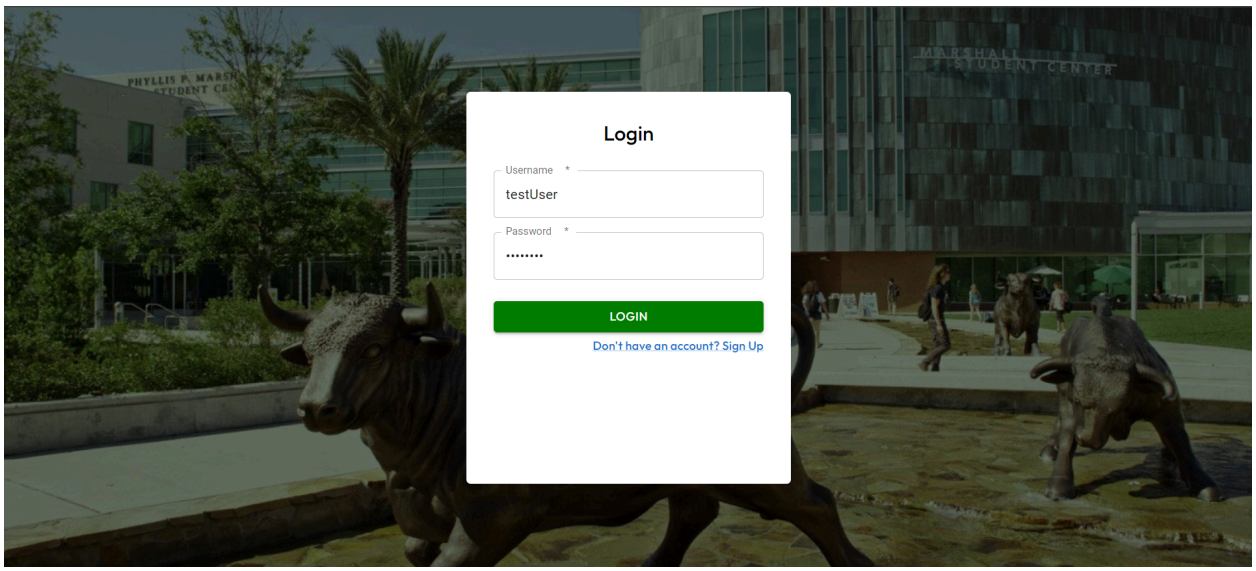
Final Project Report

(Done By: Bhuvan Biju, Rajas Bhate, Rachit Kansagra)

Our project is titled "MemoraShare," a social media web application designed exclusively for teenagers and college students. The web app centers around creating and sharing memories through a visually appealing timeline that displays photos of important events. Users can collaborate with friends on shared events, fostering a sense of collectiveness. MemoraShare offers a perfect blend of privacy and sociability, allowing users to mark certain events as private while showcasing others publicly on their profiles.

The following sections include a description along with screenshots of all the functionalities of our web app. For all the SQL queries $n refers to the n-th field provided as an input.

