Project Title:

TO-DO LIST

Submitted By:

- Savio Nazareth | 2462362 | savio.john@btech.christuniversity.in
- Tenzin Rigzin | 2460462 | tenzin.rigzin@btech.christuniversity.in
- Shamith Gowda | 2462530 | shamith.m@btech.christuniversity.in

Course: Web Design Fundamentals
Instructor Name: Narendra Kumar
Institution: Christ University
Date of Submission: 24/09/2025

2. Abstract

The To-Do List application is a simple, interactive, and responsive web app that helps users manage their daily tasks efficiently. It allows users to add, edit, delete, filter, and mark tasks as completed. Tasks are saved in the browser's localStorage, so they persist even after refreshing the page. The project demonstrates the practical use of front-end technologies such as HTML, CSS, JavaScript, Bootstrap, and jQuery in building a real-world productivity tool.

3. Objectives

- To design an intuitive and user-friendly interface for managing daily tasks.
- To implement CRUD (Create, Read, Update, Delete) functionality for tasks.
- To ensure tasks are stored persistently using localStorage.
- To use Bootstrap and jQuery for faster development and improved user experience.
- To make the app responsive across devices.

4. Scope of the Project

The app focuses on **personal task management**.

Scope is limited to **client-side** (no server/database).

Users can:

- Add new tasks.
- Edit or delete tasks.
- Mark tasks as completed or active.
- Filter tasks (All, Active, Completed).
- Clear all completed tasks.

Persistence is provided via **localStorage**, so the data is available until cleared by the user/browser.

5. Tools & Technologies Used

- **HTML5** – Structure of the application.
- **CSS3** – Styling, responsiveness, and UI presentation.
- **JavaScript (ES6)** – Logic for task management and localStorage.
- **jQuery (v3.7.1)** – Simplified DOM manipulation and event handling.
- **Bootstrap 5.3.2** – Responsive grid system, buttons, and UI components.
- **Browser LocalStorage** – To store tasks persistently.

6.HTML Structure Overview

- – Main container with Bootstrap grid.
- Task Input Section – Input box + Add button.
- Filter Buttons – Buttons for All, Active, Completed.
- Task List (<ul id="taskList">) – Dynamic list where tasks are displayed.
- Empty Message Div – Shown when no tasks exist.
- Footer Summary – Shows items left + clear completed option.
- Scripts – jQuery, Bootstrap bundle, and custom JS logic at the bottom.

7. CSS Styling Strategy

- Light background theme (#f8f9fa) for a clean UI.
- .task-item → Flexbox layout for task rows (checkbox, text, buttons).
- .task-text.completed → Line-through style for completed tasks.
- Hover effects to improve interactivity.
- Responsive behavior using media queries for small screens.

8. JavaScript Implementation

Maintains a **tasks array** with objects { id, text, completed }.

Implements CRUD operations:

- addTask() → Adds new task.
- deleteTask(id) → Deletes task by ID.
- toggleComplete(id, completed) → Marks as done/undone.
- saveEdit() → Updates edited task.

Uses **localStorage** to persist data (saveTasks() and loadTasks()).

renderTasks() dynamically updates the DOM with the latest tasks.

Handles keyboard shortcuts: Enter (save task), Esc (cancel edit).

9. Bootstrap & jQuery Implementation

- **Bootstrap**:
    - Grid system (row, col-md-8, col-md-4) for layout.
    - Components like **buttons, input groups, forms, cards**.
    - Utility classes (mt-3, text-muted, shadow-sm) for quick styling.
- **jQuery**:
    - Simplifies event handling ($('#addTaskBtn').on('click', ...)).
    - Used for DOM manipulation ($('#taskList').empty().append(...)).
    - Provides delegated events for dynamically created elements (edit/delete buttons).
    - Manages filter button states with .addClass('active') / .removeClass('active').

10. Key Features

- Add, edit, and delete tasks.
- Mark tasks as completed with checkboxes.
- Filter tasks: All / Active / Completed.
- LocalStorage persistence (data stays after refresh).
- Responsive design (mobile-friendly).
- “Clear Completed” button for quick cleanup.
- Inline editing of tasks with save/cancel buttons.

11. Challenges Faced & Solutions

1. Persisting tasks after refresh
    - *Challenge*: Without a backend, tasks vanished on reload.
    - *Solution*: Implemented localStorage to save and load tasks.
2. Editing inline without breaking layout
    - *Challenge*: Switching between text and input fields was tricky.
    - *Solution*: Used conditional display (.hide() / .show()) and dynamic event binding.
3. Filtering tasks efficiently
    - *Challenge*: Managing filtered vs. full task lists.
    - *Solution*: Applied filter() on tasks array before rendering.

