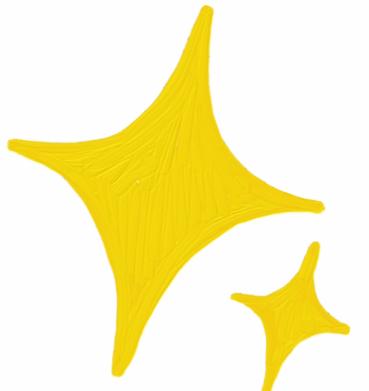


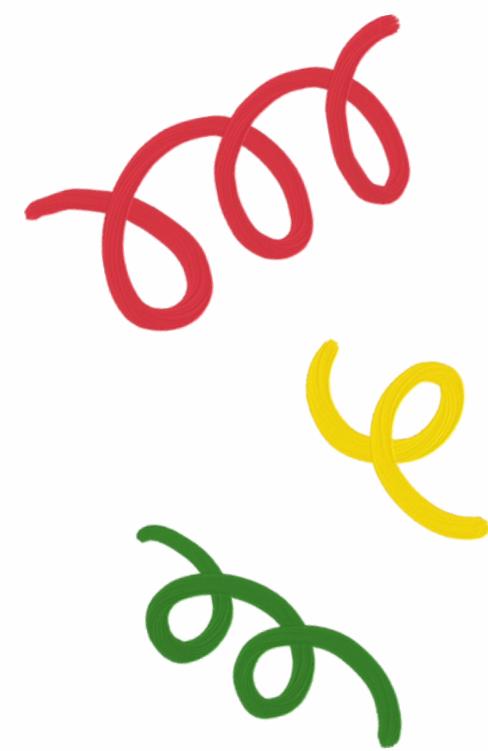
# OPEN SOURCE PRACTICE

LOVE

413855593 加藤大地  
410855281 安藤輝翔  
413855171 李婕亞  
413855114 耶律楚材



# 1.SQL Part



# create.sql

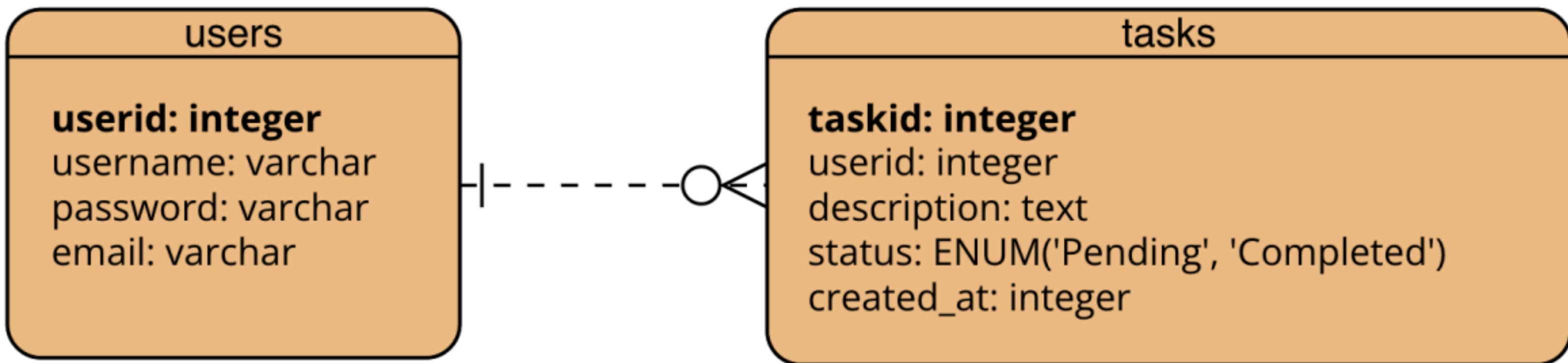
```
1
2 CREATE TABLE users (
3     user_id INT AUTO_INCREMENT PRIMARY KEY,
4     username VARCHAR(50) NOT NULL,
5     password VARCHAR(255) NOT NULL,
6     email VARCHAR(100) UNIQUE
7 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
8
9
10 CREATE TABLE tasks (
11     task_id INT AUTO_INCREMENT PRIMARY KEY,
12     user_id INT NOT NULL,
13     description TEXT NOT NULL,
14     status ENUM('pending', 'completed') DEFAULT 'pending',
15     created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
16     FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE
17 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

# test\_data.sql

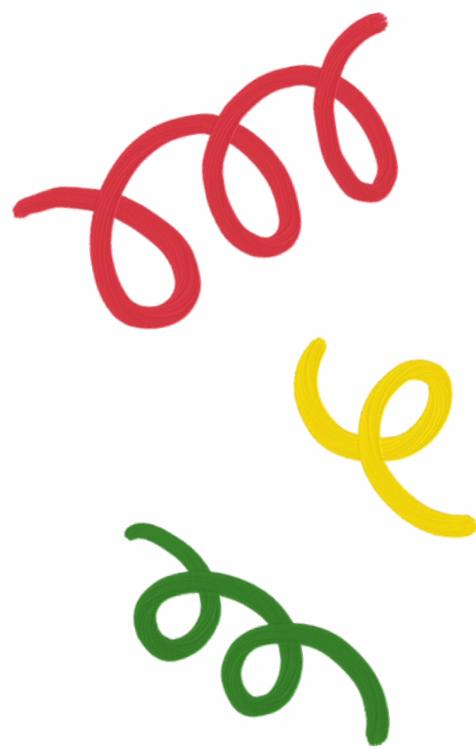
```
1  -- Insert users
2  INSERT INTO users (username, password, email) VALUES
3  ('hoge', 'hashed_password1', 'hoge@example.com'),
4  ('piyo', 'hashed_password2', 'piyo@example.com'),
5  ('kiki', 'hashed_password3', 'kiki@example.com'),
6  ('tutu', 'hashed_password4', 'tutu@example.com');
7
8  -- Insert tasks
9  INSERT INTO tasks (user_id, description, status) VALUES
10
11 -- User 1: hoge
12 (1, 'Research Chinese literature report at the library', 'pending'),
13 (1, 'Submit English reading comprehension assignment', 'completed'),
14 (1, 'Check exam coverage for midterm', 'pending'),
15 (1, 'Practice traditional Chinese character typing', 'completed'),
16
17 -- User 2: piyo
18 (2, 'Record living expenses (using budgeting app)', 'completed'),
19 (2, 'Check balance of transportation IC card (EasyCard)', 'completed'),
20
21 -- User 3: kiki
22 (3, 'Submit Python exercise assignment', 'pending'),