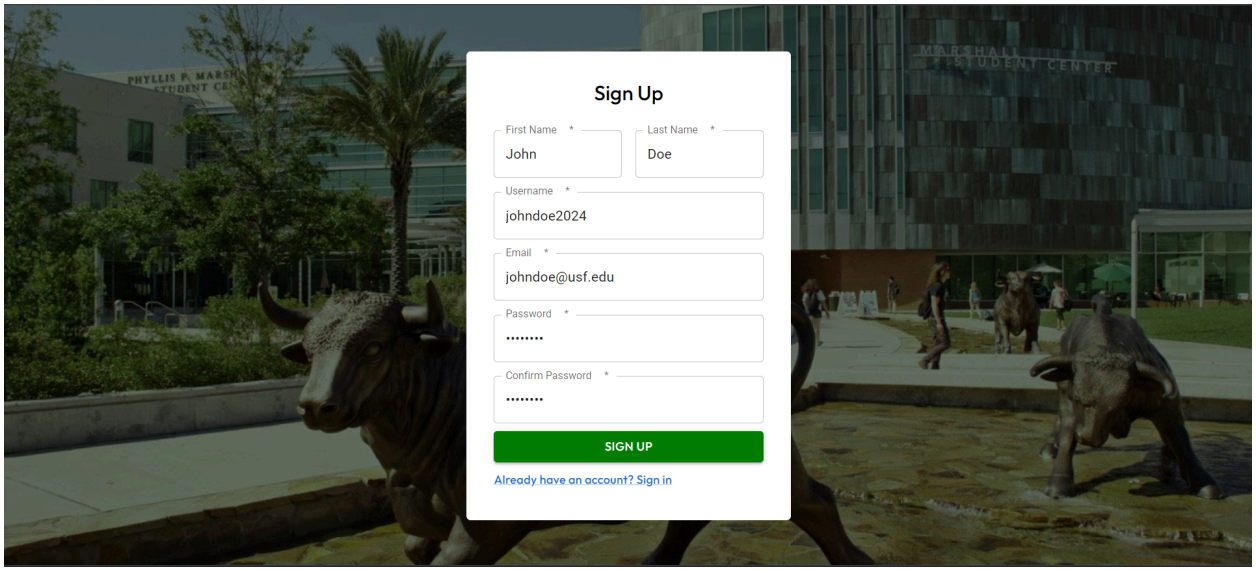
Logging In to The Website



Any user with an existing account can easily log in by entering their username and password. The backend code validates the user details & also ensures that there are no special characters (preventing SQL injection attacks). The query used for this section is:

- SELECT * FROM users WHERE username = $1 AND password = $2;
  (with input [username, password])

Signing Up As a New User

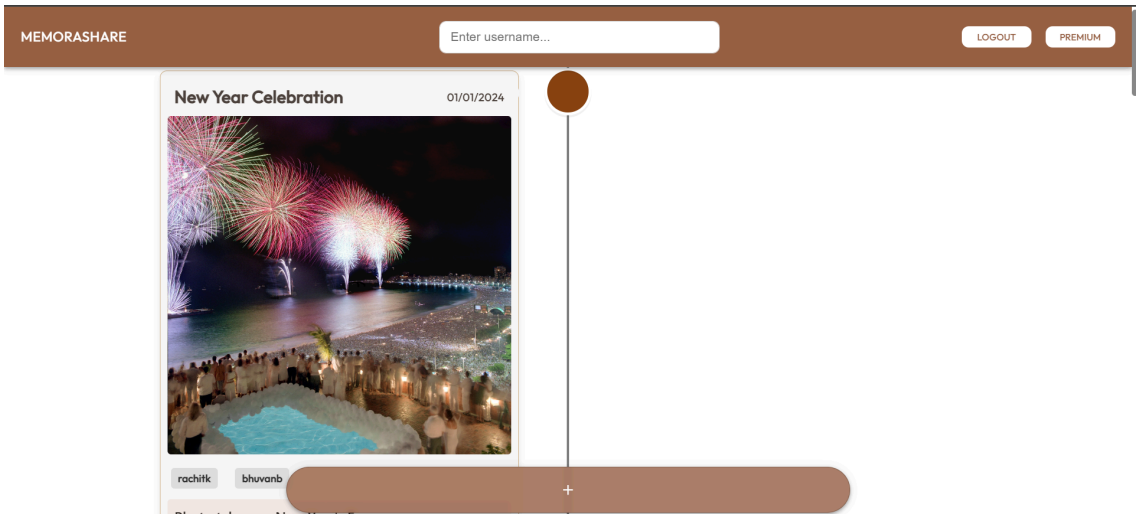


New users can create their accounts by entering valid details. By default, any new user is considered a free user and has a limit on the number of events they are allowed to post. The SQL queries used to handle signup are as follows:

- **To check whether the username is already taken**
  SELECT * FROM users WHERE username = $1 or email = $2;
  (with input [username, email])
- **To insert the entry into the users table**
  INSERT INTO users (firstname, lastname, username, password, email) VALUES ($1, $2, $3, $4, $5) RETURNING userid;
  (with input [firstname, lastname, username, password, email])
- **To insert the entry into the free_users table**
  INSERT INTO free_users (userid, count) VALUES ($1, $2)
  (with input [userID, 0])

The Events Timeline

After logging in, the user is greeted with a timeline showcasing all the events along with the photos posted for the event. The event name and description, along with the date this event occurred are all included for the user to see. Each event also displays all the other users who collaborated on the event. For a new user, their timeline is empty and they also see an option to become a premium user on their home page.

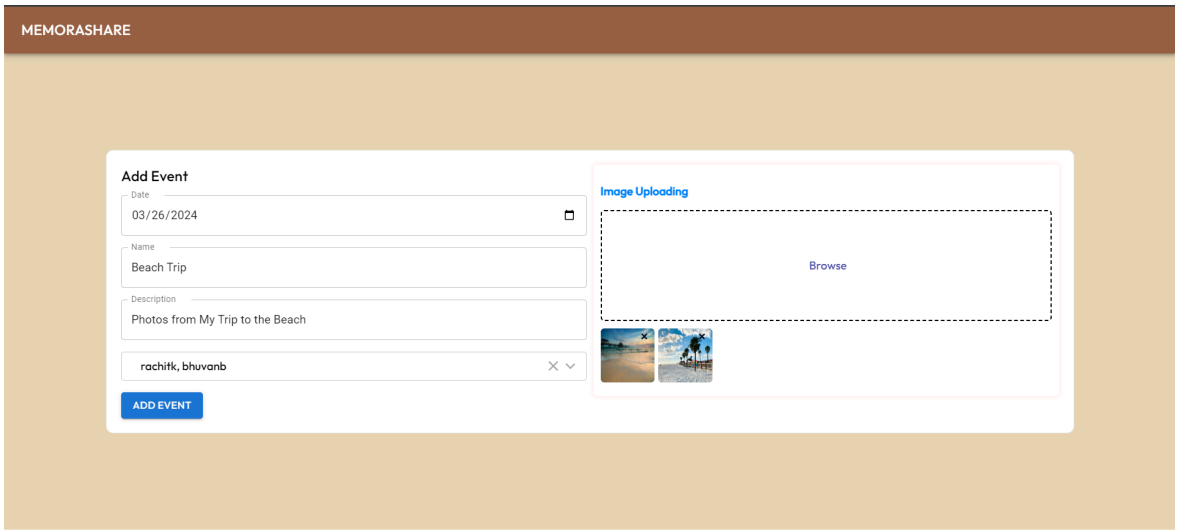


The SQL queries used to handle the timeline page are as follows:

- **To select all events (including collaborated) for a user**
  SELECT * FROM events WHERE eventid IN (SELECT eventid FROM collab WHERE userid = $1);
  (with input [userId])
- **To select all the details for those events**
  SELECT name, date, description FROM content c JOIN events e ON c.eventid = e.eventid WHERE c.eventid = $1;
  (with input [eventid])
- **To select the images for those events**
  SELECT file_path FROM content c JOIN events e ON c.eventid = e.eventid WHERE c.eventid = $1
  (with input [eventid])
- **To check whether the user has a premium account (used to display a message in the navigation bar)**
  SELECT userid FROM premium_users WHERE userid = $1;
  (with input [userId])

Adding an Event

A user can add an event by clicking the '+' button at the bottom of the timeline. The event add page allows the user to add an event with details such as the date the event took place, the name of the event, and a brief description. If it's a collaborated/group event, then a user can add the usernames of other people. Additionally, there is an option to upload or attach images related to the event.

