Institute of
Space Technology

DATABASE SYSTEMS

Pomodoro Timer

Project Report

Ayman Shaheen 241201060
Nabiha Noor 241201052

Project Submission: 15th June ,2025

Table of Contents

1) Introduction

The Pomodoro Timer is a productivity tool that is designed to help users manage their time more efficiently using the well-known and widely used Pomodoro Technique. This method allows users to work in 25-minute sessions followed by short breaks, improving focus and reducing burnout. The project allows users to customize session lengths, track completed tasks, and monitor their productivity history.

We built our project using HTML, CSS, JavaScript, PHP, and a MySQL backend, the project also supports user registration and login functionality and offers a personalized experience and stores session data securely. By combining functionality with a calming and clean UI, pomodoro transforms time management into a more engaging and productive process.

Problem Statement

In today's rapidly changing environment, many people struggle with consistent concentration, managing time properly, and avoiding exhaustion. Procrastination, multitasking, and frequently occurring distractions lead to reduced productivity and mental as well as emotional fatigue. Traditional time management methods often lack what it takes for users to stay on a task.

Real-Life Use Case

Scenario:

We can take our own example for this. Being university students its critical for us to stay on track and not get distracted. Even if we try to focus in class, we can feel our minds getting heavy and exhausted after 25 minutes maximum. This is exactly how pomodoro is useful.

How the app helps us:

- We log in the tasks that need to be done.
- We start a 25-minute work session for our selected task.
- After each session, we get notified to take a short break.
- Our session data gets stored and we can view the displayed data to track our productivity.
- Motivational quotes keep us going.

This real-life use case shows how the Pomodoro Timer is highly effective when it comes to time management.

2) Requirements Analysis

• Functional Requirements

a) User Authentication

- sign up and log in with a username and password.

b) Pomodoro Timer Functionality

- Start, pause, resume work sessions with 5-minute breaks that can be skipped.
- Custom work/break also available

c) Task Management

- create, edit, and delete tasks also add notes with tasks
- Priority and estimated sessions are also set

d) Session Tracking

- Sessions, their duration and their dates are stored
- History page displays the completed sessions

e) Goals Feature

- Set goals, view them, delete them and mark them as achieved
- Creation and achieved date are also stored

f) Motivational Features

- Page with rotating motivational quotes

• Non-Functional Requirements

a) Usability

- Interactive and easy to use Interface
- Forms and buttons are to be clearly named and should be responsive

