TO-DO TASK WEB APPLICATION

Course – ITE220 (Web Development II)
Ajarn. Susanta Malakar

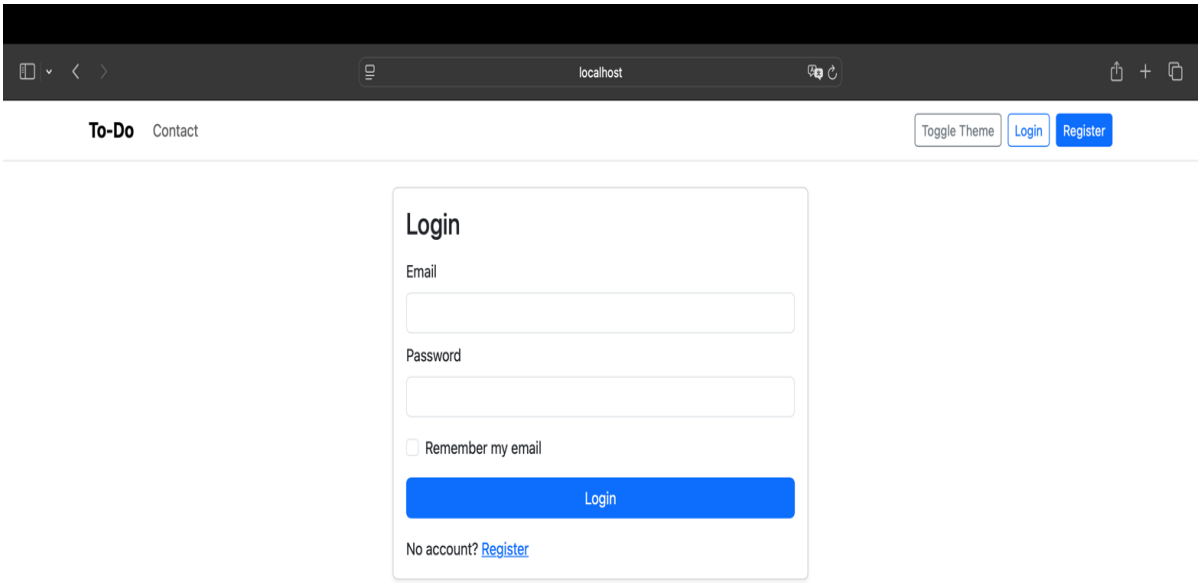
By – Su Yati Win (2302230004)
Thaw Hein (2302210002)
Sai Aung Lin (2302280007)

Contents

1. Project Objectives

The goal of this project is to build a simple and easy-to-use responsive To-Do web application. It helps users organize their daily work by adding, editing, or deleting tasks online. This project also shows how the front end and back end of a full-stack website work together using PHP, MySQL, JavaScript, and Bootstrap.

The system includes sessions to keep users logged in and cookies to remember their theme preference (dark or light) and email. The Fetch API allows smooth updates without reloading pages, making the experience fast and user-friendly.



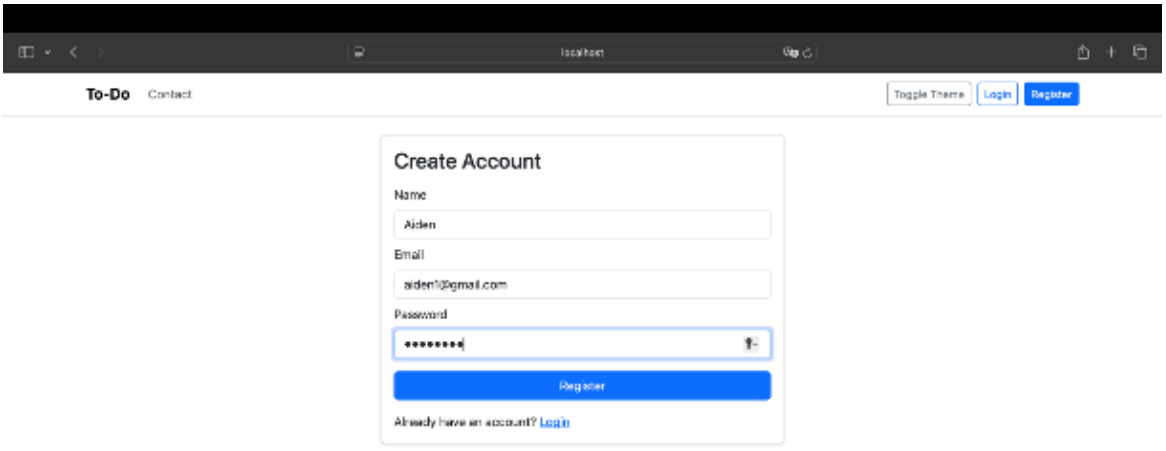
2. Functional Explanation of Core Features

