```
// Form validation and data management
class UserManagement {
    constructor() {
        this.users = [];
        this.form = document.getElementById('userForm');
        this.tableBody = document.getElementById('userTableBody');

        // Initialize event listeners
        this.initializeEventListeners();

        // Load any existing users (could be from localStorage or API)
        this.loadUsers();
    }

    initializeEventListeners() {
        // Form submission
        this.form.addEventListener('submit', (e) => this.handleFormSubmit(e));

        // Real-time validation
        const inputs = this.form.querySelectorAll('input, select');
        inputs.forEach(input => {
            input.addEventListener('blur', () => this.validateField(input));
            input.addEventListener('input', () => this.validateField(input));
        });

        // Table actions
        this.tableBody.addEventListener('click', (e) =>
this.handleTableActions(e));
    }

    validateField(field) {
        const errorElement = field.nextElementSibling;
        let isValid = true;
        let errorMessage = '';

        // Reset field state
        field.classList.remove('is-invalid');
        errorElement.textContent = '';

        // Required field validation
        if (!field.value.trim()) {
            isValid = false;
            errorMessage = ${field.id.charAt(0).toUpperCase() + field.id.slice(1)}
is required;
        }

        // Specific field validations
        switch (field.id) {
            case 'email':
                if (field.value && !this.isValidEmail(field.value)) {
                    isValid = false;
                    errorMessage = 'Please enter a valid email address';
                }
                break;

            case 'password':
                if (field.value && !this.isValidPassword(field.value)) {
                    isValid = false;
                    errorMessage = 'Password must be at least 8 characters long and
```

```
contain at least one number and one letter';
                }
                break;

            case 'confirmPassword':
                const password = document.getElementById('password').value;
                if (field.value !== password) {
                    isValid = false;
                    errorMessage = 'Passwords do not match';
                }
                break;
        }

        // Update UI
        if (!isValid) {
            field.classList.add('is-invalid');
            errorElement.textContent = errorMessage;
        }

        return isValid;
    }

    isValidEmail(email) {
        return /^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(email);
    }

    isValidPassword(password) {
        return password.length >= 8 && /[A-Za-z]/.test(password) && /[0-
9]/.test(password);
    }

    handleFormSubmit(e) {
        e.preventDefault();

        // Validate all fields
        const fields = this.form.querySelectorAll('input, select');
        let isValid = true;

        fields.forEach(field => {
            if (!this.validateField(field)) {
                isValid = false;
            }
        });

        if (isValid) {
            this.addUser();
        }
    }

    addUser() {
        const newUser = {
            id: Date.now(), // Simple unique ID
            firstName: document.getElementById('firstName').value,
            lastName: document.getElementById('lastName').value,
            email: document.getElementById('email').value,
            role: document.getElementById('role').value,
            status: 'Active'
        };
```

```
        this.users.push(newUser);
        this.saveUsers();
        this.renderUsers();
        this.form.reset();

        // Show success message
        this.showNotification('User added successfully!', 'success');
    }

    editUser(userId) {
        const user = this.users.find(u => u.id === userId);
        if (user) {
            // Populate form with user data
            document.getElementById('firstName').value = user.firstName;
            document.getElementById('lastName').value = user.lastName;
            document.getElementById('email').value = user.email;
            document.getElementById('role').value = user.role;

            // Change form submit button
            const submitBtn = this.form.querySelector('button[type="submit"]');
            submitBtn.textContent = 'Update User';
            this.form.dataset.editingId = userId;
        }
    }

    deleteUser(userId) {
        if (confirm('Are you sure you want to delete this user?')) {
            this.users = this.users.filter(user => user.id !== userId);
            this.saveUsers();
            this.renderUsers();
            this.showNotification('User deleted successfully!', 'success');
        }
    }

    handleTableActions(e) {
        const button = e.target.closest('button');
        if (!button) return;

        const row = button.closest('tr');
        const userId = parseInt(row.dataset.userId);

        if (button.classList.contains('btn-primary')) {
            this.editUser(userId);
        } else if (button.classList.contains('btn-danger')) {
            this.deleteUser(userId);
        }
    }

    renderUsers() {
        this.tableBody.innerHTML = this.users.map(user => `
            <tr data-user-id="${user.id}">
                <td>${user.firstName} ${user.lastName}</td>
                <td>${user.email}</td>
                <td>${user.role}</td>
                <td><span class="badge bg-success">${user.status}</span></td>
                <td>
                    <button class="btn btn-sm btn-primary me-2">Edit</button>
                    <button class="btn btn-sm btn-danger">Delete</button>
                </td>
```

```
            </tr>
        `).join('');
    }

    showNotification(message, type = 'info') {
        // Create notification element
        const notification = document.createElement('div');
        notification.className = alert alert-${type} position-fixed top-0 end-0
m-3;
        notification.style.zIndex = '1050';
        notification.textContent = message;

        document.body.appendChild(notification);

        // Remove after 3 seconds
        setTimeout(() => {
            notification.remove();
        }, 3000);
    }

    saveUsers() {
        localStorage.setItem('users', JSON.stringify(this.users));
    }

    loadUsers() {
        const savedUsers = localStorage.getItem('users');
        if (savedUsers) {
            this.users = JSON.parse(savedUsers);
            this.renderUsers();
        }
    }
}

// Initialize the user management system
document.addEventListener('DOMContentLoaded', () => {
    const userManagement = new UserManagement();
});
```