# **Technical Design Document**

Contents

1. [**Technical Design Document** 1](#_Toc200414871)
2. [Application Overview 3](#_Toc200414872)
3. [Description 3](#_Toc200414873)
4. [Functional Requirements 3](#_Toc200414874)
5. [Technology Stack 3](#_Toc200414875)
6. [Frontend Technologies 3](#_Toc200414876)
7. [Data Storage 3](#_Toc200414877)
8. [Development Tools 3](#_Toc200414878)
9. [Architecture and Design Details 4](#_Toc200414879)
10. [System Architecture 4](#_Toc200414880)
11. [Components Used 4](#_Toc200414881)
12. [Data Flow Process 4](#_Toc200414882)
13. [File Structure and Detailed Analysis 4](#_Toc200414883)
14. [i. index.html - Structure and Layout 4](#_Toc200414884)
15. [ii. styles.css - Visual Design and Layout 5](#_Toc200414885)
16. [iii.script.js - Application Logic and Functionality 6](#_Toc200414886)
17. [Testing Considerations 8](#_Toc200414887)
18. [Manual Testing Scenarios 8](#_Toc200414888)
19. [Deployment Strategy 13](#_Toc200414889)
20. [Deployment Options 13](#_Toc200414890)
21. [Conclusion 13](#_Toc200414891)

## Application Overview

### Description

The **Application Management System** is a web-based form application that allows users to add certain details (Application name, Application Type and associated owner Contact email. This system provides a simple interface for adding application details and viewing them in a structured format.

## Functional Requirements

1. **Application Registration**: Users should be able to add new applications through form.
2. **Data Validation**: System must validate application name (3-50 characters, alphanumeric with spaces) and email format.
3. **Data Persistence**: All submitted applications should be saved within the local storage of the client browser.
4. **Data Display**: Saved applications should be displayed in a structured grid format.
5. **Form Management**: Users should be able to cancel/reset the form at any time.
6. **Error Handling**: Clear error messages should be displayed for invalid inputs.

## Technology Stack

### Frontend Technologies

* **HTML5**: Structure and semantic markup
* **CSS3**: Styling, layout, and responsive design
* **JavaScript (ES6+)**: Client-side logic and DOM manipulation

### Data Storage

* **localStorage API**: Browser-based persistent storage
* **JSON**: Data serialization format

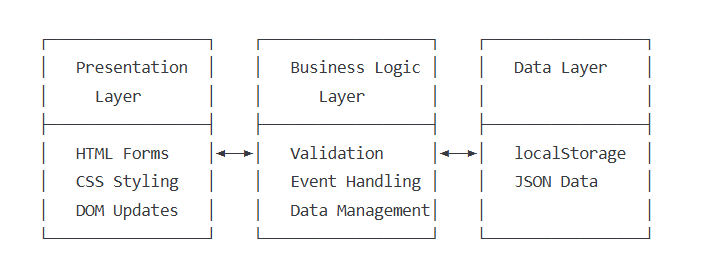
### Development Tools

* **Modern Web Browser**: Development and testing environment
* **Text Editor/IDE**: VS Code development tool.
* **Browser Developer Tools**: Debugging and performance analysis

## Architecture and Design Details

### System Architecture

The application follows a Client-Side Single Page Application (SPA) architecture:

****

### Components Used

1. **HTML (index.html)**
2. **CSS (styles.css)**
3. **JavaScript(script.js)**

### Data Flow Process

1. Initialize: Load existing data from localStorage on page load.
2. Validate: Check form inputs against defined rules.
3. Store: Add new application to array and serialize to localStorage.
4. Display: Update DOM table with current application list.
5. Persist: Data remains available across browser sessions.

## File Structure and Detailed Analysis

### index.html - Structure and Layout

The HTML file serves as the foundation, defining the document structure and user interface elements.

|  |  |  |
| --- | --- | --- |
| **Components** | **Code Snippets** | **Functionality** |
| Document Head(<head> </head>) | **<head>**  **<meta charset="UTF-8">**  **<meta name="viewport" content="width=device-width, initial-scale=1.0">**  **<title>Applications</title>**  **<link rel="stylesheet" href="styles.css">**  **</head>** | * UTF 8 encoding used for proper character display * For mobile compatibility the viewport meta tag helps * Her the external CSS file is linked |
| Application Form | **<div class="form-section">**  **<h2>Add Application</h2>**  **<form id="applicationForm">**  **<div class="form-row">**  **<div class="form-group">**  **<label for="appName">Application Name</label>**  **<input type="text" id="appName" name="appName" required>**  **<div id="appNameError" class="error">Application Name must be 3-50 characters long with no special characters</div>**  **</div>**  **<!-- Additional form groups... -->**  **</div>**  **</form>**  **</div>** | * **Application Name Input**: Text field with client-side validation * **Application Type Select**: Dropdown with predefined options (COTS/Custom) * **Contact Email Input**: Email field with format validation * **Button Group**: Save and Cancel buttons with distinct styling |
| Display Tabel data | **<table id="applicationsTable">**  **<thead>**  **<tr>**  **<th>Application Name</th>**  **<th>Application Type</th>**  **<th>Contact</th>**  **</tr>**  **</thead>**  **<tbody id="tableBody">**  **<!-- Dynamic content inserted here -->**  **</tbody>**  **</table>** | * Creates the table in order to fill the form with data |