This section explains in simple words how each main function works on the To-Do website. It describes how buttons and forms connect the front-end and back-end parts of the website.

Each task has a small delete icon beside it. When clicked, JavaScript sends the task ID to task_delete.php. The PHP file runs an SQL DELETE command to remove the record. Once deleted, the task disappears from the page right away using JavaScript.

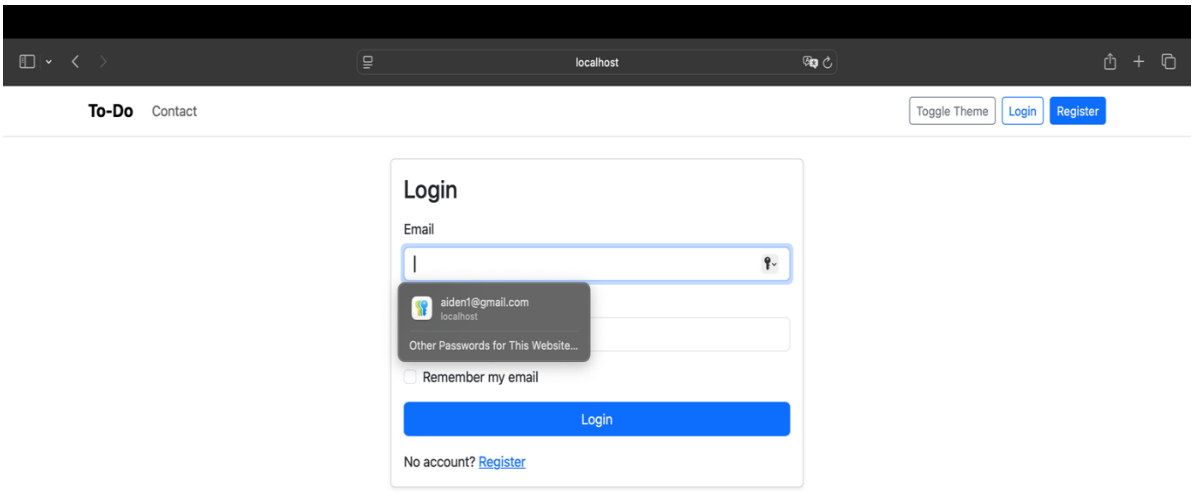
Account Creation (Register Button)

When a new user fills in their name, email, and password, the data is sent to register.php. PHP checks if the email already exists in the database. If not, it saves the new account after encrypting the password using password_hash(). If the email is already used, an error message appears.



