```
    // Search functionality
    this.searchInput.addEventListener('input', () => this.render());

    // Filter functionality
    this.filterBtns.forEach(btn => {
        btn.addEventListener('click', (e) => {
            this.setFilter(e.target.dataset.filter);
        });
    });

    // Bulk actions
    this.clearCompletedBtn.addEventListener('click', () => this.clearCompleted());
    this.markAllCompleteBtn.addEventListener('click', () => this.markAllComplete());

    // Modal events
    this.closeModal.addEventListener('click', () => this.closeEditModal());
    this.cancelEdit.addEventListener('click', () => this.closeEditModal());
    this.saveEdit.addEventListener('click', () => this.saveEditTodo());
    this.editInput.addEventListener('keypress', (e) => {
        if (e.key === 'Enter') this.saveEditTodo();
    });

    // Close modal when clicking outside
    this.editModal.addEventListener('click', (e) => {
        if (e.target === this.editModal) this.closeEditModal();
    });
}

// CREATE: Add new todo
addTodo() {
    const text = this.todoInput.value.trim();
    if (!text) {
        this.showNotification('Please enter a task!', 'warning');
        return;
    }

    const todo = {
        id: Date.now().toString(),
        text: text,
        completed: false,
        createdAt: new Date().toISOString()
    };

    this.todos.unshift(todo);
```

```
    this.todoInput.value = '';
    this.saveTodos();
    this.render();
    this.showNotification('Task added successfully!', 'success');
}

// READ: Get todos based on current filter and search
getFilteredTodos() {
    let filtered = [...this.todos];

    // Apply search filter
    const searchTerm = this.searchInput.value.toLowerCase().trim();
    if (searchTerm) {
        filtered = filtered.filter(todo =>
            todo.text.toLowerCase().includes(searchTerm)
        );
    }

    // Apply status filter
    switch (this.currentFilter) {
        case 'active':
            filtered = filtered.filter(todo => !todo.completed);
            break;
        case 'completed':
            filtered = filtered.filter(todo => todo.completed);
            break;
        default:
            // 'all' - no additional filtering
            break;
    }

    return filtered;
}

// READ: Render todos to DOM
render() {
    const filteredTodos = this.getFilteredTodos();

    // Clear current list
    this.todoList.innerHTML = '';

    if (filteredTodos.length === 0) {
        this.emptyState.style.display = 'block';
        this.todoList.style.display = 'none';
```

```
    } else {
        this.emptyState.style.display = 'none';
        this.todoList.style.display = 'block';

        filteredTodos.forEach(todo => {
            const todoElement = this.createTodoElement(todo);
            this.todoList.appendChild(todoElement);
        });
    }

    this.updateStatistics();
    this.updateFilterButtons();
}

// Create individual todo element
createTodoElement(todo) {
    const li = document.createElement('li');
    li.className = `todo-item ${todo.completed ? 'completed' : ''}`;
    li.dataset.id = todo.id;

    li.innerHTML = `
        <input type="checkbox" class="todo-checkbox" ${todo.completed ? 'checked' : ''}>
        <span class="todo-text">${this.escapeHtml(todo.text)}</span>
        <div class="todo-actions">
            <button class="action-btn edit-btn" onclick="todoApp.editTodo('${todo.id}')">
                <i class="fas fa-edit"></i> Edit
            </button>
            <button class="action-btn delete-btn" onclick="todoApp.deleteTodo('${todo.id}')">
                <i class="fas fa-trash"></i> Delete
            </button>
        </div>
    `;

    // Add event listener for checkbox
    const checkbox = li.querySelector('.todo-checkbox');
    checkbox.addEventListener('change', () => this.toggleTodo(todo.id));

    return li;
}

// UPDATE: Toggle todo completion status
toggleTodo(id) {
    const todo = this.todos.find(t => t.id === id);
    if (todo) {
```

```
        todo.completed = !todo.completed;
        todo.updatedAt = new Date().toISOString();
        this.saveTodos();
        this.render();

        const status = todo.completed ? 'completed' : 'marked as active';
        this.showNotification(`Task ${status}!`, 'info');
    }
}

// UPDATE: Edit todo (open modal)
editTodo(id) {
    const todo = this.todos.find(t => t.id === id);
    if (todo) {
        this.editingId = id;
        this.editInput.value = todo.text;
        this.editModal.style.display = 'block';
        this.editInput.focus();
        this.editInput.select();
    }
}

// UPDATE: Save edited todo
saveEditTodo() {
    const text = this.editInput.value.trim();
    if (!text) {
        this.showNotification('Please enter a task!', 'warning');
        return;
    }

    const todo = this.todos.find(t => t.id === this.editingId);
    if (todo) {
        todo.text = text;
        todo.updatedAt = new Date().toISOString();
        this.saveTodos();
        this.render();
        this.closeEditModal();
        this.showNotification('Task updated successfully!', 'success');
    }
}

// DELETE: Remove todo
deleteTodo(id) {
    if (confirm('Are you sure you want to delete this task?')) {
```

```
        this.todos = this.todos.filter(t => t.id !== id);
        this.saveTodos();
        this.render();
        this.showNotification('Task deleted successfully!', 'info');
    }
}

// DELETE: Clear all completed todos
clearCompleted() {
    const completedCount = this.todos.filter(t => t.completed).length;