b) Performance

- Timer and session should respond instantly when initiated

- Pages should load quickly and smooth transition between pages

c) Reliability

- Sessions get securely stored in the database

d) Security

- Passwords must be securely hashed

e) Compatibility

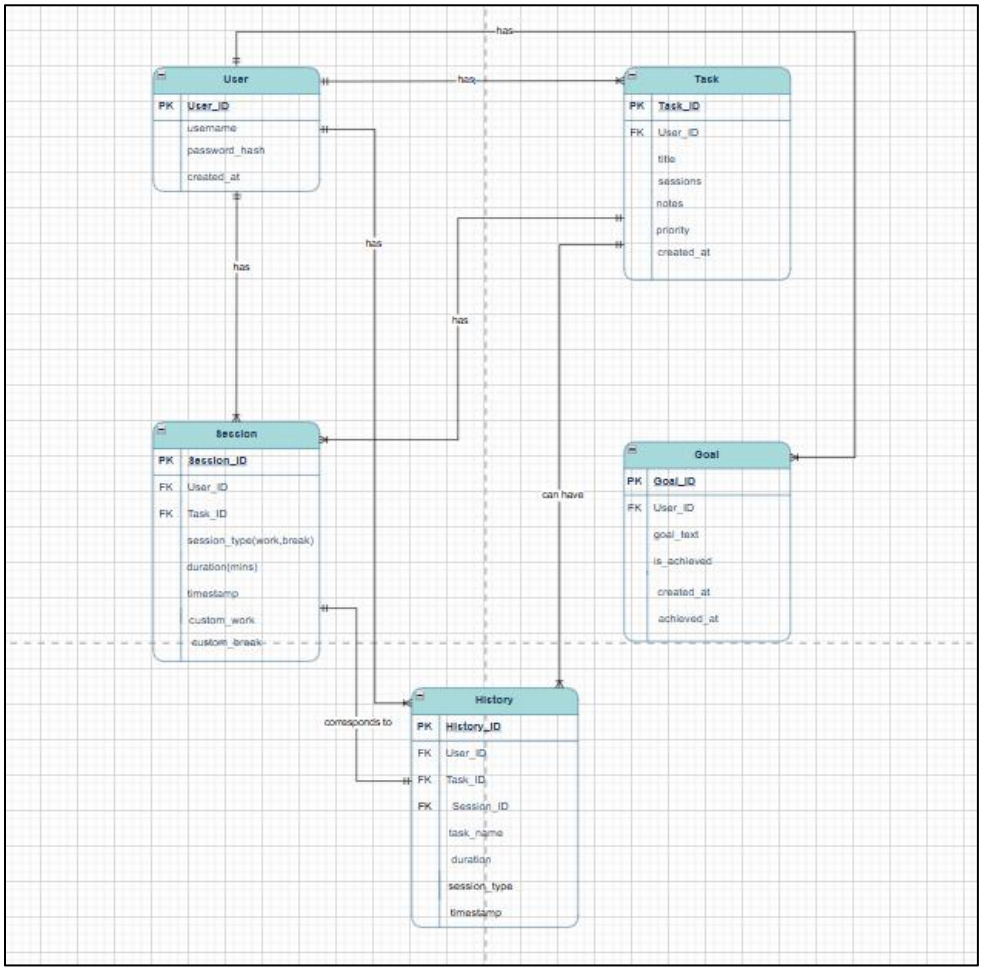
- Responsive design to ensure usability on different devices

f) Scalability

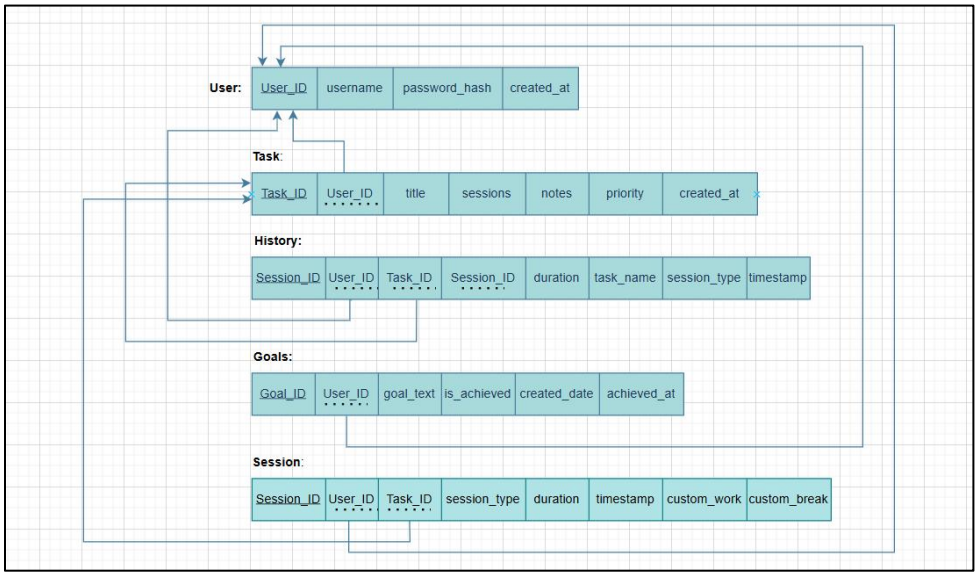
- The app should is to be designed in a modular way to allow addition of new features easily

3) ER Diagram & Relational Schema

1. ER Diagram



2. Relational Schema



3. Normalization steps

The System is already in 3NF. We can say this after ensuring the following conditions:

- **1NF**

All Tables have single valued columns and every table has its unique Primary Key (user_ID, task_ID etc).

- **2NF**

Every Non-key column depends on the full primary key.