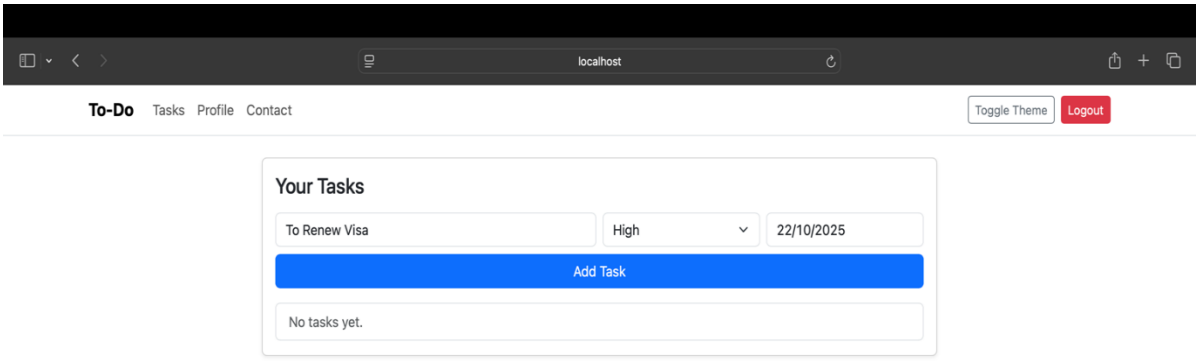
Login Function

When a user logs in, the entered email and password are sent to login.php. PHP checks the email and verifies the password with password_verify(). If correct, a session starts using $_SESSION to keep the user logged in. If the user checks 'Remember Me', their email is stored in a cookie for next time.



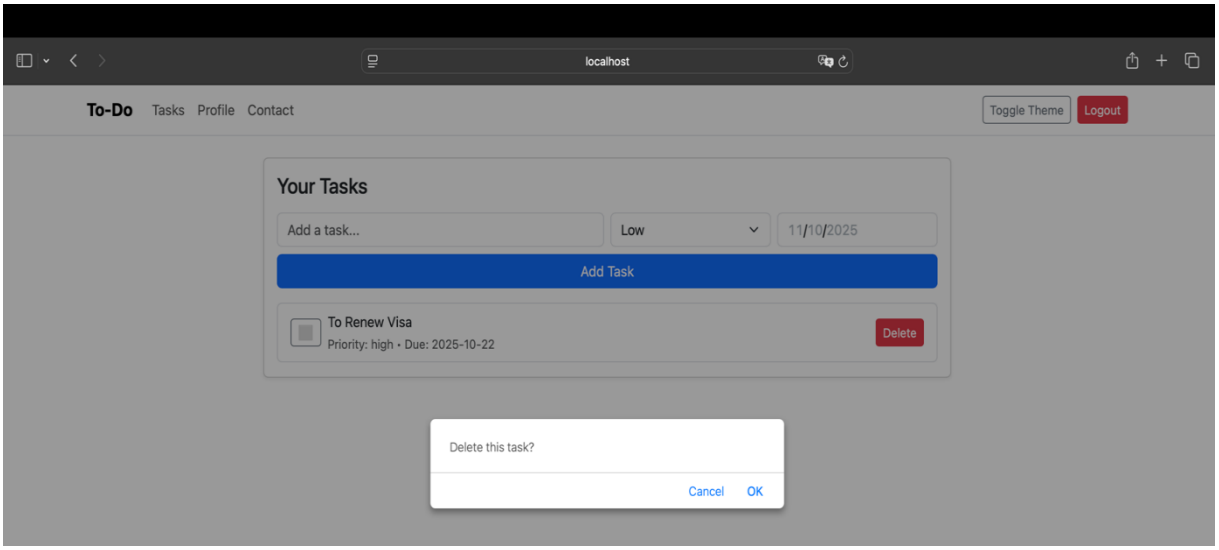
Add Task Button

When a user types a task and clicks the Add button, JavaScript sends that information to task_add.php using Fetch API. The PHP script checks who is logged in, saves the new task into the MySQL database, and sends back a message to confirm. The new task appears instantly on the list without refreshing the page.