23 (3, 'Prepare for international student welcome event', 'completed'),
24 (3, 'Organize GitHub account', 'pending'),
25
26 -- User 4: tutu
27 (4, 'Submit experiment report', 'completed'),
28 (4, 'Schedule meeting with academic advisor', 'pending'),
29 (4, 'Summarize experiment data in Excel', 'completed');
```

---

# Entity Relationship Diagram (ERD)



# 2. PHP Part



# db.php

```
1 <?php
2
3 You, 22 時間前 | 1 author (You)
4 class Database
5 {
6     private PDO $pdo;
7
8     public function __construct()
9     {
10         $dsn = 'mysql:host=127.0.0.1;port=3301;dbname=todoApp;charset=utf8mb4';
11         $username = 'root';
12         $password = 'fo39ajf3';
13
14         try {
15             $this->pdo = new PDO($dsn, $username, $password);
16         } catch (PDOException $e) {
17             die('Connection failed: ' . $e->getMessage());
18         }
19     }
20
21     public function getPdo(): PDO
22     {
23         return $this->pdo;
24     }
25 }
26
27 $db = new Database();
28 $pdo = $db->getPdo();
29
30 session_start();
```

# login.php

```
1 <?php
2 require 'db.php';
3
4 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
5     $username = isset($_POST['username']) ? trim($_POST['username']) : '';
6     $password = isset($_POST['password']) ? $_POST['password'] : '';
7
8     if ($username === '') {
9         $errorMessage = "Username is required.";
10        exit;
11    }
12
13    $stmt = $pdo->prepare("SELECT * FROM users WHERE username = ?");
14    $stmt->execute([$username]);
15    $user = $stmt->fetch(PDO::FETCH_ASSOC);
16
17    if ($user && $password === $user['password']) {
18        $_SESSION['user_id'] = $user['user_id'];
19        header('Location: index.php');
20        exit();
21    } else {
22        $errorMessage = 'Login failed: Incorrect username or password.';
23    }
24 }
25 ?>
26
```

# index.php