- **3NF**

Non-key columns (e.g. title, priority) depend only on the Primary key and not on the other non-key fields.

4) Database Design

a) Tables:

1) User
2) Task
3) Session
4) History
5) Goals

b) Constraints:

1. **User:**

- **Primary Key:** user_ID
- **Unique:** username

2. **Task:**

- **Primary Key:** Task_ID
- **Foreign Key:** user_ID

3. **Session:**

- **Primary Key:** Session_ID
- **Foreign Key:**

user_ID, Task_ID

4. **History:**

- **Primary Key:** Id
- **Foreign Key:**
  Task_ID, User_ID,Session_ID

5. **Goals:**

- **Primary Key:** Id
- **Foreign Key:** user_ID

c) Relationships:

1. **users (1:N) → tasks**
   - One user can have many tasks

2. **users (1:N) → sessions**
   - One user can have many sessions

3. **tasks (1:N) → sessions**
   - One task can have many sessions

4. **users (1:N) → history**
   - One user can have many history records

5. **sessions (1:1) → history**
   - One session corresponds to one history record

6. **tasks (1:N) → history**
   - One task can have many history records

7. **users (1:N) → goals**
   - One user can have many goals

5) SQL Scripts

a) DDL

```
Press Ctrl+Enter to execute query
>
CREATE DATABASE IF NOT EXISTS pomodoro_task_manager;
USE pomodoro_task_manager;


CREATE TABLE users (
    user_id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    UNIQUE KEY (username)
);

CREATE TABLE tasks (
    task_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    title VARCHAR(255) NOT NULL,
    priority ENUM('Low', 'Medium', 'High') DEFAULT 'Low',
    sessions INT DEFAULT 1,
    notes TEXT,
    created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE
);
```

```
CREATE TABLE sessions (
    session_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    task_id INT,
    session_type ENUM('work', 'break'),
    duration INT,
    custom_work_duration INT,
    custom_break_duration INT,
    timestamp TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE SET NULL,
    FOREIGN KEY (task_id) REFERENCES tasks(task_id) ON DELETE SET NULL
);
```

```
CREATE TABLE history (
    history_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    session_id INT NOT NULL,
    task_id INT NOT NULL,
    task_name VARCHAR(255) NOT NULL,
    session_type ENUM('work', 'short_break', 'long_break') NOT NULL,
    duration INT NOT NULL,
    timestamp TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE,
    FOREIGN KEY (session_id) REFERENCES sessions(session_id) ON DELETE CASCADE,
    FOREIGN KEY (task_id) REFERENCES tasks(task_id) ON DELETE CASCADE
);
```

```sql
CREATE TABLE goals (
    goal_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    goal_text VARCHAR(255) NOT NULL,
    is_achieved TINYINT(1) DEFAULT 0,
    created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    achieved_at TIMESTAMP NULL DEFAULT NULL,
    FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE
);
```

b) DML:

- **Add Task**

```php
$stmt = $pdo->prepare(query: "INSERT INTO tasks (user_id, title, priority, sessions, notes) VALUES (?, ?,
```

