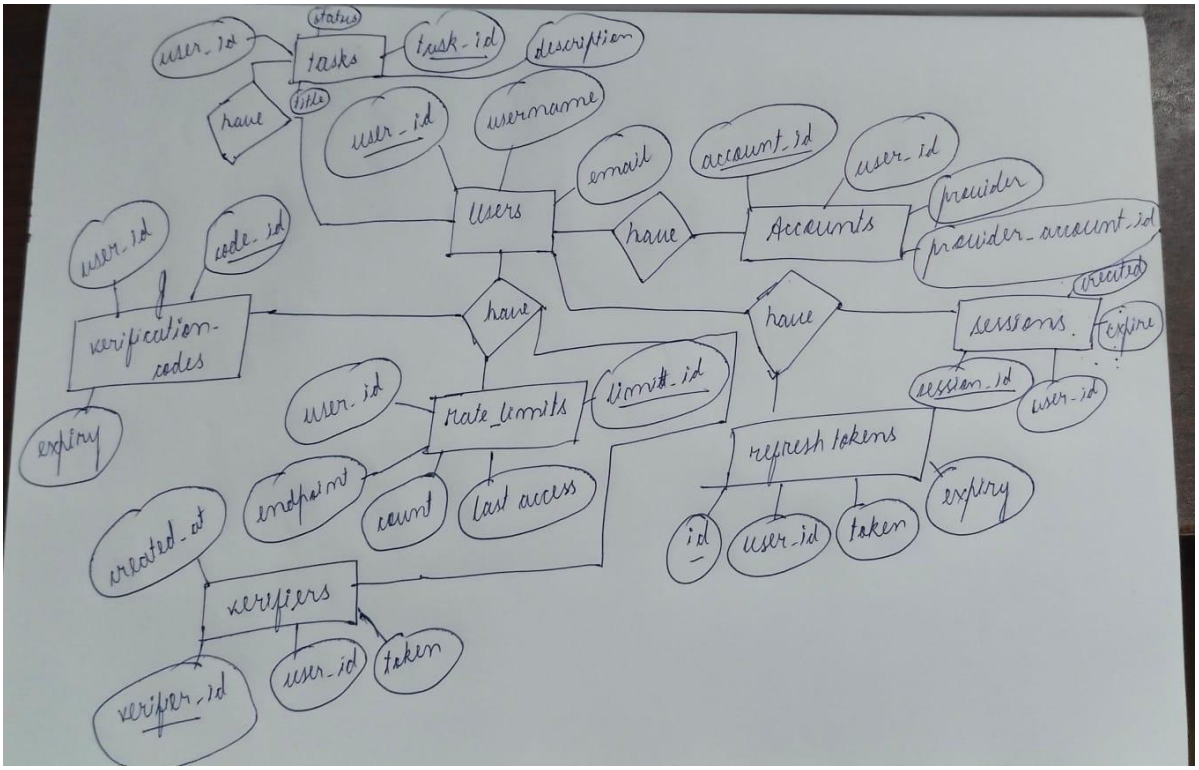
Diagrams and Database codes for the to do list