```
-----  
1 <?php  
2 require 'db.php';  
3  
4 if (!isset($_SESSION['user_id'])) {  
5     header('Location: login.php');  
6     exit();  
7 }  
8  
9 $stmt = $pdo->prepare("SELECT * FROM tasks WHERE user_id = ? ORDER BY created_at DESC");  
10 $stmt->execute([$_SESSION['user_id']]);
11 $tasks = $stmt->fetchAll(PDO::FETCH_ASSOC);
12 ?>
13
```

# add\_task.php

```
1 <?php
2 require 'db.php';
3
4 if (!isset($_SESSION['user_id']) || empty($_POST['task_description'])) {
5     header('Location: index.php');
6     exit();
7 }
8
9 $stmt = $pdo->prepare("INSERT INTO tasks (user_id, description) VALUES (?, ?)");
10 $stmt->execute([\$_SESSION['user_id'], \$_POST['task_description']]);
11 header('Location: index.php');
12 exit();
13
```

# complete\_task.php

```
1 <?php
2 require 'db.php';
3
4 if (!isset($_SESSION['user_id']) || empty($_POST['task_id'])) {
5     header('Location: index.php');
6     exit();
7 }
8
9
10 $stmt = $pdo->prepare("UPDATE tasks SET status = 'completed' WHERE task_id = ? AND user_id = ?");
11 $stmt->execute([$POST['task_id'], $_SESSION['user_id']]);
12 header('Location: index.php');
13 exit();
14 
```

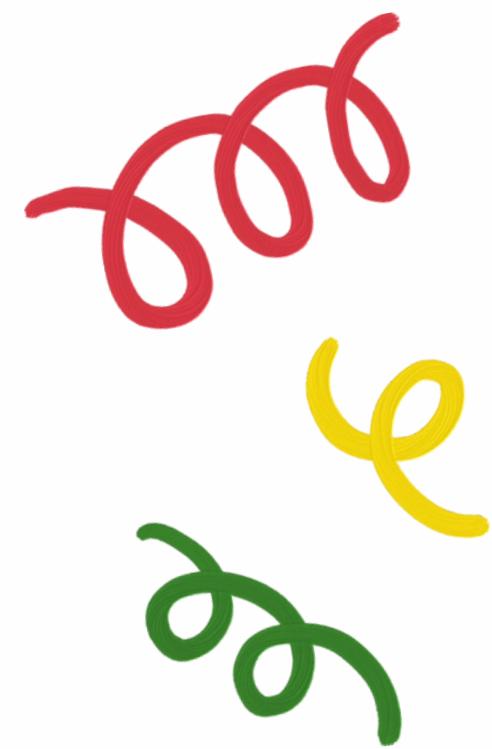
# delete\_task.php

```
1 <?php
2 require 'db.php';
3
4 if (!isset($_SESSION['user_id']) || empty($_POST['task_id'])) {
5     header('Location: index.php');
6     exit();
7 }
8
9
10 $stmt = $pdo->prepare("DELETE FROM tasks WHERE task_id = ? AND user_id = ?");
11 $stmt->execute([$_POST['task_id'], $_SESSION['user_id']]);
12 header('Location: index.php');
13 exit();
14 
```

# logout.php

```
1 <?php  
2 session_start();  
3 session_destroy();  
4 header('Location: login.php');  
5 exit();  
6
```

# 3. HTML Part



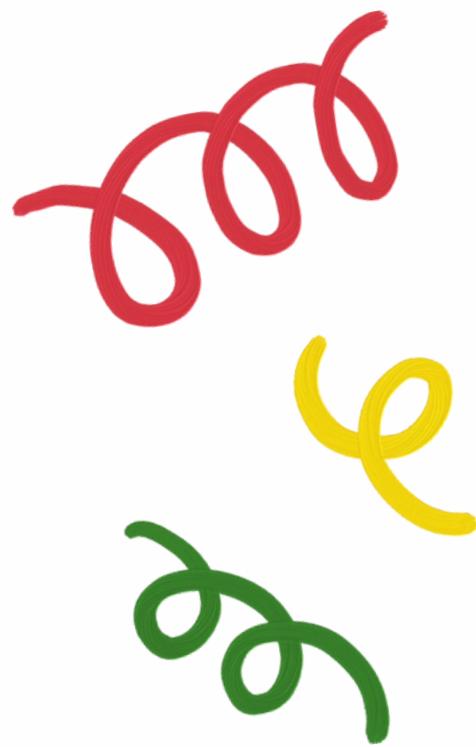
# login.php