### styles.css - Visual Design and Layout

|  |  |  |
| --- | --- | --- |
| **Components** | **Code Snippets** | **Fuctionality** |
| **font-family** | **font-family: Arial, sans-serif;** | **Used to add the font type of the HTML document** |
| **margin** | **margin: 20px;** | **Sets the space outside the element** |
| **color** | **color: #2c3e50;** | **Defines the color of the text** |
| **text-align** | **text-align: center;** | **Aligns the text within its container** |
| **text-align: right;** | **Aligns the text to the right** |
| **text-align: left;** | **Aligns the text to the left** |
| **font-size** | **font-size: 12px;** | **Sets the size of the text** |
| **display** | **display: flex;** | **Specifies the display behavior of an element** |
| **justify-content** | **justify-content: flex-end;** | **Aligns flex items along the main axis** |
| **font-weight** | **font-weight: bold;** | **Sets the weight (boldness) of the font** |
| **background-color** | **background-color: rgb(242, 228, 242);** | **Sets the background color of an element** |
| **border-radius** | **border-radius: 8px;** | **Rounds the corners of an element** |
| **margin-bottom** | **margin-bottom: 30px;** | **Sets the space outside the bottom of the element** |
| **margin-top** | **margin-top: 30px;** | **Sets the space outside the top of the element** |
| **box-size** | **box-size: bordering;** | **(Incorrect property, should be box-sizing)** |
| **border-collapse** | **border-collapse: collapse;** | **Specifies whether table borders should be collapsed** |
| **padding** | **padding: 20px;** | **Sets the space inside the element** |
| **box-shadow** | **box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);** | **Adds shadow effects around an element** |
| **flex-wrap** | **flex-wrap: wrap;** | **Specifies whether flex items should wrap or not** |
| **width** | **width: 100%;** | **Sets the width of an element** |
| **border** | **border: 1px solid #ddd;** | **Sets the border around an element** |
| **box-sizing** | **box-sizing: border-box;** | **Defines how the width and height of an element are calculated** |

### script.js - Application Logic and Functionality

The JavaScript file handles all interactive functionality, data management, and form validation.