By

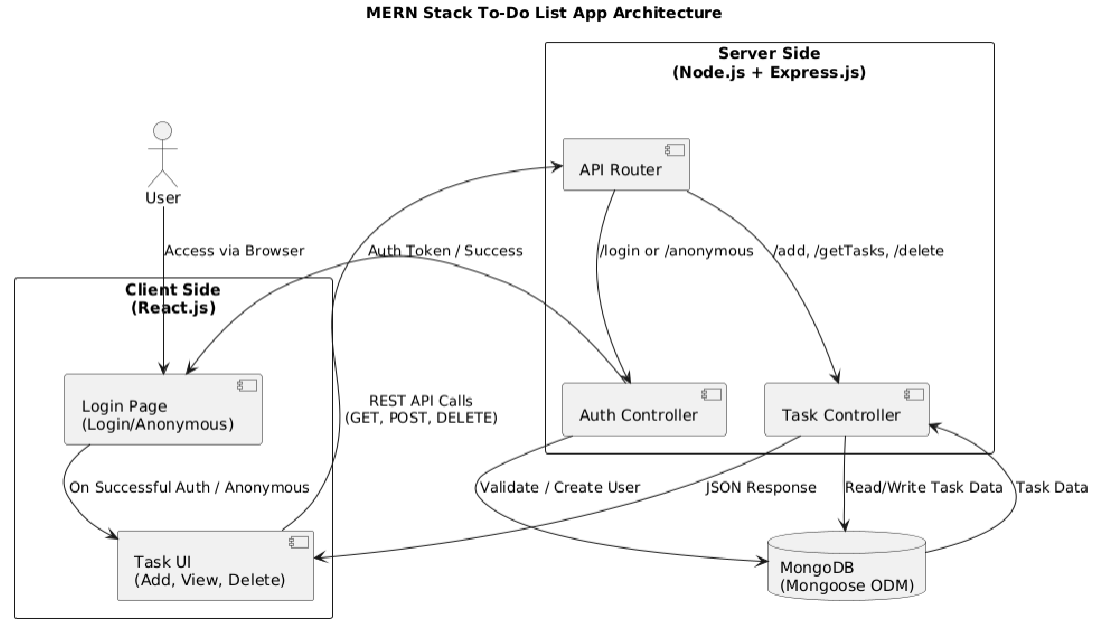
Name- Baibhab Das

Registration number – 23BCE8460

ER diagram for the to do list project



Architecture diagram for the to do list project



Creating the required databases for my project using MySQL

By

Name- Baibhab Das

Registration number- 23BCE8460

Queries for creating the required tables

```sql
-- USERS Table
CREATE TABLE users (
    id INT PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    email VARCHAR(100) NOT NULL UNIQUE
);

-- ACCOUNTS Table
CREATE TABLE accounts (
    id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT,
    provider VARCHAR(50),
    provider_account_id VARCHAR(100),
    FOREIGN KEY (user_id) REFERENCES users(id)
);

-- RATE_LIMITS Table
CREATE TABLE rate_limits (
    id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT,
    endpoint VARCHAR(100),
    count INT DEFAULT 0,
    last_access DATETIME,
    FOREIGN KEY (user_id) REFERENCES users(id)
);

-- REFRESH_TOKENS Table
CREATE TABLE refresh_tokens (
    id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT,
    token TEXT,
    expiry DATETIME,
    FOREIGN KEY (user_id) REFERENCES users(id)
);

-- SESSIONS Table
CREATE TABLE sessions (
    id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT,
    session_token VARCHAR(255),
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    expires_at DATETIME,
    FOREIGN KEY (user_id) REFERENCES users(id)
);

-- VERIFICATION_CODES Table
CREATE TABLE verification_codes (
    id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT,
    code VARCHAR(10),
    expires_at DATETIME,
    FOREIGN KEY (user_id) REFERENCES users(id)
);

-- VERIFIERS Table
CREATE TABLE verifiers (
    id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT,
    verifier_token VARCHAR(255),
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id)
);

-- TASKS Table
CREATE TABLE tasks (
    id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT,
    title VARCHAR(255),
    description TEXT,
    status ENUM('pending', 'completed') DEFAULT 'pending',
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    due_date DATETIME,
    FOREIGN KEY (user_id) REFERENCES users(id)
);
```

Queries for inserting data into these tables

```sql
-- USERS
INSERT INTO users (id, username, email) VALUES
(1, 'ktboajjb', 'gzntzu@example.com'),
(2, 'bkmryhew', 'hzzxkn@example.com'),
(3, 'qovouams', 'nqflvr@example.com'),
(4, 'wcvptitj', 'szzyat@example.com'),
(5, 'jwdmvfiv', 'mfbeer@example.com');

-- ACCOUNTS
INSERT INTO accounts (user_id, provider, provider_account_id) VALUES
(1, 'google', 'g123'),
(2, 'github', 'gh234'),
(3, 'google', 'g345'),
(4, 'facebook', 'fb456'),
(5, 'github', 'gh567');

-- RATE_LIMITS
INSERT INTO rate_limits (user_id, endpoint, count, last_access) VALUES
(1, '/tasks', 10, NOW()),
(2, '/tasks', 5, NOW()),
(3, '/auth', 3, NOW()),
(4, '/tasks', 12, NOW()),
(5, '/tasks', 7, NOW());

-- REFRESH_TOKENS
INSERT INTO refresh_tokens (user_id, token, expiry) VALUES
(1, 'rtok123', NOW() + INTERVAL 7 DAY),
(2, 'rtok234', NOW() + INTERVAL 7 DAY),
(3, 'rtok345', NOW() + INTERVAL 7 DAY),
(4, 'rtok456', NOW() + INTERVAL 7 DAY),
(5, 'rtok567', NOW() + INTERVAL 7 DAY);

-- SESSIONS
INSERT INTO sessions (user_id, session_token, expires_at) VALUES
(1, 'sess123', NOW() + INTERVAL 1 HOUR),
(2, 'sess234', NOW() + INTERVAL 1 HOUR),
(3, 'sess345', NOW() + INTERVAL 1 HOUR),
(4, 'sess456', NOW() + INTERVAL 1 HOUR),
(5, 'sess567', NOW() + INTERVAL 1 HOUR);

-- VERIFICATION_CODES
INSERT INTO verification_codes (user_id, code, expires_at) VALUES
(1, 'ABC123', NOW() + INTERVAL 5 MINUTE),
(2, 'XYZ234', NOW() + INTERVAL 5 MINUTE),
(3, 'LMN345', NOW() + INTERVAL 5 MINUTE),
(4, 'QWE456', NOW() + INTERVAL 5 MINUTE),
(5, 'RTY567', NOW() + INTERVAL 5 MINUTE);

-- VERIFIERS
INSERT INTO verifiers (user_id, verifier_token) VALUES
(1, 'veri123'),
(2, 'veri234'),
(3, 'veri345'),
(4, 'veri456'),
(5, 'veri567');

-- TASKS
INSERT INTO tasks (user_id, title, description, status, due_date) VALUES
(1, 'Buy groceries', 'Milk, eggs, bread', 'pending', NOW() + INTERVAL 2 DAY),
(2, 'Finish project', 'Submit the MERN task app', 'pending', NOW() + INTERVAL 3 DAY),
(3, 'Call plumber', 'Fix kitchen sink leak', 'completed', NOW() - INTERVAL 1 DAY),
(4, 'Book flight', 'Trip to Delhi', 'pending', NOW() + INTERVAL 5 DAY),
(5, 'Pay bills', 'Electricity and internet', 'completed', NOW() - INTERVAL 2 DAY);
```