```
114 <body>
115     <h1>Todo App Login</h1>
116
117     <?php if (!empty($errorMessage)): ?>
118         <div class="error-message"><?= htmlspecialchars($errorMessage) ?></div>
119     <?php endif; ?>
120
121     <form method="POST" autocomplete="off">
122         <input type="text" name="username" required placeholder="Username" value="<?= isset($username) ?
123             htmlspecialchars($username) : '' ?>" />
124         <input type="password" name="password" required placeholder="Password" />
125         <button type="submit">Login</button>
126     </form>
127 </body>
```

# index.php

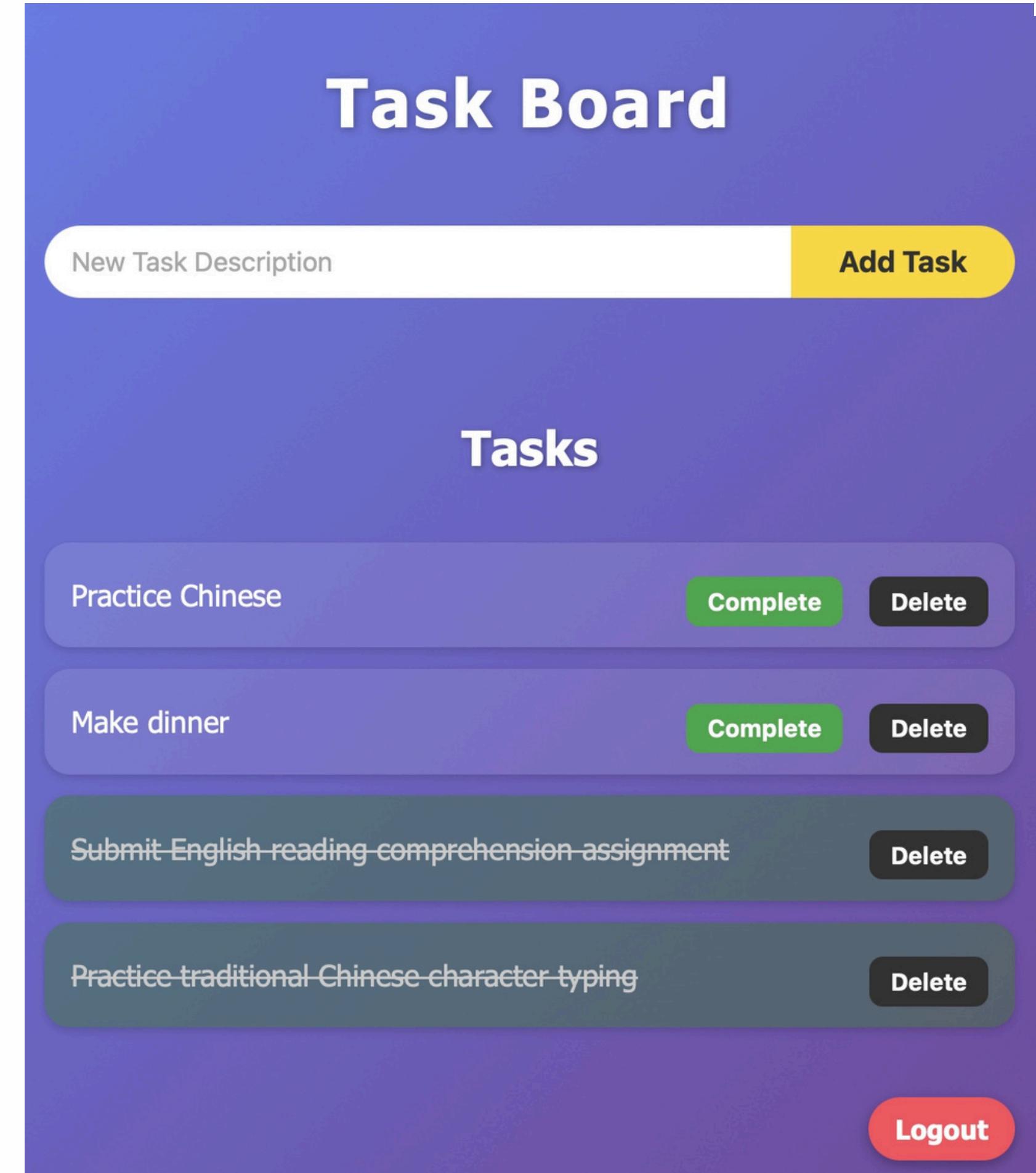
```
231 <body>
232     <h1>Task Board</h1>
233
234     <form id="addTaskForm" action="add_task.php" method="POST" autocomplete="off">
235         <input type="text" name="task_description" required placeholder="New Task Description" maxlength="255" />
236         <button type="submit">Add Task</button>
237     </form>
238
239     <h2>Tasks</h2>
240     <ul class="task-list" id="taskList">
241         <?php foreach ($tasks as $task): ?>
242             <li class="<?= $task['status'] === 'completed' ? 'completed' : '' ?>">
243                 <span class="task-desc"><?= htmlspecialchars($task['description']) ?></span>
244
245
246             <div class="btn-group">
247                 <?php if ($task['status'] === 'pending'): ?>
248                     <form action="complete_task.php" method="POST" style="display:inline;">
249                         <input type="hidden" name="task_id" value="<?= $task['task_id'] ?>">
250                         <button type="submit" class="btn-complete">Complete</button>
251                     </form>
252                 <?php endif; ?>
253                 <form action="delete_task.php" method="POST" style="display:inline;">
254                     <input type="hidden" name="task_id" value="<?= $task['task_id'] ?>">
255                     <button type="submit" class="btn-delete">Delete</button>
256                 </form>
257             </div>
258
259         </li>
260     <?php endforeach; ?>
261     </ul>
262
263     <a class="logout-link" href="logout.php">Logout</a>
264 </body>
```

# 4. CSS Part



## Key points:

- Modern and minimal layout
- Smooth user experience across devices
- Easy-to-read fonts and clear UI components
- Color scheme:
  - Purple for backgrounds
  - Yellow for action buttons
  - Green for completed tasks



- We used display: flex to align the input and button horizontally
- Rounded corners (border-radius) for a soft, friendly feel
- Shadows and focus effects (box-shadow, :focus) to guide the user
- Responsive layout with max-width for mobile compatibility