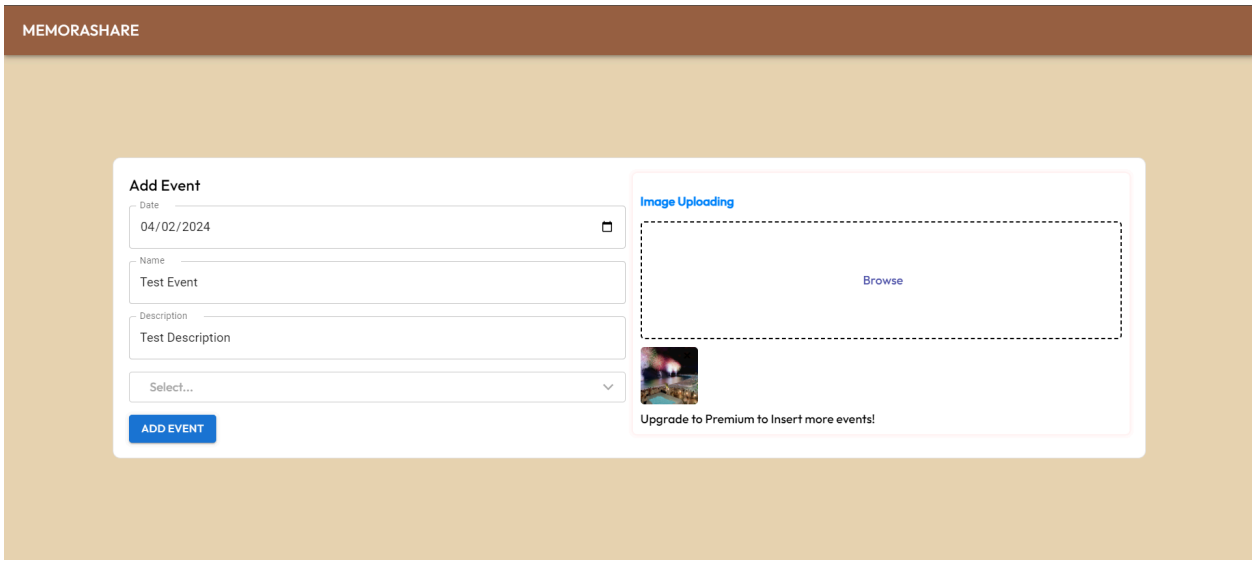


The SQL queries used to handle adding an event are as follows:

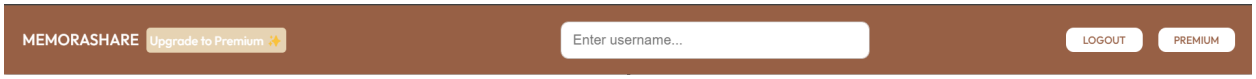
- **Adding the event to the events table**
  INSERT INTO events (date, name, description, userID) VALUES ($1, $2, $3, $4) RETURNING eventid;
  (with input [date, name, description,userid])
- **Adding an entry to the collaboration table (if the user collaborated with others)**
  INSERT INTO collab (eventid, userid) VALUES ($1, $2)
  (with input [eventid, userid])
- **Adding the images to the content table**
  INSERT INTO content (file_path, filename, eventid) VALUES ($1, $2, $3);
  (with input [path, originalname, eventId])

Free & Premium Users

For free users of Memorashare, there is a limitation on the number of events that can be added. Free accounts are restricted to a maximum of two events. If a free user tries to add more than two events, then an error message shows up on the add events page, which states “Upgrade to Premium to insert more than events!”.



However, users have the option to upgrade to a premium account, which would allow them to add an unlimited number of events without any such constraints. Free users can click the premium button on the navigation bar to become a premium user.