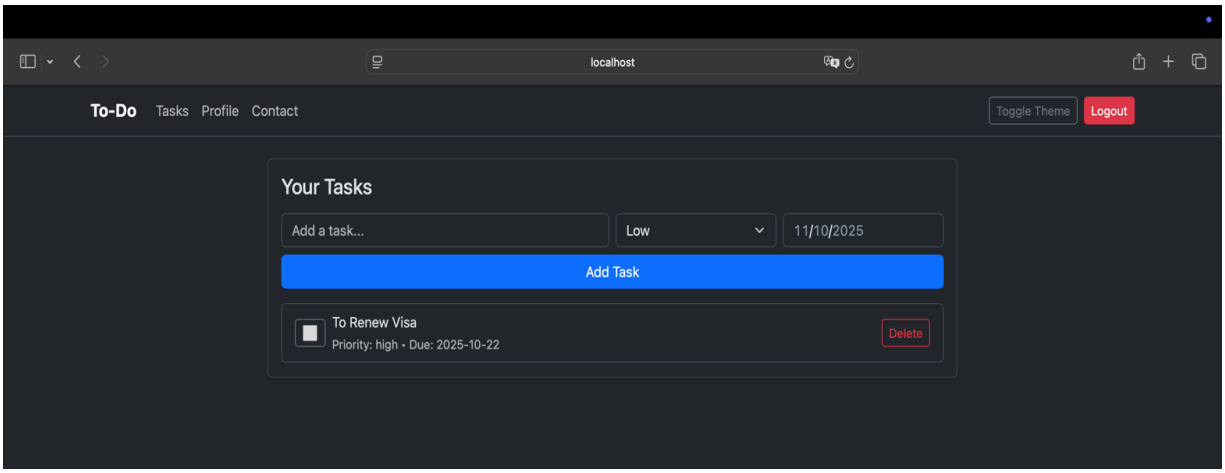
Delete Task Button

The Delete button removes a task from both the page and the database. When a user clicks the delete icon next to a task, JavaScript sends the task ID to the PHP script (task_delete.php). The server deletes that record from the tasks table using SQL DELETE. Once it is removed from the database, the page updates automatically using JavaScript to hide that task from the list.



Dark/Light Theme Switch

The theme switch lets users change the look of the website. JavaScript listens for theme changes and updates the page colors. The chosen theme is saved in a cookie for 30 days, so when the user opens the website again, it shows the same theme automatically.



3. Technologies and Frameworks Used

This project uses several web technologies that work together to build a dynamic, interactive, and secure website. Each technology plays a special role in making the To-Do system efficient and user-friendly.

Frontend:

The frontend was built using HTML5, CSS3, Bootstrap 5, and JavaScript.
HTML5 provides the main structure of the website. Every page such as login, register, and task list is created using proper HTML tags to make it readable and responsive.
CSS3 is used to make the interface look clean and simple. Colors, fonts, and layout adjustments were added to match modern web design standards.
Bootstrap 5 makes the website responsive on different devices. Its built-in grid system automatically adjusts the layout for desktop, tablet, and mobile screens.
JavaScript adds interactivity. For example, when a user clicks the Add Task button, the task appears immediately without refreshing the page. JavaScript also controls theme switching between light and dark modes.

Backend:

The backend uses PHP 8, which processes data from forms and connects to the database using PDO (PHP Data Objects).
PHP handles user registration, login, and session management.
It validates inputs and ensures security by using prepared statements to prevent SQL injection.
It also communicates directly with the database to store and retrieve tasks and messages.
Database:

The system uses MySQL as the main database. It stores all user information, tasks, and contact form messages.
The database ensures that data is consistent and easily accessible for the backend PHP scripts.

Sessions and Cookies:

Sessions are used to track logged-in users, storing their ID and name temporarily while they browse the website.
Cookies are used for saving user preferences, such as their chosen theme (light or dark) and email (if "Remember Me" is selected).
Together, these help create a smoother and more personal experience for each user.

4. Database Schema Design

The database design is a very important part of this project because it organizes how all the data is stored and linked together. It has three main tables: users, tasks, and messages.
Main Tables:
Users Table keeps details about each registered user, including name, email, and encrypted password.
Tasks Table stores the to-do list items for each user. Every task has a title, due date, and status (complete or incomplete).
Messages Table records the messages that users or guests send through the contact form.
Relationships:
A user can have many tasks, but each task belongs to only one user (One-to-Many).
A user can send many messages, but messages from guests are stored with a NULL user_id.
Constraints and Normalization:
The email field in the users table is unique to avoid duplicate accounts.
The database follows Third Normal Form (3NF) to remove duplication.
When a user is deleted, their tasks are automatically removed (ON DELETE CASCADE), while their messages become detached (ON DELETE SET NULL).
This design keeps the database clean, efficient, and reliable.

