```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, 
initial-scale=1.0">
    <title>Practice Homebase</title>
    <script src="https://apis.google.com/js/api.js"></script>
    <script src="https://accounts.google.com/gsi/client"></
script>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: 'Inter', -apple-system, 
BlinkMacSystemFont, 'Segoe UI', sans-serif;
            background: linear-gradient(135deg, #FCD5E5 0%, 
#A8F3E3 100%);
            min-height: 100vh;
            color: #2d1b69;
            line-height: 1.6;
        }

        .header {
            background: rgba(255, 255, 255, 0.95);
            backdrop-filter: blur(10px);
            padding: 1rem 2rem;
            box-shadow: 0 2px 20px rgba(162, 24, 85, 0.1);
            position: sticky;
            top: 0;
            z-index: 100;
        }

        .header-content {
            max-width: 1400px;
            margin: 0 auto;
            display: flex;
            justify-content: space-between;
            align-items: center;
        }

        .logo {
            display: flex;
            align-items: center;
            gap: 1rem;
        }

        .logo-icon {
            width: 50px;
            height: 50px;
            background: linear-gradient(135deg, #A21855,
#E13DA9);
            border-radius: 50%;
            display: flex;
            align-items: center;
            justify-content: center;
            color: white;
            font-weight: bold;
            font-size: 1.5rem;
        }
        .logo h1 {
            color: #A21855;
            font-size: 1.8rem;
            font-weight: 300;
        }

        .controls {
            display: flex;
            gap: 1rem;
            align-items: center;
        }

        .btn {
            padding: 0.5rem 1rem;
            border: none;
            border-radius: 8px;
            cursor: pointer;
            font-size: 0.9rem;
            transition: all 0.3s ease;
            text-decoration: none;
            display: inline-flex;
            align-items: center;
            gap: 0.5rem;
        }

        .btn-primary {
            background: linear-gradient(135deg, #A21855,
#E13DA9);
            color: white;
        }
```

```
.btn-secondary {
    background: white;
    color: #A21855;
    border: 2px solid #A21855;
}

.btn:hover {
    transform: translateY(-2px);
    box-shadow: 0 4px 15px rgba(162, 24, 85, 0.3);
}

.container {
    max-width: 1400px;
    margin: 0 auto;
    padding: 2rem;
}

.week-nav {
    display: flex;
    justify-content: center;
    margin-bottom: 2rem;
    background: rgba(255, 255, 255, 0.9);
    border-radius: 15px;
    padding: 1rem;
    box-shadow: 0 4px 20px rgba(162, 24, 85, 0.1);
}

.day-tab {
    padding: 0.75rem 1.5rem;
    margin: 0 0.25rem;
    border: none;
```

```
            background: transparent;
            border-radius: 10px;
            cursor: pointer;
            transition: all 0.3s ease;
            color: #6E3DA9;
            font-weight: 500;
        }

        .day-tab.active {
            background: linear-gradient(135deg, #A21855,
    #E13DA9);
            color: white;
            box-shadow: 0 4px 15px rgba(162, 24, 85, 0.3);
        }

        .day-tab.patient-day {
            border-left: 4px solid #53C3B8;
        }

        .day-tab.day-off {
            border-left: 4px solid #F4AFC9;
        }

        .day-content {
            display: none;
            animation: fadeIn 0.5s ease;
            width: 100%;
        }

        .day-content.active {
            display: block !important;
            opacity: 1;
```

```css
}

@keyframes fadeIn {
   from { opacity: 0; transform: translateY(10px); }
   to { opacity: 1; transform: translateY(0); }
}

.day-header {
   text-align: center;
   margin-bottom: 2rem;
}

.day-title {
   font-size: 2.5rem;
   color: #A21855;
   margin-bottom: 0.5rem;
   font-weight: 300;
}

.day-type {
   font-size: 1.2rem;
   color: #6E3DA9;
   font-weight: 500;
}

.patients-section {
   background: rgba(255, 255, 255, 0.95);
   border-radius: 20px;
   padding: 2rem;
   box-shadow: 0 8px 40px rgba(162, 24, 85, 0.1);
   margin-bottom: 2rem;
}
```

```
        .section-header {
            display: flex;
            justify-content: space-between;
            align-items: center;
            margin-bottom: 1.5rem;
        }

        .section-title {
            font-size: 1.5rem;
            color: #A21855;
            font-weight: 500;
        }

        .add-patient-btn {