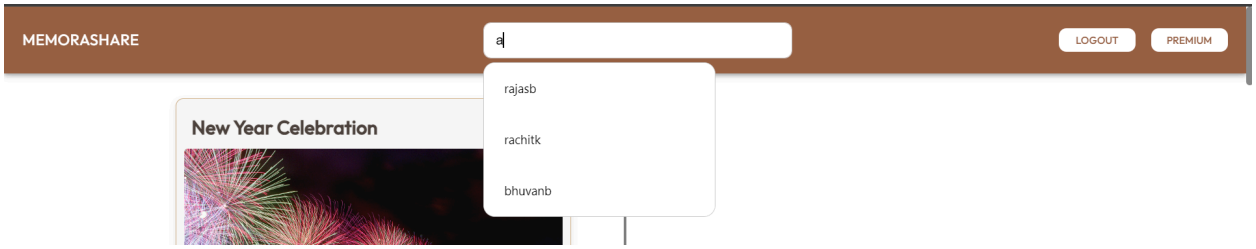


The SQL queries used to handle free and premium user accounts are as follows:

- **Converting a free user to a premium user**
  DELETE FROM free_users WHERE userid = $1;
  (with input [userId])
  INSERT INTO premium_users (userid) VALUES ($1)
  (with input [userId])

Searching other users

The home page of Memorashare also features a search bar in the navigation bar above the current user's timeline. This search functionality allows users to look for and access other people's timelines. By typing in a username and clicking on their name, the corresponding user's timeline will open up, providing a chronological view of all their shared events and memories. This social aspect encourages engagement and enables users to discover and connect with others on the platform.



The SQL queries used to handle searching for other users are as follows:

- **Finding usernames for the search bar**
  SELECT username, userid FROM users

PostgreSQL Tables:

The following screenshots include the queries used to create all the necessary tables, all the data present in the tables, as well as all of the implemented SQL sequences and SQL triggers.

Users Table:

```sql
-- Create the users table
CREATE TABLE users (
    userid INT PRIMARY KEY DEFAULT nextval('userid_sequence'),
    firstname VARCHAR(50),
    lastname VARCHAR(50),
    password VARCHAR(100),
    joined_date DATE DEFAULT CURRENT_DATE,
    username VARCHAR(50),
    email VARCHAR(100)
);
```

	userid [PK] integer	firstname character varying (50)	lastname character varying (50)	password character varying (100)	joined_date date	username character varying (50)	email character varying (100)
1	15	Rajas	Bhate	12345678	2024-04-25	rajasb	rajasbhate@usf.edu
2	16	Rachit	Kansagra	87654321	2024-04-25	rachitk	rachitkansagra@usf.edu
3	17	Bhuvan	Biju	1020304050	2024-04-25	bhuvanb	bhuvanbiju@usf.edu
4	18	William	Hendrix	12345678	2024-04-25	whendrix	whendrix@usf.edu
5	19	John	Doe	12345678	2024-04-25	jd	jd@gmail.com
6	20	Test	User	99999999	2024-04-26	testUser	testUser@usf.edu

Free_users Table:

```sql
-- Create the free_users table
CREATE TABLE free_users (
    userid INTEGER,
    count INTEGER,
    FOREIGN KEY (userid) REFERENCES users(userid)
);
```

	userid integer	count integer
1	16	0
2	17	0
3	19	1
4	20	2

Premium_users Table:

```sql
-- Create the premium_users table
CREATE TABLE premium_users (
    userid INTEGER,
    FOREIGN KEY (userid) REFERENCES users(userid)
);
```

	userid integer
1	15
2	18

Events Table:

```sql
-- Create the events table
CREATE TABLE events (
    eventid INTEGER PRIMARY KEY,
    date DATE,
    name VARCHAR(100),
    description TEXT,
    userid INTEGER,
    FOREIGN KEY (userid) REFERENCES users(userid)
);
```

	eventid [PK] integer	date date	name character varying (100)	description text	userid integer
1	38	2024-01-01	New Year Celebration	Photo taken on New Year's Eve	15
2	39	2024-01-14	My Birthday	Picture from my Birthday Party	15
3	41	2023-11-01	Diwali	Photos taken during Diwali	15
4	42	2024-04-01	Happy Birthday	Birthday with my friends	18
5	43	2024-04-25	Diwali	Diwali firecrackers	18
6	44	2024-04-25	New Year	Photos from new year	18
7	45	2024-04-25	New Year	Photos from new year	18
8	46	2024-04-01	Test	test	19
9	47	2024-03-26	Beach Trip	Photos from My Trip to the Beach	15