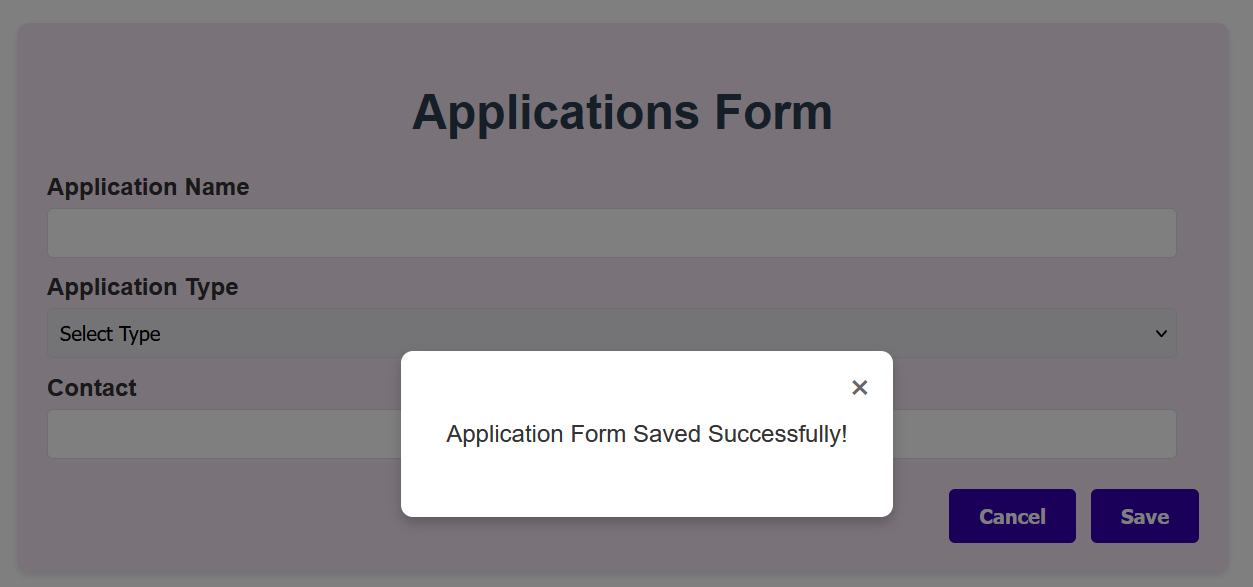
|  |  |  |
| --- | --- | --- |
| **Javascript** | **Code Snippets** | **Fuctionality** |
| **Data Initialization and Storage** | **let applications = JSON.parse(localStorage.getItem('applications')) || [];** | Attempts to load existing data from localStorage  Uses JSON parsing to convert stored string back to objects  Falls back to empty array if no data exists |
| **DOM Element References** | **const form = document.getElementById('applicationForm');**  **const appNameInput = document.getElementById('appName');**  **const appTypeInput = document.getElementById('appType');**  **const contactInput = document.getElementById('contact');**  **const appNameError = document.getElementById('appNameError');**  **const contactError = document.getElementById('contactError');** | Caches DOM element references for performance  Provides easy access to form inputs and error elements |
| **Form Validation Function** | **function validateForm() {**  **let isValid = true;**  **// Validate Application Name**  **const appName = appNameInput.value.trim();**  **const nameRegex = /^[a-zA-Z0-9 ]{3,50}$/;**  **if (!nameRegex.test(appName)) {**  **appNameError.style.display = 'block';**  **isValid = false;**  **} else {**  **appNameError.style.display = 'none';**  **}**  **// Validate Email**  **const email = contactInput.value.trim();**  **const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;**  **if (!emailRegex.test(email)) {**  **contactError.style.display = 'block';**  **isValid = false;**  **} else {**  **contactError.style.display = 'none';**  **}**  **return isValid;**  **}** | **Name Regex**: ^[a-zA-Z0-9 ]{3,50}$  ^ - Start of string  [a-zA-Z0-9 ] - Allows letters, numbers, and spaces  {3,50} - Length between 3 and 50 characters  $ - End of string  **Email Regex**: ^[^\s@]+@[^\s@]+\.[^\s@]+$  [^\s@]+ - One or more non-whitespace, non-@ characters  @ - Literal @ symbol  [^\s@]+ - Domain name part  \. - Literal dot  [^\s@]+ - Top-level domain |
| **Application Save Function** | **function saveApplication(event) {**  **event.preventDefault();**  **if (!validateForm()) {**  **return;**  **}**  **const newApplication = {**  **name: appNameInput.value.trim(),**  **type: appTypeInput.value,**  **contact: contactInput.value.trim()**  **};**  **applications.push(newApplication);**  **localStorage.setItem('applications', JSON.stringify(applications));**  **updateTable();**  **form.reset();**  **}** | **Prevent Default**: Stops normal form submission behavior  **Validation Check**: Returns early if validation fails  **Object Creation**: Creates structured application object  **Data Storage**: Adds to array and saves to localStorage  **UI Update**: Refreshes table display and resets form |