Data Manipulation Queries

```sql
-- INSERT
INSERT INTO tasks (user_id, title, description, status, due_date)
VALUES (1, 'Read book', 'Read Clean Code', 'pending', NOW() + INTERVAL 4 DAY);

-- UPDATE
UPDATE tasks SET status = 'completed' WHERE id = 1;

-- DELETE
DELETE FROM sessions WHERE user_id = 5;
```

Data retrieval queries

```sql
-- 1. Retrieve all tasks for a specific user
SELECT * FROM tasks WHERE user_id = 1;

-- 2. Join users and tasks to show task title and user email
SELECT u.username, u.email, t.title, t.status
FROM users u
JOIN tasks t ON u.id = t.user_id;

-- 3. Group tasks by status
SELECT status, COUNT(*) AS task_count
FROM tasks
GROUP BY status;

-- 4. Order users by number of tasks assigned
SELECT u.username, COUNT(t.id) AS task_count
FROM users u
LEFT JOIN tasks t ON u.id = t.user_id
GROUP BY u.id
ORDER BY task_count DESC;

-- 5. List sessions with user info
SELECT s.session_token, s.created_at, u.username
FROM sessions s
JOIN users u ON s.user_id = u.id;
```

Output:-

Output:

```
+----+---------+----------------+--------------------+-----------+----------------------+----------------------+
| id | user_id | title          | description        | status    | created_at           | due_date             |
+----+---------+----------------+--------------------+-----------+----------------------+----------------------+
|  1 |       1 | Buy groceries | Milk, eggs, bread | completed | 2025-05-22 13:18:34 | 2025-05-24 13:18:34 |
|  6 |       1 | Read book      | Read Clean Code    | pending   | 2025-05-22 13:18:34 | 2025-05-26 13:18:34 |
+----+---------+----------------+--------------------+-----------+----------------------+----------------------+
+----------+---------------------+-----------------+-----------+
| username | email               | title           | status    |
+----------+---------------------+-----------------+-----------+
| ktboajjb | gzntzu@example.com | Buy groceries   | completed |
| ktboajjb | gzntzu@example.com | Read book       | pending   |
| bkmryhew | hzzxkn@example.com | Finish project | pending   |
| qovouams | nqflvr@example.com | Call plumber    | completed |
| wcvptitj | szzyat@example.com | Book flight     | pending   |
| jwdmvfiv | mfbeer@example.com | Pay bills       | completed |
+----------+---------------------+-----------------+-----------+
+-----------+------------+
| status    | task_count |
+-----------+------------+
| completed |          3 |
| pending   |          3 |
+-----------+------------+
+----------+------------+
| username | task_count |
+----------+------------+
| ktboajjb |          2 |
| bkmryhew |          1 |
| qovouams |          1 |
| wcvptitj |          1 |
| jwdmvfiv |          1 |
+----------+------------+
+---------------+---------------------+----------+
| session_token | created_at          | username |
+---------------+---------------------+----------+
| sess123       | 2025-05-22 13:18:34 | ktboajjb |
| sess234       | 2025-05-22 13:18:34 | bkmryhew |
| sess345       | 2025-05-22 13:18:34 | qovouams |
| sess456       | 2025-05-22 13:18:34 | wcvptitj |
+---------------+---------------------+----------+
```