- **Goals**

```php
try {
    if ($method === 'GET') {
        $stmt = $pdo->prepare(query: "SELECT * FROM goals WHERE user_id = ? ORDER BY created_at DESC");
        $stmt->execute(params: [$user_id]);
        echo json_encode(value: $stmt->fetchAll(mode: PDO::FETCH_ASSOC));

    } elseif ($method === 'POST') {
        $goal_text = $input['goal_text'] ?? '';
        $stmt = $pdo->prepare(query: "INSERT INTO goals (user_id, goal_text, is_achieved) VALUES (?, ?, 0)");
        $stmt->execute(params: [$user_id, $goal_text]);
        echo json_encode(value: ['message' => 'Goal added']);

    } elseif ($method === 'PUT') {
        $goal_id = $input['goal_id'] ?? 0;
        $is_achieved = $input['is_achieved'] ?? 0;
        if ($is_achieved) {
            $stmt = $pdo->prepare(query: "UPDATE goals SET is_achieved = 1, achieved_at = CURRENT_TIMESTAMP WHERE
        } else {
            $stmt = $pdo->prepare(query: "UPDATE goals SET is_achieved = 0, achieved_at = NULL WHERE goal_id = ?
        }
        $stmt->execute(params: [$goal_id, $user_id]);
        echo json_encode(value: ['message' => 'Goal updated']);
    } elseif ($method === 'DELETE') {
        parse_str(string: file_get_contents(filename: "php://input"), result: &$data);
        $goal_id = $data['goal_id'] ?? null;

        if ($goal_id === 'all_achieved') {
            $stmt = $pdo->prepare(query: "DELETE FROM goals WHERE user_id = ? AND is_achieved = 1");
            $stmt->execute(params: [$user_id]);
            echo json_encode(value: ['message' => 'Achieved goals cleared']);
        } else {
            $stmt = $pdo->prepare(query: "DELETE FROM goals WHERE goal_id = ? AND user_id = ?");
            $stmt->execute(params: [$goal_id, $user_id]);
            echo json_encode(value: ['message' => 'Goal deleted']);
        }

    } else {
        http_response_code(response_code: 405);
        echo json_encode(value: ['error' => 'Method not allowed']);
    }
```

- **Get Tasks**

```php
$stmt = $pdo->prepare(query: "SELECT * FROM tasks WHERE user_id = ? ORDER BY created_at DESC");
```

- **Log Sessions**

```
try {
    $stmt = $pdo->prepare(query: "
        INSERT INTO sessions (user_id, task_id, session_type, duration, custom_work_duration, custom_break_durat
        VALUES (:user_id, :task_id, :session_type, :duration, :custom_work_duration, :custom_break_duration)
    ");
```

- **Login**

```
$stmt = $pdo->prepare(query: "SELECT user_id, password_hash FROM users WHERE username = ?");
```

- **Get User**

```
try {
    $stmt = $pdo->prepare(query: "SELECT username, created_at FROM users WHERE user_id = :user_id");
```

- **Get Tasks**

```
try {
    $stmt = $pdo->prepare(query: "SELECT task_id, title, sessions FROM tasks WHERE user_id = ?");
```

- **Delete Tasks**

```
$stmt = $pdo->prepare(query: "DELETE FROM tasks WHERE task_id = ? AND user_id = ?");
```

- **Update Tasks**

```
$stmt = $pdo->prepare(query: "UPDATE tasks SET title = ?, priority = ?, sessions = ?, notes = ? WHERE ta
```

- **Signup**

```
$stmt = $pdo->prepare(query: "SELECT user_id FROM users WHERE username = ?");
```

- **Select History**

```
try {
    $stmt = $pdo->prepare(query: "
        SELECT task_name, session_type, duration, timestamp
        FROM history
        WHERE user_id = :user_id
        ORDER BY timestamp DESC
    ");
    $stmt->execute(params: ['user_id' => $user_id]);
    $history = $stmt->fetchAll(mode: PDO::FETCH_ASSOC);
```

- **Save Sessions**

```
$stmt = $pdo->prepare(query: "
    INSERT INTO sessions
        (user_id, task_id, session_type, duration, custom_work_duration, custom_break_duration, timestamp)
    VALUES
        (?, ?, ?, ?, ?, ?, NOW())
");
```