    if (completedCount === 0) {
        this.showNotification('No completed tasks to clear!', 'info');
        return;
    }

    if (confirm(`Are you sure you want to delete ${completedCount} completed task(s)?`)) {
        this.todos = this.todos.filter(t => !t.completed);
        this.saveTodos();
        this.render();
        this.showNotification(`${completedCount} completed task(s) cleared!`, 'info');
    }
}

// UPDATE: Mark all todos as complete
markAllComplete() {
    const activeTodos = this.todos.filter(t => !t.completed);
    if (activeTodos.length === 0) {
        this.showNotification('No active tasks to complete!', 'info');
        return;
    }

    activeTodos.forEach(todo => {
        todo.completed = true;
        todo.updatedAt = new Date().toISOString();
    });

    this.saveTodos();
    this.render();
    this.showNotification(`${activeTodos.length} task(s) marked as complete!`, 'success');
}

// Set current filter
setFilter(filter) {
    this.currentFilter = filter;
```

```
    this.render();
}

// Update filter buttons UI
updateFilterButtons() {
    this.filterBtns.forEach(btn => {
        btn.classList.toggle('active', btn.dataset.filter === this.currentFilter);
    });
}

// Update statistics
updateStatistics() {
    const total = this.todos.length;
    const active = this.todos.filter(t => !t.completed).length;
    const completed = this.todos.filter(t => t.completed).length;

    this.totalTasksEl.textContent = total;
    this.activeTasksEl.textContent = active;
    this.completedTasksEl.textContent = completed;
}

// Close edit modal
closeEditModal() {
    this.editModal.style.display = 'none';
    this.editingId = null;
    this.editInput.value = '';
}

// Save todos to localStorage
saveTodos() {
    localStorage.setItem('todos', JSON.stringify(this.todos));
}

// Load todos from localStorage
loadTodos() {
    const stored = localStorage.getItem('todos');
    return stored ? JSON.parse(stored) : [];
}

// Escape HTML to prevent XSS
escapeHtml(text) {
    const div = document.createElement('div');
    div.textContent = text;
    return div.innerHTML;
```

```
}

// Show notification
showNotification(message, type = 'info') {
    // Create notification element
    const notification = document.createElement('div');
    notification.className = `notification notification-${type}`;
    notification.innerHTML = `
        <i class="fas fa-${this.getNotificationIcon(type)}"></i>
        <span>${message}</span>
    `;

    // Add styles
    notification.style.cssText = `
        position: fixed;
        top: 20px;
        right: 20px;
        background: ${this.getNotificationColor(type)};
        color: white;
        padding: 15px 20px;
        border-radius: 8px;
        box-shadow: 0 4px 12px rgba(0,0,0,0.15);
        z-index: 10000;
        display: flex;
        align-items: center;
        gap: 10px;
        font-weight: 500;
        animation: slideInRight 0.3s ease;
    `;

    // Add animation keyframes
    if (!document.querySelector('#notification-styles')) {
        const style = document.createElement('style');
        style.id = 'notification-styles';
        style.textContent = `
            @keyframes slideInRight {
                from { transform: translateX(100%); opacity: 0; }
                to { transform: translateX(0); opacity: 1; }
            }
            @keyframes slideOutRight {
                from { transform: translateX(0); opacity: 1; }
                to { transform: translateX(100%); opacity: 0; }
            }
        `;
```

```
            document.head.appendChild(style);
        }

        document.body.appendChild(notification);

        // Remove notification after 3 seconds
        setTimeout(() => {
            notification.style.animation = 'slideOutRight 0.3s ease';
            setTimeout(() => {
                if (notification.parentNode) {
                    notification.parentNode.removeChild(notification);
                }
            }, 300);
        }, 3000);
    }

    getNotificationIcon(type) {
        const icons = {
            success: 'check-circle',
            warning: 'exclamation-triangle',
            error: 'times-circle',
            info: 'info-circle'
        };
        return icons[type] || 'info-circle';
    }

    getNotificationColor(type) {
        const colors = {
            success: '#28a745',
            warning: '#ffc107',
            error: '#dc3545',
            info: '#17a2b8'
        };
        return colors[type] || '#17a2b8';
    }
}

// Initialize the application
let todoApp;
document.addEventListener('DOMContentLoaded', () => {
    todoApp = new TodoApp();
});

// Add some sample data if no todos exist
```

```
if (localStorage.getItem('todos') === null) {
    const sampleTodos = [
        {
            id: '1',
            text: 'Welcome to your To-Do List!',
            completed: false,
            createdAt: new Date().toISOString()
        },
        {
            id: '2',
            text: 'Click the checkbox to mark tasks as complete',
            completed: false,
            createdAt: new Date().toISOString()
        },
        {
            id: '3',
            text: 'Use the Edit button to modify tasks',
            completed: false,
            createdAt: new Date().toISOString()
        },
        {
            id: '4',
            text: 'Try the search and filter features',
            completed: true,
            createdAt: new Date().toISOString()
        }
    ];
    localStorage.setItem('todos', JSON.stringify(sampleTodos));
}
```