| **Table Update Function** | **function updateTable() {**  **tableBody.innerHTML = '';**  **if (applications.length === 0) {**  **tableBody.innerHTML = '<tr><td colspan="3" style="text-align: center;">No applications found</td></tr>';**  **return;**  **}**  **applications.forEach(app => {**  **const row = document.createElement('tr');**  **row.innerHTML = `**  **<td>${app.name}</td>**  **<td>${app.type}</td>**  **<td>${app.contact}</td>**  **`;**  **tableBody.appendChild(row);**  **});**  **}** | Clears existing table content to prevent duplication  Shows friendly message when no data exists  Uses template literals for clean HTML generation  Dynamically creates DOM elements for each application |
| **Event Listener Setup** | **form.addEventListener('submit', saveApplication);**  **cancelBtn.addEventListener('click', cancelForm);**  **document.addEventListener('DOMContentLoaded', updateTable);** | **Submit Event**: Handles form submission with validation  **Cancel Click**: Resets form and clears errors  **DOM Ready**: Initializes table on page load |
| **Data Persistence** | **// Saving data**  **localStorage.setItem('applications', JSON.stringify(applications));**  **// Loading data**  **let applications = JSON.parse(localStorage.getItem('applications')) || [];** | Uses browser's localStorage API for client-side persistence  JSON serialization handles object-to-string conversion  Data survives browser restarts and tab closures |
| **Form Reset and Cleanup** | **function cancelForm() {**  **form.reset();**  **appNameError.style.display = 'none';**  **contactError.style.display = 'none';**  **}** | Clears all form inputs  Hides any visible error messages  Returns form to pristine state |
| **Validation Patterns** | **// Application Name: Letters, numbers, spaces, 3-50 chars**  **const nameRegex = /^[a-zA-Z0-9 ]{3,50}$/;**  **// Email: Basic email format validation**  **const emailRegex = /^[^\s@]**[**+@[^\s@]+\.[^\s@]+$/**](mailto:+@[%5e\s@%5d+\.%5b%5e\s@%5d+$/)**;** | Provides the validation condition for application name, email. |
| **Data Flow and User Interaction** | **// Save application**  **function saveApplication(event) {**  **event.preventDefault();**    **if (!validateForm()) {**  **return;**  **}**    **const newApplication = {**  **name: appNameInput.value.trim(),**  **type: appTypeInput.value,**  **contact: contactInput.value.trim()**  **};**    **applications.push(newApplication);**  **localStorage.setItem('applications', JSON.stringify(applications));**  **updateTable();**  **form.reset();**  **}** | **Adding a New Application**  User fills out the form fields  Upon clicking "Save", saveApplication() is triggered  Form validation runs via validateForm()  If valid, new application object is created  Object is added to applications array  Array is serialized and saved to localStorage  Table is updated via updateTable()  Form is reset for next entry |