12. Outcome

- A fully functional, responsive, and user-friendly To-Do List web app.
- Demonstrates **frontend development skills** using modern libraries.
- Provides a foundation for building more advanced task managers.

13. Future Enhancements

- Add due dates and reminders for tasks.
- Sync with a backend database for multi-device access.
- Enable drag-and-drop reordering of tasks.
- Add dark mode for better accessibility.
- Implement categories/tags for task grouping.

14. Sample Code

Program 1 - Todo List

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>To-Do List App</title>

  <!-- Bootstrap CSS (v5) -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">

  <style>
    /* Small custom styles */
    body { background: #f8f9fa; }
    .task-item { display:flex; align-items:center; gap:.75rem; padding:.6rem .75rem; border-radius:.5rem; }
    .task-item:hover { background: rgba(0,0,0,0.02); }
    .task-text { flex:1; word-break:break-word; }
    .task-text.completed { text-decoration: line-through; color: #6c757d; }
    .btn-icon { padding: .2rem .5rem; }
    .no-tasks { color: #6c757d; text-align:center; padding:1.2rem 0; }
    @media (max-width:576px) {
      .controls-row .col-auto { flex: 1 1 100%; }
      .controls-row .input-group { width:100%; }
    }
  </style>
</head>
<body>
```

```
<div class="container py-4">
  <div class="row justify-content-center">
    <div class="col-12 col-md-9 col-lg-7">
      <div class="card shadow-sm">
        <div class="card-body">
          <h3 class="card-title mb-3 text-center">To-Do List</h3>

          <!-- Add task -->
          <div class="row g-2 align-items-center controls-row mb-3">
            <div class="col-12 col-md-8">
              <div class="input-group">
                <input id="taskInput" type="text" class="form-control" placeholder="Add a new task..." aria-label="New task">
                <button id="addTaskBtn" class="btn btn-primary" type="button">Add</button>
              </div>
            </div>

            <div class="col-12 col-md-4 text-md-end mt-2 mt-md-0">
              <div class="btn-group" role="group" aria-label="filter">
                <button class="btn btn-outline-secondary active filter-btn" data-filter="all">All</button>
                <button class="btn btn-outline-secondary filter-btn" data-filter="active">Active</button>
                <button class="btn btn-outline-secondary filter-btn" data-filter="completed">Completed</button>
              </div>
            </div>
          </div>
```

```
        <!-- Task list -->
        <ul id="taskList" class="list-unstyled mb-0"></ul>

        <div id="emptyMsg" class="no-tasks mt-3" style="display:none;">
          No tasks yet — add something to get started!
        </div>

        <!-- Footer summary -->
        <div class="d-flex justify-content-between align-items-center mt-3 small text-muted">
          <div id="itemsLeft">0 items left</div>
          <div>
            <button id="clearCompleted" class="btn btn-sm btn-outline-danger">Clear completed</button>
          </div>
        </div>

      </div>
    </div>

    <div class="text-center mt-3 small text-muted">
      Built with HTML, CSS, JavaScript, Bootstrap &amp;amp; jQuery
    </div>
  </div>
</div>

<!-- jQuery -->
<script src="https://code.jquery.com/jquery-3.7.1.min.js"></script>
```

```
  <!-- Bootstrap JS bundle -->
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></script>

  <script>
    // Data model in localStorage
    const STORAGE_KEY = 'todo_tasks_v1';

    // state
    let tasks = []; // { id, text, completed }
    let currentFilter = 'all'; // all | active | completed

    // helpers
    function saveTasks() {
      localStorage.setItem(STORAGE_KEY, JSON.stringify(tasks));
    }

    function loadTasks() {
      const raw = localStorage.getItem(STORAGE_KEY);
      tasks = raw ? JSON.parse(raw) : [];
    }

    function renderTasks() {
      const $list = $('#taskList').empty();
      const filtered = tasks.filter(t => {
        if (currentFilter === 'active') return !t.completed;
        if (currentFilter === 'completed') return t.completed;
        return true;
```