Complex queries

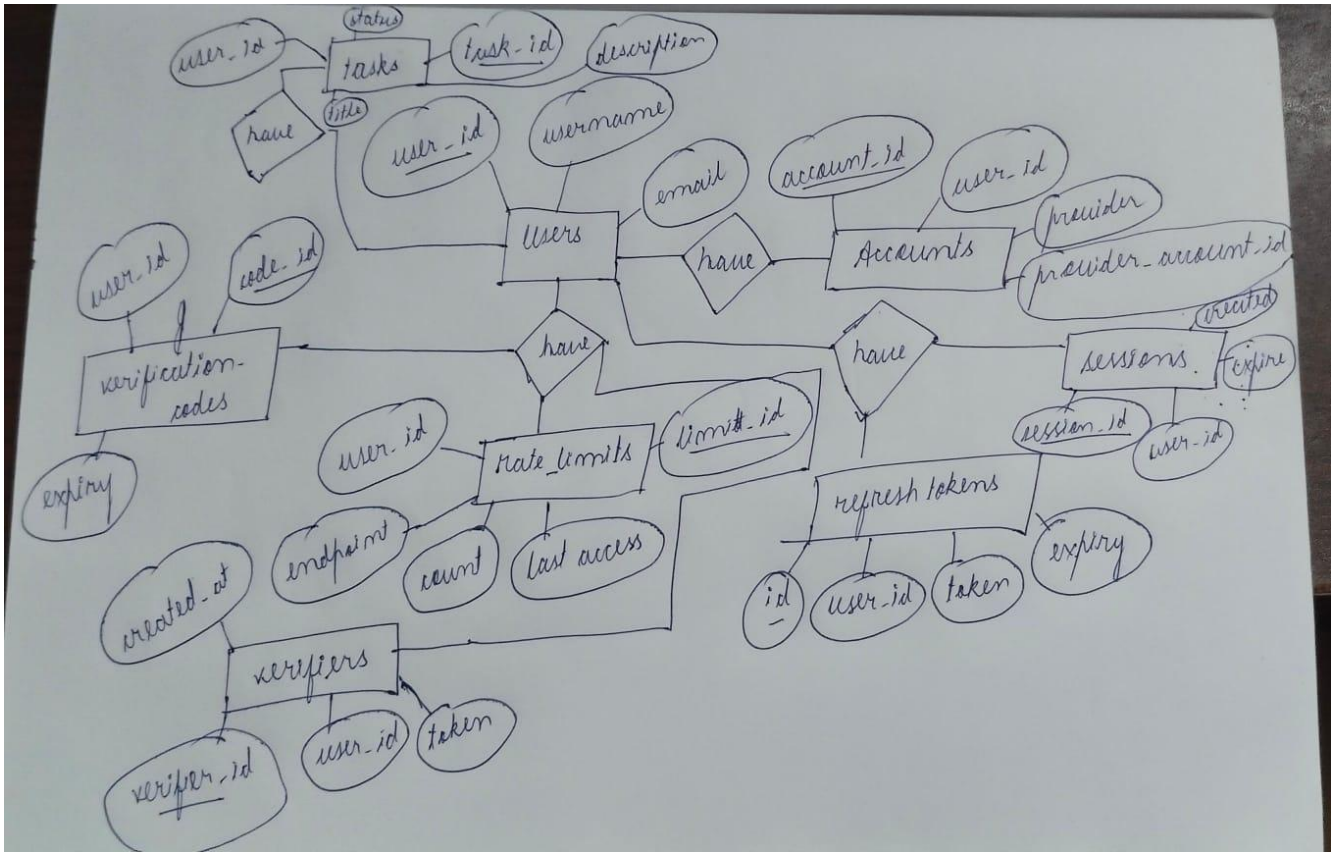
```sql
-- 1. Get users who have more than one pending task
SELECT user_id
FROM tasks
WHERE status = 'pending'
GROUP BY user_id
HAVING COUNT(*) > 1;

-- 2. Find users who haven't logged in (no session)
SELECT u.username
FROM users u
LEFT JOIN sessions s ON u.id = s.user_id
WHERE s.id IS NULL;
```

Output

```
+----------+
| username |
+----------+
| jwdmvfiv |
+----------+
```

ER DIAGRAM


tasks
status
user_id
task_id
description
title
have
Users
user_id
username
email
have
Accounts
account_id
user_id
provider
provider_account_id
verification-codes
user_id
code_id
expiry
have
have
sessions
created
expire
session_id
user_id
rate_limits
user_id
limit_id
endpoint
count
last access
refresh tokens
id
user_id
token
expiry
verifiers
created_at
verifier_id
user_id
token