## Testing Considerations

### Manual Testing Scenarios

#### Valid Form Submission

* + Input: Valid name, type, and email
  + Expected: Application saved and displayed in table

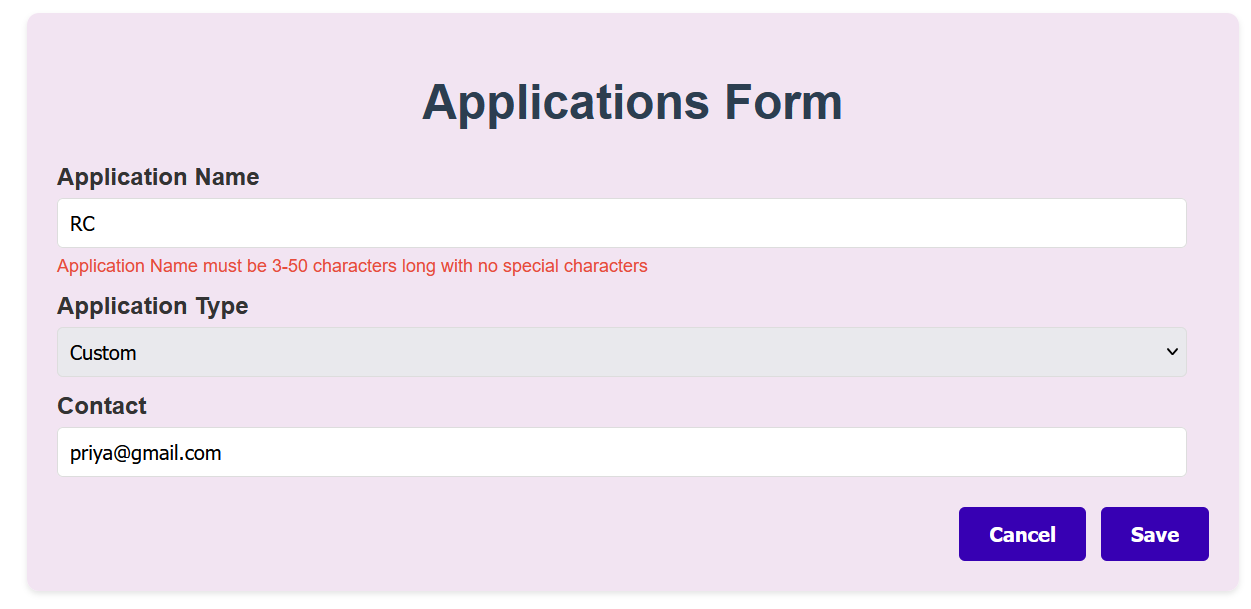


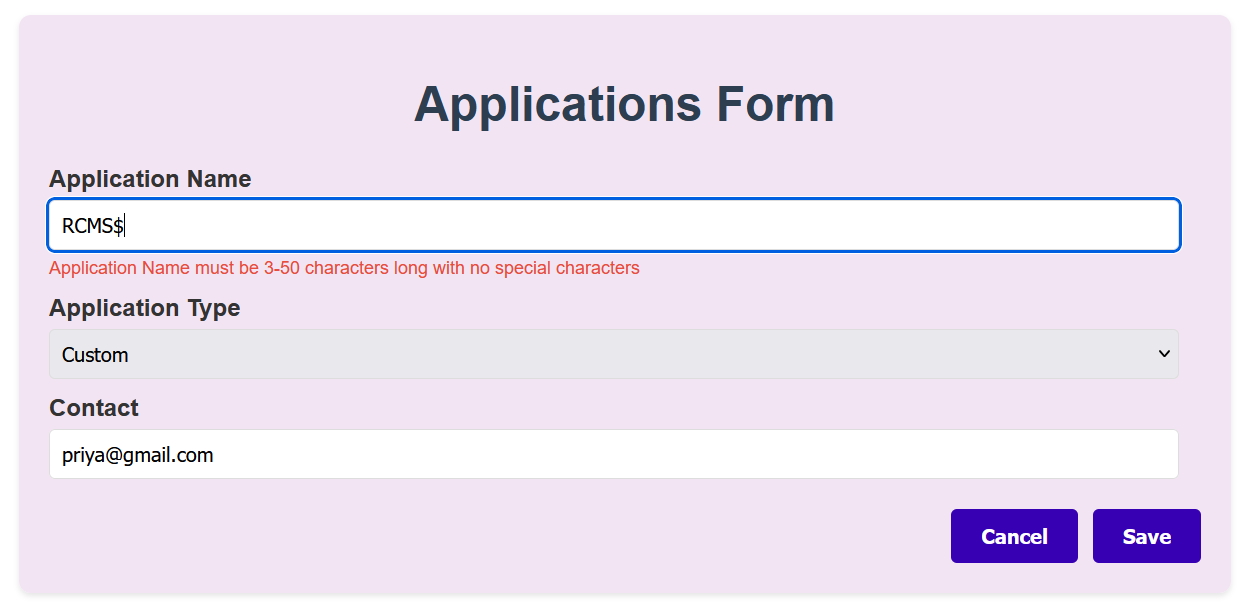
A screenshot of a computer

AI-generated content may be incorrect.

#### Name Validation Testing

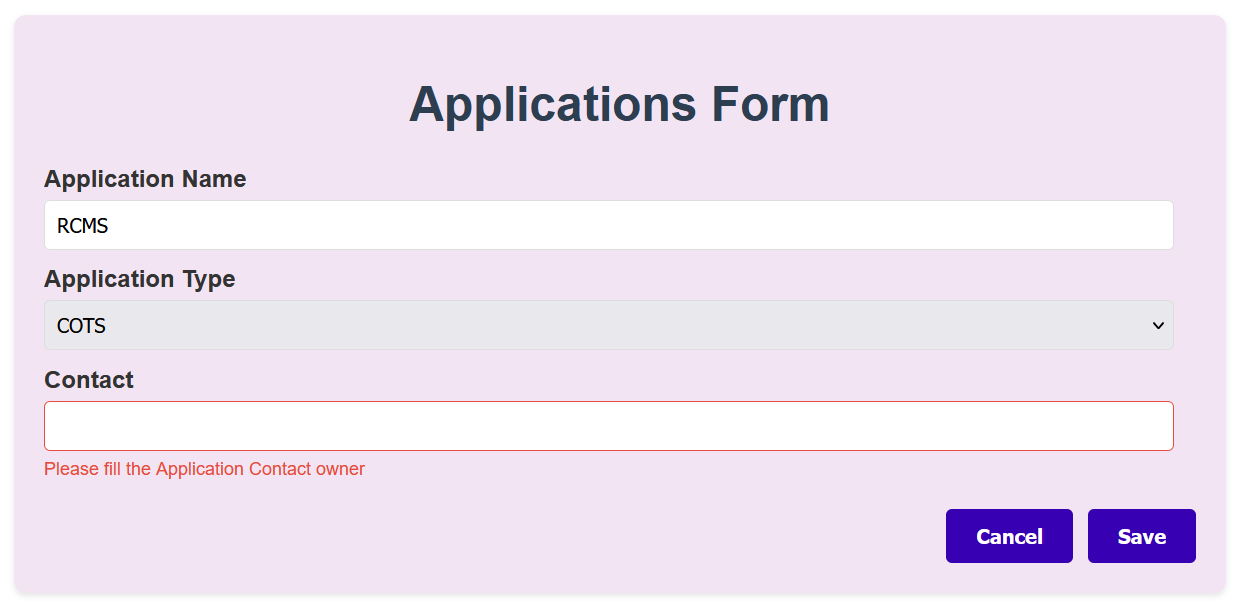
* + Test cases: Empty, too short (<3 chars), too long (>50 chars), special characters
  + Expected: Error message displayed, form not submitted