```
  });

  if (filtered.length === 0) {
    $('#emptyMsg').show();
  } else {
    $('#emptyMsg').hide();
  }

  filtered.forEach(task => {
    const $li = $(`
      <li class="task-item mb-2" data-id="${task.id}">
        <div class="form-check">
          <input class="form-check-input task-checkbox" type="checkbox" ${task.completed ? 'checked' : ''} id="chk-${task.id}">
        </div>
        <div class="task-text ${task.completed ? 'completed' : ''}" data-role="text">${escapeHtml(task.text)}</div>
        <div class="input-edit" style="display:none; flex:1;">
          <div class="input-group">
            <input type="text" class="form-control form-control-sm edit-input" value="${escapeAttr(task.text)}">
            <button class="btn btn-sm btn-success save-edit" title="Save">Save</button>
            <button class="btn btn-sm btn-outline-secondary cancel-edit" title="Cancel">Cancel</button>
          </div>
        </div>
        <div class="btn-group ms-2" role="group">
          <button class="btn btn-sm btn-outline-secondary btn-icon edit-btn" title="Edit">
```

```
            <svg xmlns="http://www.w3.org/2000/svg" width="14" height="14"
fill="currentColor" viewBox="0 0 16 16"><path d="M15.502 1.94a1.5 1.5 0 0 0-2.121
0L5.5 9.821V12h2.179l7.884-7.884a1.5 1.5 0 0 0 0-2.176z"/></svg>

          </button>

          <button class="btn btn-sm btn-outline-danger btn-icon delete-btn"
title="Delete">

            <svg xmlns="http://www.w3.org/2000/svg" width="14" height="14"
fill="currentColor" viewBox="0 0 16 16"><path d="M5.5 5.5v7a.5.5 0 0 0 1 0v-7a.5.5
0 0 0-1 0zM8 5.5v7a.5.5 0 0 0 1 0v-7a.5.5 0 0 0-1 0zM10.5 5.5v7a.5.5 0 0 0 1
0v-7a.5.5 0 0 0-1 0z"/><path fill-rule="evenodd" d="M14.5 3a1 1 0 0 1-1 1H13v9a2 2
0 0 1-2 2H5a2 2 0 0 1-2-2V4h-.5a1 1 0 1 1 0-2h3.592a1 1 0 0 1 .95.69L8
3h2l.457-1.31A1 1 0 0 1 11.407 1H15a1 1 0 0 1 0 2h-.5z"/></svg>

          </button>

        </div>

      </li>

    `);

    $list.append($li);

  });


  updateFooter();

}


function updateFooter() {

  const left = tasks.filter(t => !t.completed).length;

  $('#itemsLeft').text(${left} item${left !== 1 ? 's' : ''} left);

}


function addTask(text) {

  const trimmed = text.trim();

  if (!trimmed) return;

  const task = {
```

```
    id: Date.now().toString(),
    text: trimmed,
    completed: false
  };
  tasks.unshift(task); // newest first
  saveTasks();
  renderTasks();
}

function deleteTask(id) {
  tasks = tasks.filter(t => t.id !== id);
  saveTasks();
  renderTasks();
}

function toggleComplete(id, completed) {
  const t = tasks.find(x => x.id === id);
  if (t) t.completed = completed;
  saveTasks();
  renderTasks();
}

function startEdit($li) {
  $li.find('[data-role="text"]').hide();
  $li.find('.input-edit').show();
  $li.find('.edit-input').focus().select();
}
```

```
function cancelEdit($li) {
  $li.find('.input-edit').hide();
  $li.find('[data-role="text"]').show();
}

function saveEdit($li) {
  const id = $li.data('id').toString();
  const newText = $li.find('.edit-input').val().trim();
  if (newText === '') {
    // if empty, treat as delete
    deleteTask(id);
    return;
  }
  const t = tasks.find(x => x.id === id);
  if (t) t.text = newText;
  saveTasks();
  renderTasks();
}

function setFilter(filter) {
  currentFilter = filter;
  $('.filter-btn').removeClass('active');
  $(.filter-btn[data-filter="${filter}"]).addClass('active');
  renderTasks();
}