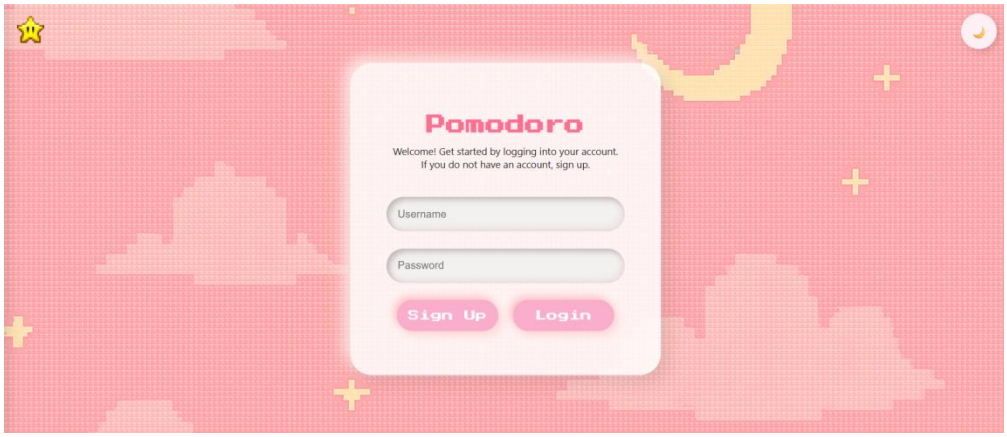
- **Save History**

```
$history = $pdo->prepare(query: "
    INSERT INTO history
        (user_id, session_id, task_id, task_name, session_type, duration, timestamp)
    VALUES
        (:user_id, :session_id, :task_id, :task_name, :session_type, :duration, NOW())
");
```

6) Application Interface

Front-End:

- **Light mode**



- **Dark mode**

Description:

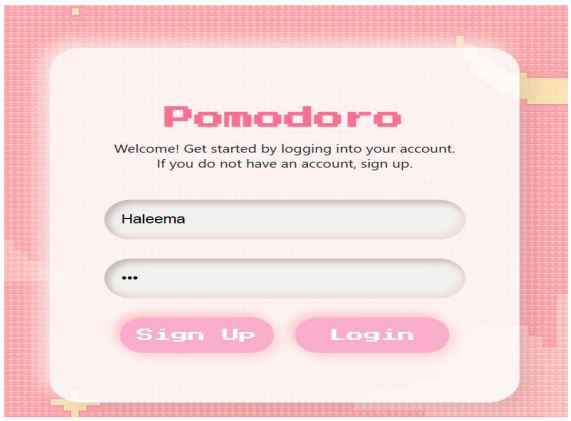
The front end of the Pomodoro Timer is a soft, pixel-art themed design with a cozy and attractive pink background. The background includes retro-style animations giving it a welcoming and calming aesthetic that is perfect for our project.

- **Pixel Theme:**
  The website features a pixel theme and the color scheme is bright and calming.
- **Clear typography**:
  Readable text hierarchy with the timer as the main focus.
- **Logical Arrangement**:
  Logical arrangement of timer controls and task editor

7) Testing & Results

1) Sample inputs and outputs

Sign Up:



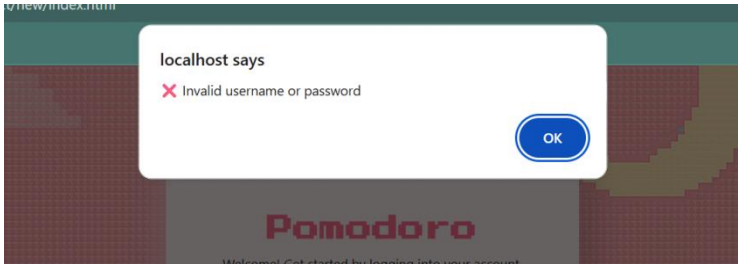
Result:

Add Task:

Results:

2) Validation examples

Password Match:



Only INT values for timer:

8) Challenges & Learnings

1) Challenges Faced

- **Database Connection**: Allowing updates between the frontend timer and backend session data required careful and thorough planning
- **User Authentication(Login)**:Securing password using hash
- **Real Time Timer Data Storage:** Using proper php

2) Key Learnings

- **Backend-Frontend Integration**: Learned about the connection between frontend(using html css) and backend(using php and mySQL)
- **Team Collaboration**: Git working and modular coding allowed us to work efficiently as a team.

9) Conclusion & Future Improvements

o Conclusion

The Pomodoro Timer successfully provides a functional and interactive web app with task logging, session implementation, and displaying proper session history. Its secure authentication and responsive design meets the core functionality and showcases the need of using this timer in this fast moving world.

o Future Enhancements

- **Advanced Analysis**: To allow visual productivity tracking with charts
- **Social Features**: Add shared goals and allow connecting with friends
- **Reduced Distractions** : Block or reduce distractions during work sessions.