```
form#addTaskForm {  
  margin: 1.5rem 0;  
  display: flex;  
  width: 100%;  
  max-width: 600px;  
}  
  
input[name="task_description"] {  
  flex-grow: 1;  
  padding: 0.8rem 1rem;  
  border-radius: 25px 0 0 25px;  
  border: none;  
  font-size: 1rem;  
  outline: none;  
  transition: box-shadow 0.3s ease;  
}  
  
input[name="task_description"]:focus {  
  box-shadow: 0 0 10px 2px #ffd700;  
}  
  
button[type="submit"] {  
  background: #ffd700;  
  border: none;  
  padding: 0 1.8rem;  
  . . .  
}
```

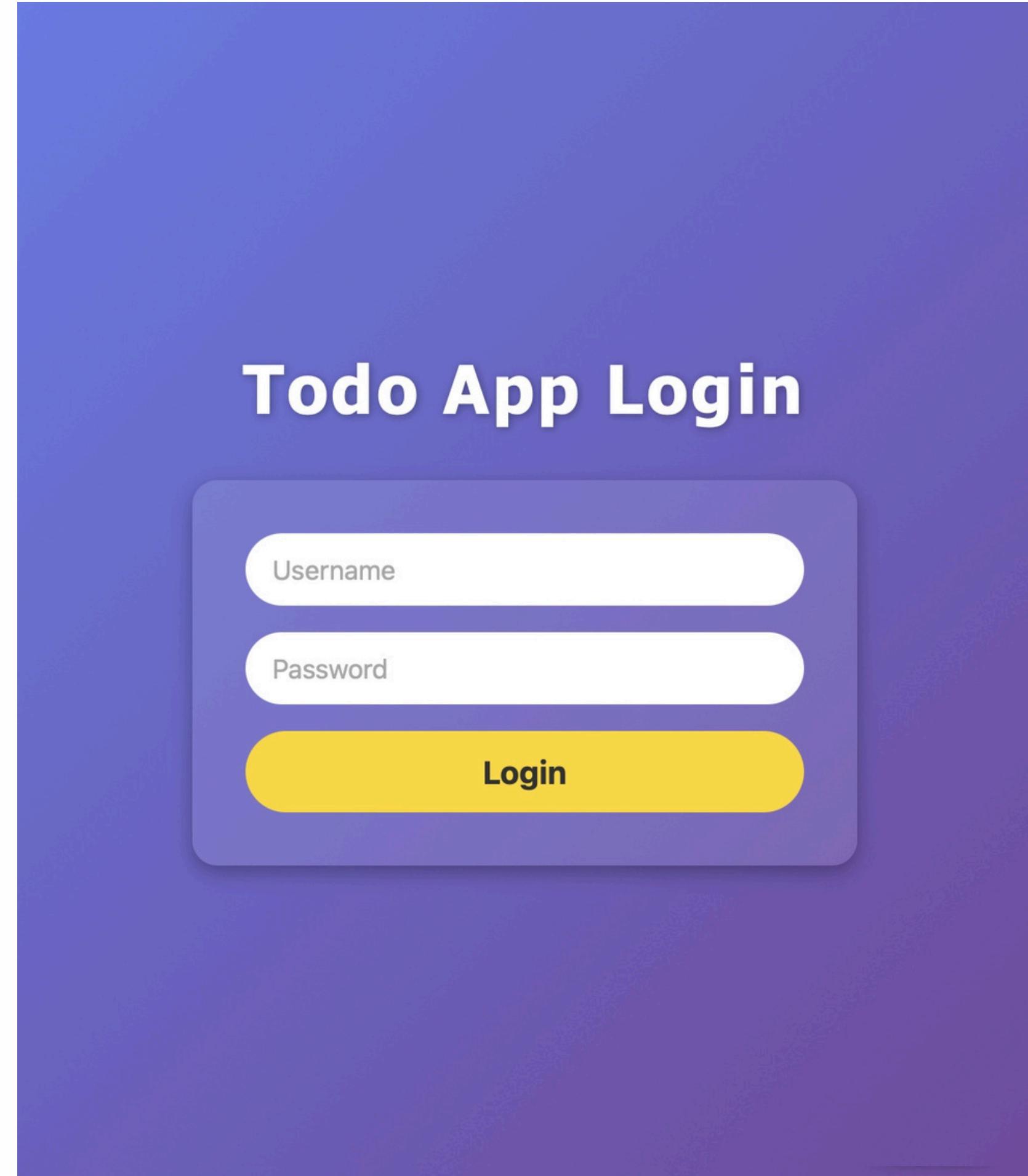
- Each task is styled as a card (li with background and shadow)
- Completed tasks have the .completed class:
- Strikethrough text
- Different color and badge
- State badges (e.g., “Pending”, “Done”) help users understand status at a glance

```
ul.task-list li {  
    background: □rgba(255, 255, 255, 0.1);  
    margin-bottom: 0.8rem;  
    border-radius: 15px;  
    padding: 0.8rem 1rem;  
    display: flex;  
    align-items: center;  
    justify-content: space-between;  
    box-shadow: 0 2px 5px □rgba(0, 0, 0, 0.15);  
    transition: background-color 0.3s ease;  
}  
  
ul.task-list li.completed {  
    background: □rgba(0, 128, 0, 0.4);  
    text-decoration: line-through;  
    color: #ccc;  
}  
  
.task-desc {  
    flex-grow: 1;  
    font-size: 1.1rem;  
    margin-right: 1rem;  
    word-break: break-word;  
}
```

- Green "Complete" button: friendly and intuitive
- Black "Delete" button: minimal but clearly visible
- Hover effects (:hover) with slight scaling (transform: scale(1.05))
- Transitions make the UI feel smooth and responsive