function clearCompleted() {
  tasks = tasks.filter(t => !t.completed);
```

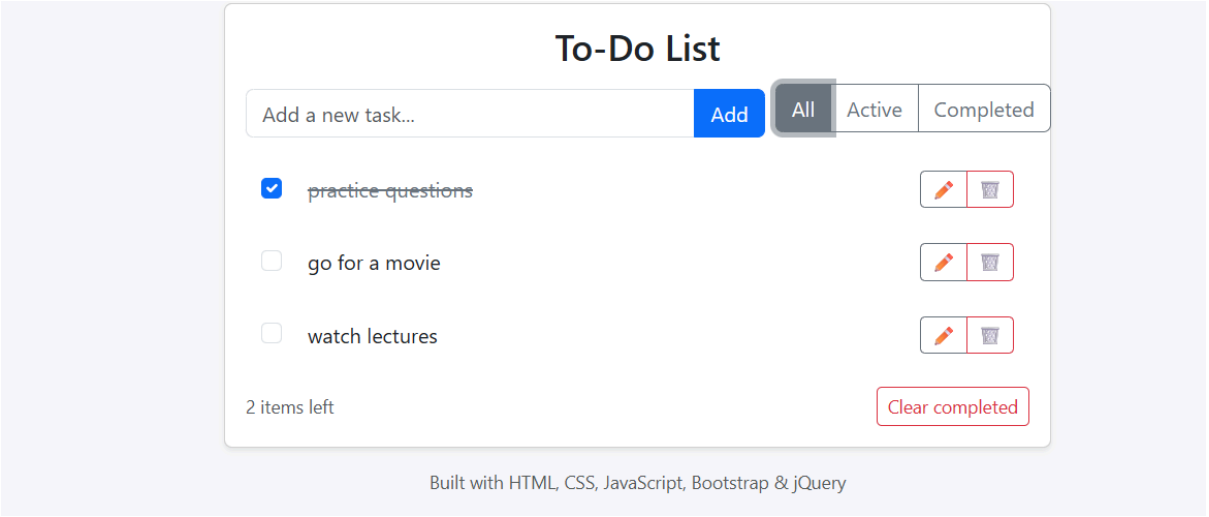
```
  saveTasks();
  renderTasks();
}

// Simple escaping helpers to avoid accidental HTML injection in text nodes.
function escapeHtml(str) {
  return String(str)
    .replaceAll('&', '&amp;')
    .replaceAll('<', '&lt;')
    .replaceAll('>', '&gt;')
    .replaceAll('"', '&quot;')
    .replaceAll("'", '&#39;');
}
function escapeAttr(str) {
  return String(str).replaceAll('"', '&quot;').replaceAll("'", '&#39;');
}

// --- Event wiring ---
$(function(){
  loadTasks();
  renderTasks();

  // Add task (button)
  $('#addTaskBtn').on('click', function(){
    const text = $('#taskInput').val();
    addTask(text);
    $('#taskInput').val('').focus();
  });
```

```
// Add task (enter)
$('#taskInput').on('keypress', function(e){
  if (e.which === 13) {
    $('#addTaskBtn').click();
  }
});

// Delegated event: checkbox toggle
$('#taskList').on('change', '.task-checkbox', function(){
  const id = $(this).closest('li').data('id').toString();
  const checked = $(this).is(':checked');
  toggleComplete(id, checked);
});

// Delegated: delete
$('#taskList').on('click', '.delete-btn', function(){
  const id = $(this).closest('li').data('id').toString();
  if (confirm('Delete this task?')) {
    deleteTask(id);
  }
});

// Delegated: edit button
$('#taskList').on('click', '.edit-btn', function(){
  const $li = $(this).closest('li');
  startEdit($li);
});
```

```
// Delegated: cancel edit
$('#taskList').on('click', '.cancel-edit', function(){
  const $li = $(this).closest('li');
  cancelEdit($li);
});

// Delegated: save edit
$('#taskList').on('click', '.save-edit', function(){
  const $li = $(this).closest('li');
  saveEdit($li);
});

// Delegated: pressing Enter in edit input saves
$('#taskList').on('keypress', '.edit-input', function(e){
  if (e.which === 13) {
    $(this).closest('li').find('.save-edit').click();
  } else if (e.which === 27) { // Esc cancels
    $(this).closest('li').find('.cancel-edit').click();
  }
});

// Filter buttons
$('.filter-btn').on('click', function(){
  const filter = $(this).data('filter');
  setFilter(filter);
});
```

```
    // Clear completed
    $('#clearCompleted').on('click', function(){
      if (confirm('Remove all completed tasks?')) clearCompleted();
    });

    // Allow clicking on text to toggle completed (optional)
    $('#taskList').on('click', '.task-text', function(){
      const $li = $(this).closest('li');
      const id = $li.data('id').toString();
      const t = tasks.find(x => x.id === id);
      if (t) toggleComplete(id, !t.completed);
    });

  });
  </script>
</body>
</html>
```

15. Screenshots of Final Output



16. Conclusion

The To-Do List project successfully showcases the integration of HTML, CSS, JavaScript, Bootstrap, and jQuery to build a practical web application. It provides all the core features of a task manager while ensuring persistence with localStorage. The project demonstrates how clean UI/UX combined with simple yet effective logic can solve real-world productivity needs.