Collab Table:

```
-- Create the collab table
CREATE TABLE collab (
    eventid INT,
    userid INT,
    PRIMARY KEY (eventid, userid),
    FOREIGN KEY (eventid) REFERENCES events(eventid),
    FOREIGN KEY (userid) REFERENCES users(userid)
);
```

	eventid [PK] integer	userid [PK] integer
1	38	15
2	38	16
3	38	17
4	39	15
5	39	16
6	39	17
7	41	15
8	42	15
9	42	16
10	42	18

Content Table:

```sql
-- Create the content table
CREATE TABLE content (
    contentid INTEGER PRIMARY KEY,
    file_path VARCHAR(255),
    upload_date DATE,
    eventid INTEGER,
    FOREIGN KEY (eventid) REFERENCES events(eventid)
);
```

	contentid [PK] integer	file_path character varying (255)	eventid integer	filename character varying (255)
1	38	uploads\1714082838777-NewYears (1).jpg	38	NewYears (1).jpg
2	39	uploads\1714082866935-Birthday_1.jpg	39	Birthday_1.jpg
3	40	uploads\1714082866936-Birthday_2.jpg	39	Birthday_2.jpg
4	41	uploads\1714082936820-Diwali_1.jpg	41	Diwali_1.jpg
5	42	uploads\1714082936820-Diwali_2.avif	41	Diwali_2.avif
6	43	uploads\1714082936821-Diwali_3.jpg	41	Diwali_3.jpg
7	44	uploads\1714088465082-Birthday_1.jpg	42	Birthday_1.jpg
8	45	uploads\1714088465084-Birthday_2.jpg	42	Birthday_2.jpg
9	46	uploads\1714088541541-Diwali_2.avif	43	Diwali_2.avif
10	47	uploads\1714088541552-Diwali_3.jpg	43	Diwali_3.jpg
11	48	uploads\1714088623056-NewYears (1).jpg	44	NewYears (1).jpg
12	49	uploads\1714088623233-NewYears (1).jpg	45	NewYears (1).jpg
13	50	uploads\1714160408165-Beach_1.jpg	47	Beach_1.jpg
14	51	uploads\1714160408171-Beach_2.jpg	47	Beach_2.jpg
15	52	uploads\1714160455627-Diwali_2.avif	48	Diwali_2.avif

user_id sequence:

```
CREATE SEQUENCE IF NOT EXISTS public.userid_sequence
    INCREMENT 1
    START 1
    MINVALUE 1
    MAXVALUE 9223372036854775807
    CACHE 1;
```

event_id sequence:

```
CREATE SEQUENCE IF NOT EXISTS public.eventid_sequence
    INCREMENT 1
    START 1
    MINVALUE 1
    MAXVALUE 9223372036854775807
    CACHE 1;
```

content_id sequence:

```
CREATE SEQUENCE IF NOT EXISTS public.contentid_sequence
    INCREMENT 1
    START 1
    MINVALUE 1
    MAXVALUE 9223372036854775807
    CACHE 1;
```

check_event_count trigger:

```
CREATE OR REPLACE FUNCTION public.check_event_count()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
DECLARE
    user_count INTEGER;
BEGIN
    -- Check if the user exists in free_users
    IF EXISTS (SELECT 1 FROM free_users WHERE userid = NEW.userID) THEN
        -- Get the count of events for the user
        SELECT count INTO user_count FROM free_users WHERE userid = NEW.userID;

        -- Check if the count is already 2
        IF user_count >= 2 THEN
            -- Cancel the insert operation
            RAISE EXCEPTION 'User already has 2 events';
        ELSE
            -- Increment the event count for the user
            UPDATE free_users SET count = user_count + 1 WHERE userid = NEW.userID;
        END IF;
    ELSE
        -- User does not exist => User is a premium user
    END IF;

    -- If all checks pass, allow the insert operation
    RETURN NEW;
END;
$BODY$;
```