```
button.btn-complete {  
    background-color: #28a745;  
    color: white;  
    border: none;  
    padding: 0.4rem 0.8rem;  
    border-radius: 8px;  
    font-size: 0.9rem;  
    font-weight: bold;  
    cursor: pointer;  
    transition: background-color 0.3s ease, transform 0.1s ease;  
}  
  
button.btn-complete:hover {  
    background-color: #218838;  
    transform: scale(1.05);  
}  
  
button.btn-delete {  
    background: #333;  
    color: white;  
    border: none;  
    padding: 0.4rem 0.8rem;  
    border-radius: 8px;  
    font-size: 0.9rem;  
    font-weight: bold;  
    cursor: pointer;
```

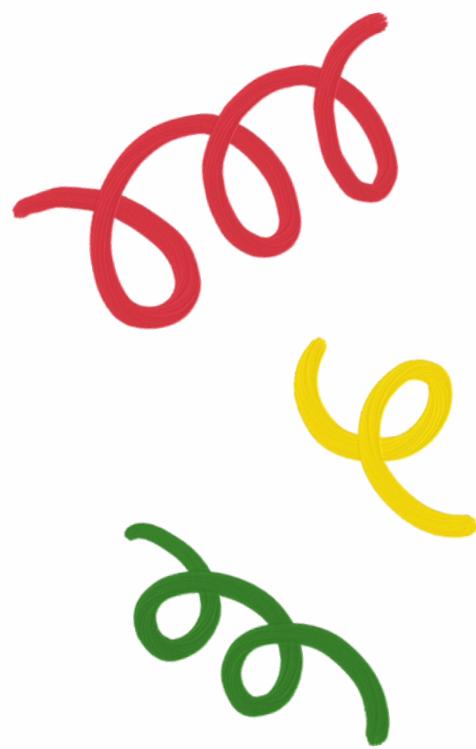
- Login box is centered using flex layout
- Background slightly transparent for a modern look
- Error messages in red for visibility and quick feedback
- Overall layout is clean to reduce distractions





# 5.Javascript

## Part



```
const deleteForms = document.querySelectorAll('form[action="delete_task.php"]');

deleteForms.forEach(form => {
    form.addEventListener('submit', (e) => {
        const li = form.closest('li');
        if (!li) return;

        if (!li.classList.contains('completed')) {
            const confirmed = window.confirm('This task is not completed. Are you
sure you want to delete it?');
            if (!confirmed) {
                e.preventDefault();
            }
        }
    });
});
```

```
const deleteForms = document.querySelectorAll('form[action="delete_task.php"]');

deleteForms.forEach(form => {
    form.addEventListener('submit', (e) => {
        const li = form.closest('li');
        if (!li) return;

        if (!li.classList.contains('completed')) {
            const confirmed = window.confirm('This task is not completed. Are you
sure you want to delete it?');
            if (!confirmed) {
                e.preventDefault();
            }
        }
    });
});
```

```
const deleteForms = document.querySelectorAll('form[action="delete_task.php"]');
```

- This selects all <form> elements on the page that are submitting to delete\_task.php and stores them in deleteForms.

```
deleteForms.forEach(form => {
```

- Loops through each of those selected forms.

```
    form.addEventListener('submit', (e) => {
```

- For each form, it listens for the submit event (when a user tries to delete a task).

```
if (!li.classList.contains('completed')) {
```

- Checks if the list item (task) does NOT have the completed class.

