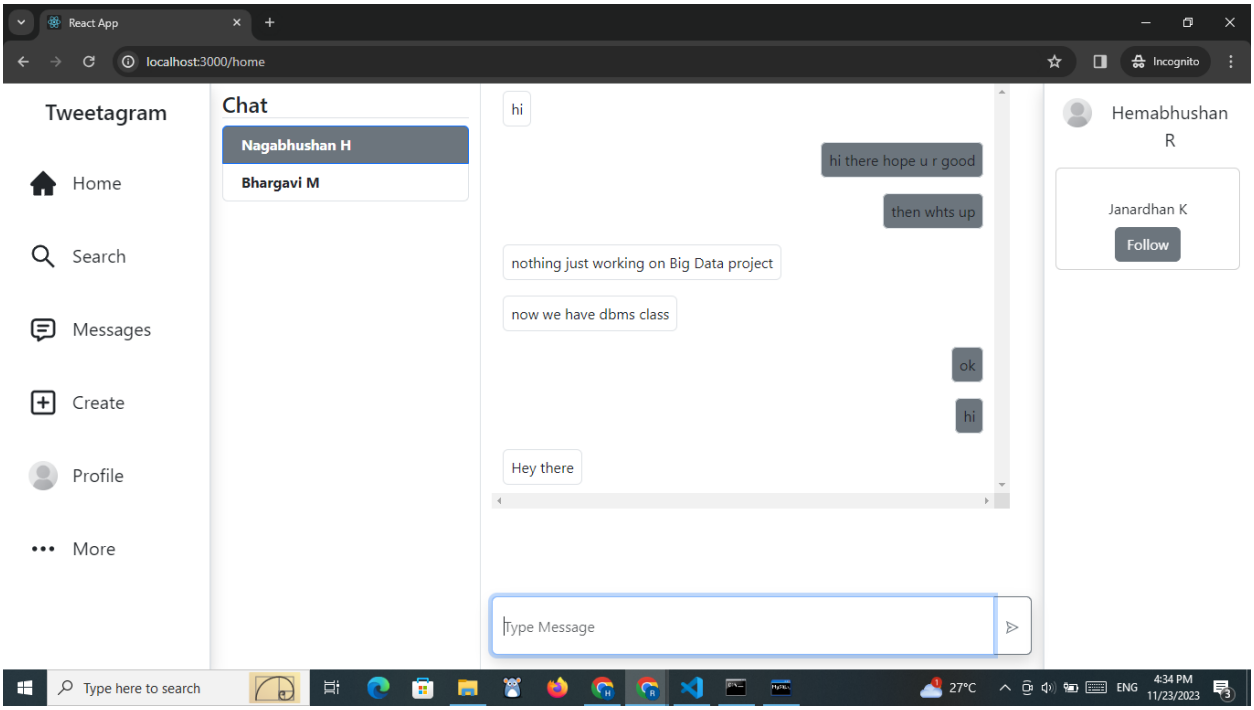


Photo before reacting to the post

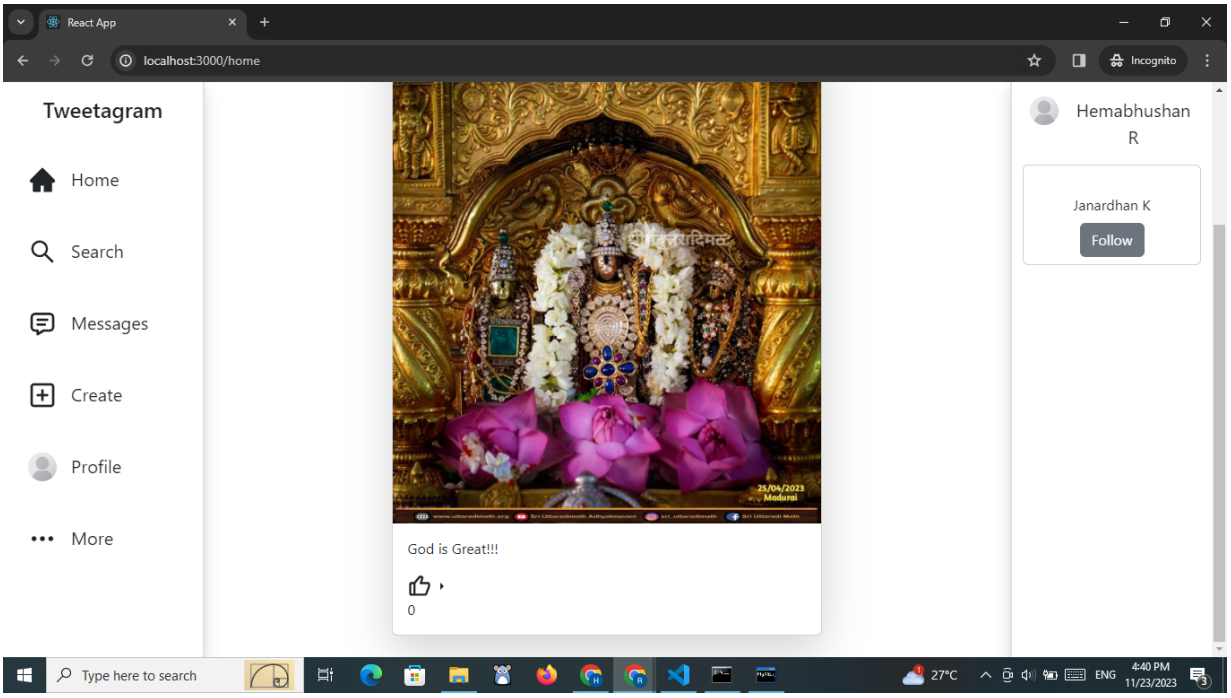
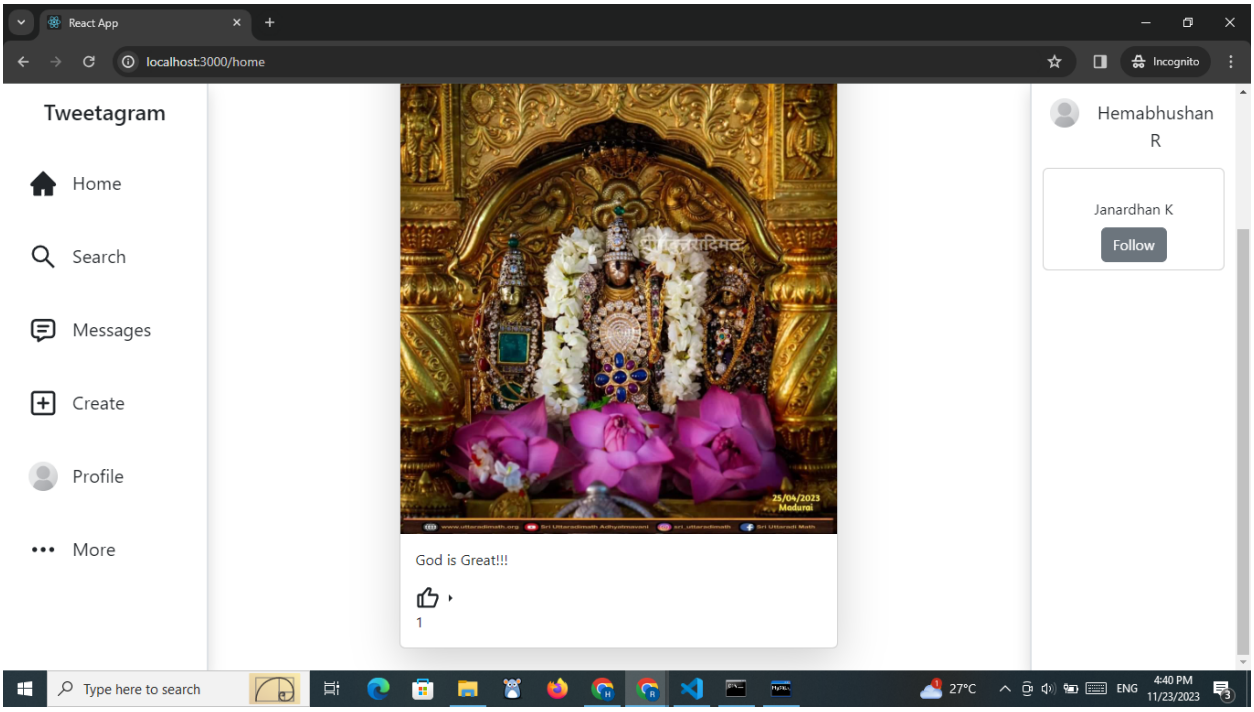


Photo after reacting to the post



Triggers:

```
DELIMITER ;;
CREATE TRIGGER `delete_photo_reactions` BEFORE DELETE ON `photo`
FOR EACH ROW
begin
delete from reactions r where r.photoid=old.photoid;
end ;;
DELIMITER ;
```

```
DELIMITER ;;
CREATE TRIGGER `account_delete` BEFORE DELETE ON `profile` FOR
EACH ROW begin
delete from location loc where
loc.currentlocofprofileid=old.profileid;
```

```
delete from profile_interests pi where
pi.profileid=old.profileid;
delete from photo p where p.postedbyprofileid=old.profileid;
delete from video v where v.postedbyprofileid=old.profileid;
delete from reactions r where
r.reactionbyprofileid=old.profileid;
delete from userstatus u where u.profileid=old.profileid;
delete from  follower f where f.followedprofileid=old.profileid
or f.followerprofileid=old.profileid;
delete from  message m where m.senderprofileid=old.profileid or
m.receiverprofileid=old.profileid;
end ;;
DELIMITER ;
```

```
DELIMITER ;;
CREATE  TRIGGER `user_create` AFTER INSERT ON `user_auth` FOR
EACH ROW begin
insert into profile (profileid) values (new.profileid);
insert into location (currentlocofprofileid) values
(new.profileid);
end ;;
DELIMITER ;
```

```
DELIMITER ;;
CREATE TRIGGER `delete_video_reactions` BEFORE DELETE ON `video`
FOR EACH ROW begin
delete from reactions r where r.videoid=old.videoid;
end ;;
DELIMITER ;
```