5. System Structure and Dynamic Functionality

The To-Do website is made up of multiple pages and backend scripts that work together smoothly.
Main Pages:
index.php – the main dashboard that shows all tasks and allows adding or deleting them.
login.php – checks user credentials and starts a session when login is successful.
register.php – creates a new user account after verifying that the email isn't already used.
profile.php – displays information about the logged-in user.
contact.php – allows users and guests to send messages to the system admin.
Backend Endpoints:
There are also small backend scripts inside an api folder that respond to AJAX requests.

task_add.php adds new tasks.
task_delete.php removes tasks.
task_toggle.php marks tasks as completed.
contact_submit.php saves contact form messages.
login.php, register.php, and logout.php handle user authentication.
Dynamic Features:
The system uses JavaScript Fetch API to send and receive data without reloading the page.
For example, when a user adds or deletes a task, the change appears instantly. This makes the app feel fast and modern.

6. Cookies and Sessions

Cookies and sessions are used together to keep the website interactive and personalized.
Sessions:
Sessions are created when a user logs in successfully. The session stores important data like the user's ID, name, and email.
This allows the user to move between pages without logging in again.
When the user logs out, the session is destroyed using session_unset() and session_destroy().
Cookies:
Cookies store less sensitive data that helps improve the user experience.
The theme cookie remembers if the user prefers dark or light mode.
The remember_email cookie keeps the user's email in the login form for 30 days.
Cookies and sessions together help make the website faster and easier to use while keeping important data secure.

7. Responsiveness and User Experience (UX)

The website is designed to look and work well on any device.
Using Bootstrap 5, the layout automatically changes based on screen size – large on desktop, medium on tablets, and stacked on phones.
Mobile-friendly design: Buttons, links, and input fields are big enough to tap easily on touch screens.
Clean layout: The grid system ensures that text and tasks are well spaced and aligned.
Instant feedback: Success or error messages appear as Bootstrap alerts after every action, so users know what's happening.
Dark mode: Users can switch between light and dark themes, which helps with visibility and comfort in different environments.
Overall, the user experience is simple, modern, and accessible.

8. Security and Optimization

Security is one of the most important parts of the project.
All sensitive actions such as login, registration, and task management are protected by server-side validation.
SQL Injection Protection: Every database query uses PDO prepared statements, which prevent malicious code from being injected.
Password Encryption: All passwords are encrypted using bcrypt with PHP's password_hash() before storing them in the database.
Session Protection: User sessions are checked on every page that requires login to prevent unauthorized access.
Data Validation: All form inputs are validated before being saved or sent.
Optimization:
AJAX calls reduce full-page reloads and make the system faster.
Caching and simple database indexing improve performance.
Optional CSRF tokens can be added later for even better security.
Together, these features keep the website safe, smooth, and stable for every user.

9. Challenges and Solutions

During development, a few issues came up:
Session Handling:
At first, sessions didn't carry between pages correctly. This was solved by including session_start() at the top of every PHP file that needed login data.
AJAX Without Frameworks:
Writing Fetch API manually was tricky at first, but using reusable functions helped simplify requests and responses.
Theme Persistence:
The theme would reset on page reload, so a cookie system was added to save the user's choice.
These challenges helped improve understanding of backend logic, JavaScript handling, and cookie management.

10. Testing and Validation

Testing is important to make sure all features work correctly.
Each part of the website was tested several times to confirm that it works as expected.
Tests included:
Registering new users and checking that duplicate emails are blocked.
Logging in and verifying that sessions remain active.
Adding, deleting, and toggling tasks in the database.
Submitting messages through the contact form.
Testing responsiveness on desktop, tablet, and mobile using Chrome DevTools.
All tests were successful, confirming that the system works correctly and efficiently.

11. Conclusion

The To-Do Web Application successfully combines design, functionality, and security.
It demonstrates how frontend and backend systems can work together to create a smooth, interactive website.
The use of PHP, MySQL, Bootstrap, and JavaScript made it possible to build a real, working system that looks good and performs well.
Through this project, I learned how to handle sessions, store data securely, and make responsive web pages.
It's a complete project that meets the requirements and provides a strong foundation for future improvements such as progress tracking, reminders, or sharing tasks between users.