```
const confirmed = window.confirm('This task is not completed. Are you sure you want to  
delete it?');
```

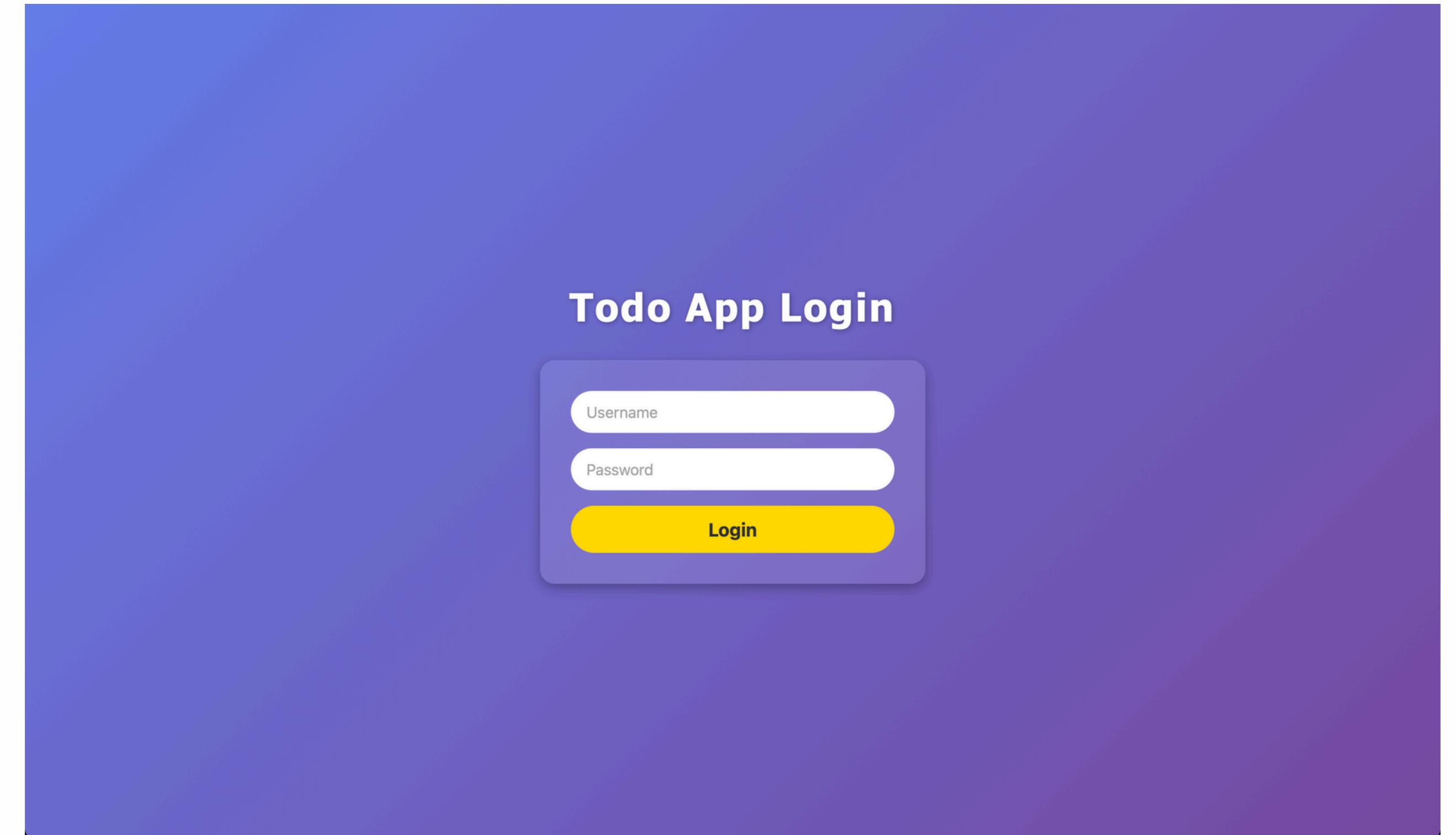
- If the task is not completed, show a confirmation popup to the user.

```
if (!confirmed) {  
  e.preventDefault();  
}
```

- If the user clicks “Cancel” (i.e. they are not sure), it cancels the form submission (e.preventDefault()), so the task won’t be deleted.

# Demonstration

- Login page where users can enter the data from `test_data.sql` to enter.



# Demonstration

- Home page where users can add and delete tasks

The image shows a screenshot of a web-based task management application titled "Task Board". The interface has a dark blue header and a light blue body. At the top, there is a search bar labeled "New Task Description" and a yellow "Add Task" button. Below the header, the word "Tasks" is centered. A list of four tasks is displayed in a grid format:

Task Description	Actions
Research Chinese literature report at the library	<button>Complete</button> <button>Delete</button>
Submit English reading comprehension assignment	<button>Delete</button>
Check exam coverage for midterm	<button>Complete</button> <button>Delete</button>
Practice traditional Chinese character typing	<button>Delete</button>

In the bottom right corner of the main area, there is a red "Logout" button.

# Thank you for paying attention!

Link to our repository: <https://github.com/Not-Kronox101/MyTaskBoard>

Link to our demonstration video: <https://youtu.be/bCv15eQM6KM>