Functions:

```
CREATE FUNCTION `total_followed`(profid varchar(40))
RETURNS int DETERMINISTIC
begin
declare followerd int;
select count(f.followedprofileid) into followerd from follower f
where f.followerprofileid=profid;
return followerd;
end ;
```

```
CREATE FUNCTION `total_followers`(profid varchar(40))
RETURNS int DETERMINISTIC
begin
declare followers int;
select count(f.followerprofileid) into followers from follower f
where f.followedprofileid=profid;
return followers;
end ;
CREATE  FUNCTION `total_photo_reactions`(phid bigint)
RETURNS int DETERMINISTIC
begin
declare t_reactions int;
select count(ReactionId) into t_reactions from reactions r where
r.PhotoId=phid;
return t_reactions;
end ;
```

```
CREATE FUNCTION `total_video_reactions`(vidid bigint)
RETURNS int DETERMINISTIC
begin
declare t_reactions int;
```

```
select count(ReactionId) into t_reactions from reactions r where
r.VideoId=vidid;
return t_reactions;
end ;
```

Procedures:

```
CREATE PROCEDURE `delete_account`(profid bigint)
begin
delete from profile p where p.profileid=profid;
end ;
```

Nested Queries:

```
    const photoRes = await query(
          "SELECT * FROM photo WHERE postedbyprofileid IN (SELECT
followedprofileid FROM follower WHERE followerprofileid=(SELECT
profileid FROM user_auth WHERE email = ?))",
          [email]
        );
        const videoRes = await query(
          "SELECT * FROM video WHERE postedbyprofileid IN (SELECT
followedprofileid FROM follower WHERE followerprofileid=(SELECT
profileid FROM user_auth WHERE email = ?))",
          [email]
        );
```

```
    const photoRes = await query(
          "SELECT * FROM photo WHERE postedbyprofileid=(SELECT
profileid FROM user_auth WHERE email = ?)",
          [email]
        );
```

```
        const videoRes = await query(
          "SELECT * FROM video WHERE postedbyprofileid=(SELECT
profileid FROM user_auth WHERE email = ?)",
          [email]
        );
```

```
    const profileRes = await query(
          "SELECT * FROM profile WHERE profileid NOT IN (SELECT
followedprofileid FROM follower WHERE followerprofileid=(SELECT
profileid FROM user_auth WHERE email=?)) AND profileid <> (SELECT
profileid FROM user_auth WHERE email=?)",
          [email, email]
        );
```

Join Queries:

```
     const messageRes = await query(
          "SELECT distinct
m.receiverprofileid,m.Message,m.mes_timestamp,p.FirstName,p.LastN
ame FROM message m RIGHT JOIN profile p ON
p.profileid=m.receiverprofileid WHERE m.senderprofileid = (SELECT
u.profileid FROM user_auth u WHERE u.email = ?) ORDER BY
m.mes_timestamp DESC",
          [email]
        );
```

```
      const messageRes = await query(
          "SELECT
m.senderprofileid,m.messageid,m.Message,m.mes_timestamp,p.FirstNa
me,p.LastName FROM message m INNER JOIN profile p ON
p.profileid=m.senderprofileid WHERE (m.senderprofileid = ? AND
m.receiverprofileid = (SELECT profileid FROM user_auth u WHERE
```

```
u.email = ?)) OR (m.senderprofileid = (SELECT profileid FROM
user_auth u WHERE u.email = ?) AND m.receiverprofileid = ?) GROUP
BY
m.senderprofileid,p.FirstName,m.messageid,m.Message,m.mes_timesta
mp ORDER BY m.mes_timestamp",
          [chatwithprofileid, email, email, chatwithprofileid]
        );
```

Aggregate Queries:

```
select count(f.followedprofileid) into followerd from follower f
where f.followerprofileid=profid;
```

```
select count(ReactionId) into t_reactions from reactions r where
r.PhotoId=phid;
```

Code snippets for invoking Triggers/Functions/Procedures

```
     const followerNum = await query(
          "SELECT total_followers((SELECT profileid FROM
user_auth WHERE email = ?)) as followercount",
          [email]
        );
```

```
     const followedNum = await query(
          "SELECT total_followed((SELECT profileid FROM user_auth
WHERE email = ?)) as followedcount",
          [email]
        );
```

```
    if (isPhoto) {
          queryStr = "SELECT total_photo_reactions(?) as
reactioncount";
```

```
        } else {
          queryStr = "SELECT total_video_reactions(?) as
reactioncount";
        }
```

```
    const deleteRes = await query(
          "call delete_account((SELECT profileid FROM user_auth
WHERE email=?))",
          [email]
        );
```