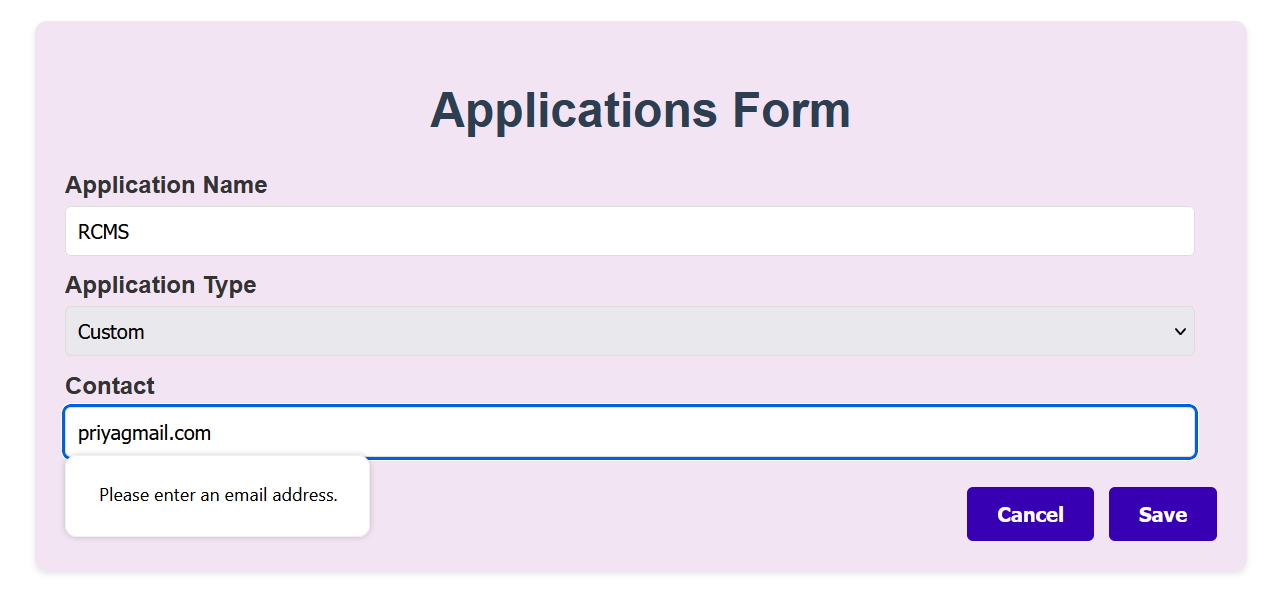


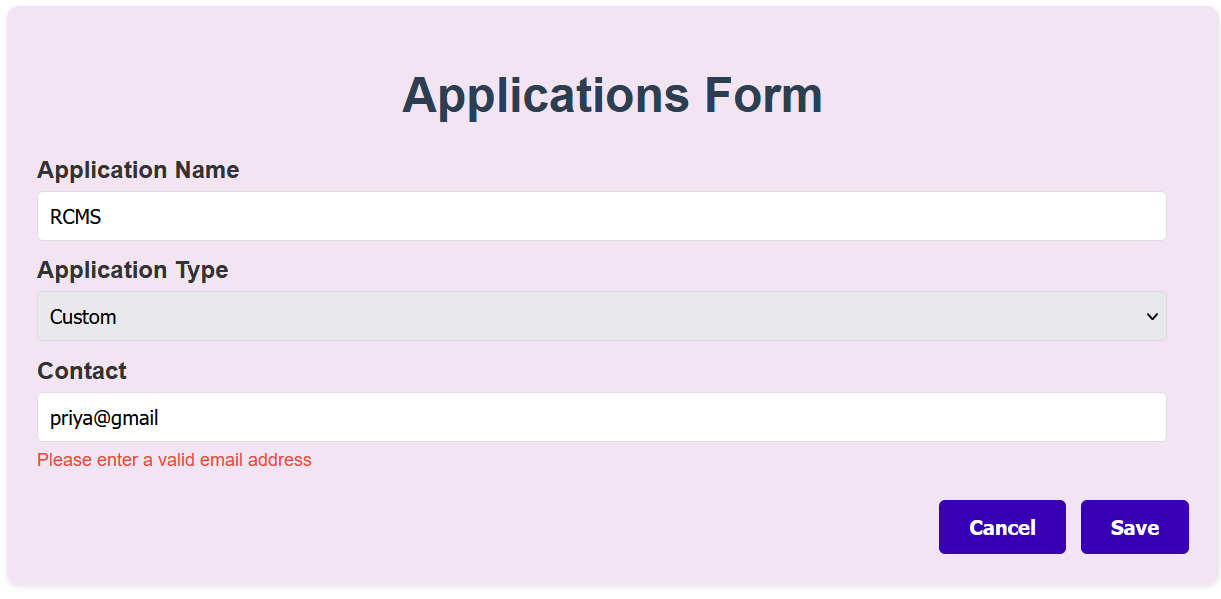


#### Email Validation Testing

* + Test cases: Invalid format, missing @, missing domain
  + Expected: Email error message shown

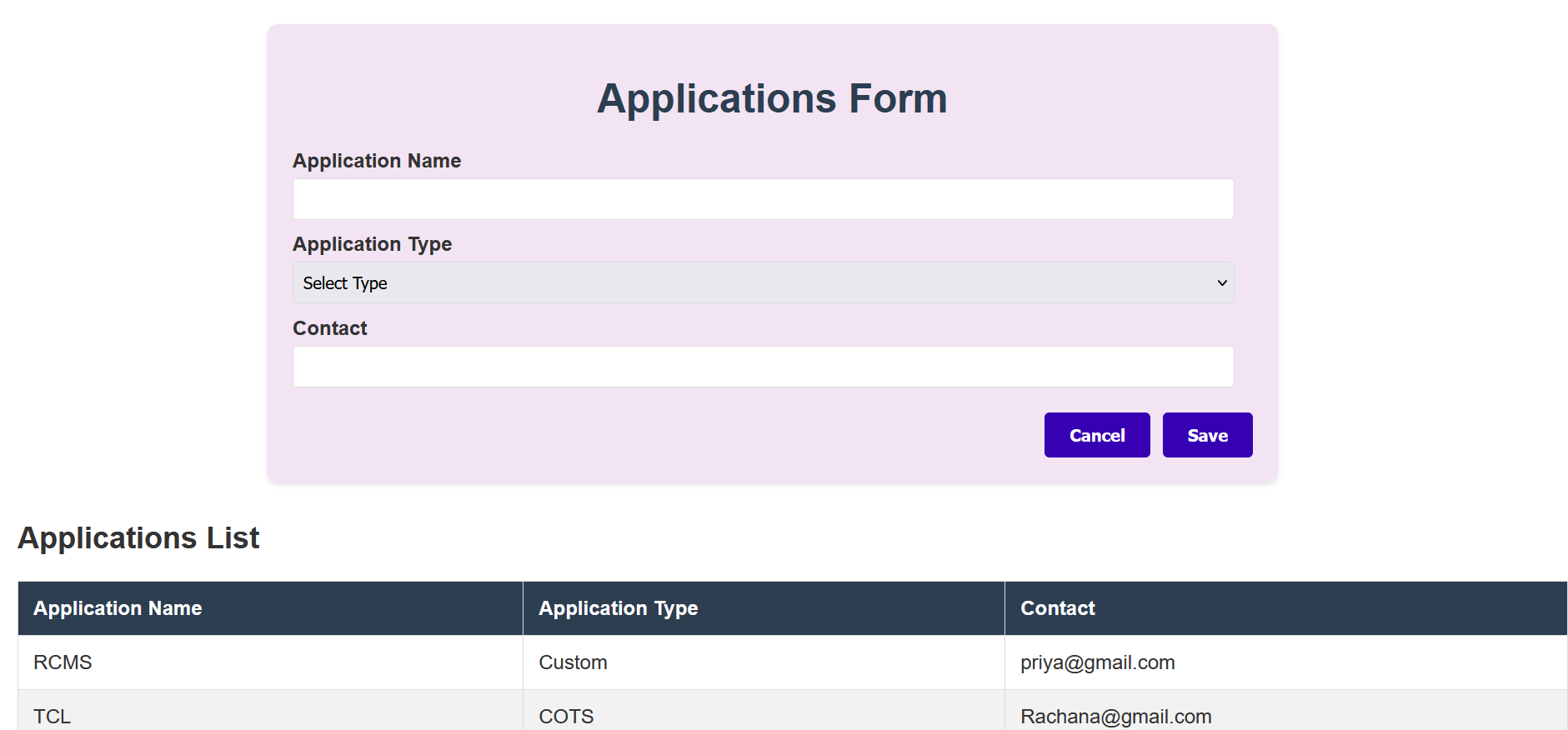






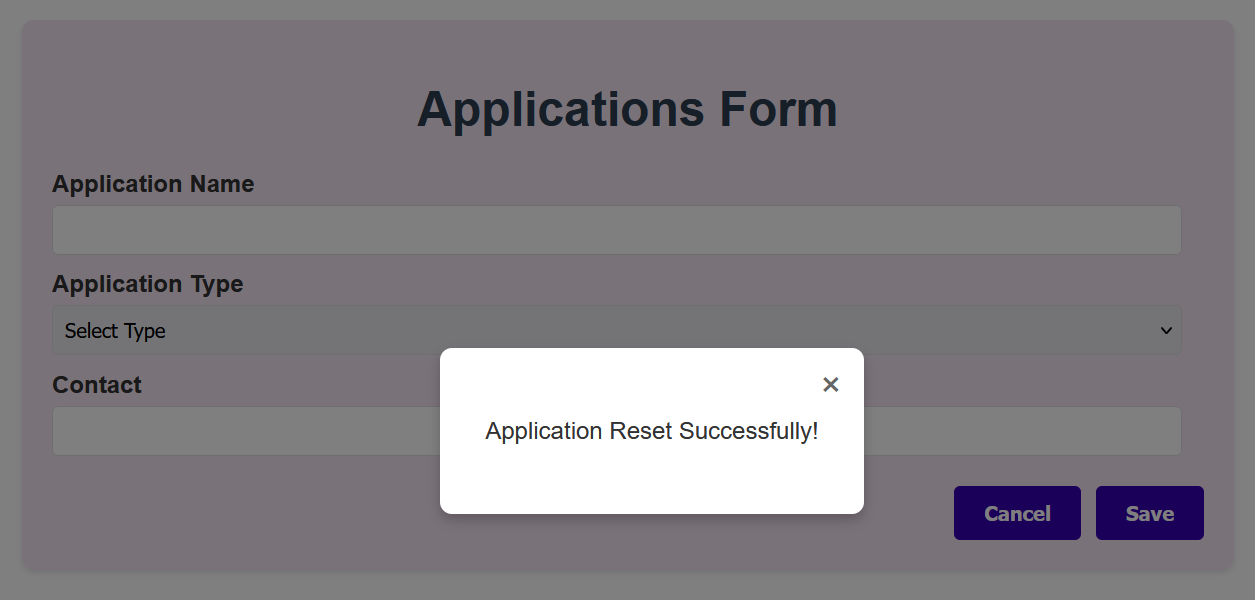
#### Data Persistence Testing

* + Action: Add application, refresh browser
  + Expected: Data remains in table



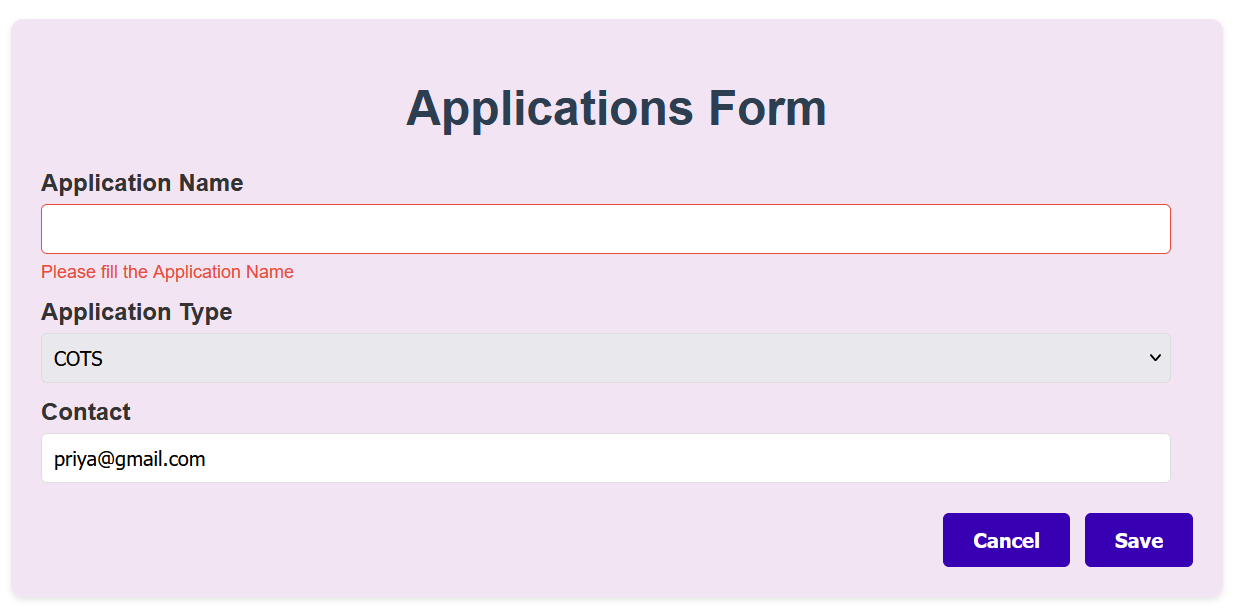
#### Cancel Functionality

* + Action: Fill form, click cancel
  + Expected: Form cleared, errors hidden



#### Empty Form

* Test required field validation



## Deployment Strategy

### Deployment Options

1.The code was developed in the VS Code tool in html,css and javascript and was hosted in local client environment.

2.No external or paid server was used.

## Conclusion

This Application Management System provides a solid foundation for application inventory management with a focus on simplicity and user experience. The client-side architecture ensures fast performance and easy deployment, while the modular design allows for future enhancements and scalability improvements.