            background: linear-gradient(135deg, #53C3B8,
#A8F3E3);
            color: #2d1b69;
            border: none;
            padding: 0.5rem 1rem;
            border-radius: 8px;
            cursor: pointer;
            font-weight: 500;
            transition: all 0.3s ease;
        }

        .add-patient-btn:hover {
            transform: translateY(-2px);
            box-shadow: 0 4px 15px rgba(83, 195, 184, 0.3);
        }

        .patient-card {
            background: linear-gradient(135deg, rgba(252,
213, 229, 0.3), rgba(168, 243, 227, 0.3));
            border: 2px solid rgba(162, 24, 85, 0.1);
            border-radius: 15px;
            padding: 1.5rem;
            margin-bottom: 1rem;
            position: relative;
            transition: all 0.3s ease;
        }

        .patient-card:hover {
            box-shadow: 0 6px 25px rgba(162, 24, 85, 0.15);
            transform: translateY(-2px);
        }

        .patient-header {
            display: flex;
            justify-content: space-between;
            align-items: center;
            margin-bottom: 1rem;
            padding-bottom: 0.5rem;
            border-bottom: 1px solid rgba(162, 24, 85, 0.2);
        }

        .patient-number {
            font-size: 1.2rem;
            font-weight: 600;
            color: #A21855;
        }

        .remove-patient {
            background: none;
            border: none;
            color: #E13DA9;
            cursor: pointer;
            font-size: 1.2rem;
            padding: 0.25rem;
            border-radius: 50%;
            transition: all 0.3s ease;
        }

        .remove-patient:hover {
            background: rgba(225, 61, 169, 0.1);
            transform: scale(1.1);
        }

        .form-grid {
            display: grid;
            grid-template-columns: repeat(auto-fit,
minmax(300px, 1fr));
            gap: 1rem;
        }

        .form-group {
            margin-bottom: 0.75rem;
        }

        .form-label {
            display: block;
            margin-bottom: 0.4rem;
            font-weight: 500;
            color: #6E3DA9;
            font-size: 0.9rem;
        }
```

```css
.checkbox-group {
    display: flex;
    gap: 1rem;
    flex-wrap: wrap;
}

.checkbox-item {
    display: flex;
    align-items: center;
    gap: 0.5rem;
}

.checkbox-item input[type="checkbox"] {
    width: 18px;
    height: 18px;
    accent-color: #A21855;
    cursor: pointer;
}

.form-input, .form-textarea {
    width: 100%;
    padding: 0.6rem;
    border: 2px solid rgba(110, 61, 169, 0.2);
    border-radius: 8px;
    font-size: 0.9rem;
    transition: all 0.3s ease;
    background: rgba(255, 255, 255, 0.8);
}

.form-input:focus, .form-textarea:focus {
    outline: none;
    border-color: #A21855;
    box-shadow: 0 0 0 3px rgba(162, 24, 85, 0.1);
}

.form-textarea {
    height: 60px;
    resize: vertical;
}

.todo-section {
    background: rgba(255, 255, 255, 0.95);
    border-radius: 20px;
    padding: 2rem;
    box-shadow: 0 8px 40px rgba(162, 24, 85, 0.1);
}

.todo-item {
    display: flex;
    align-items: center;
    gap: 0.75rem;
    padding: 0.75rem;
    margin-bottom: 0.5rem;
    background: rgba(244, 175, 201, 0.2);
    border-radius: 8px;
    transition: all 0.3s ease;
}

.todo-item:hover {
    background: rgba(244, 175, 201, 0.3);
}

.todo-checkbox {
        width: 20px;
        height: 20px;
        accent-color: #A21855;
    }

    .todo-input {
        flex: 1;
        border: none;
        background: transparent;
        font-size: 1rem;
        color: #2d1b69;
    }

    .todo-input:focus {
        outline: none;
    }

    .todo-input.completed {
        text-decoration: line-through;
        opacity: 0.7;
    }

    .add-todo {
        background: linear-gradient(135deg, #F4AFC9,
#FCD5E5);
        color: #A21855;
        border: 2px dashed rgba(162, 24, 85, 0.3);
        padding: 1rem;
        border-radius: 8px;
        width: 100%;
        cursor: pointer;
        font-size: 1rem;
```

```
            transition: all 0.3s ease;
        }

        .add-todo:hover {
            background: linear-gradient(135deg, #FCD5E5,
#F4AFC9);
            border-color: #A21855;
        }

        .auto-save-indicator {
            position: fixed;
            bottom: 20px;
            right: 20px;
            background: rgba(83, 195, 184, 0.9);
            color: white;
            padding: 0.5rem 1rem;
            border-radius: 25px;
            font-size: 0.8rem;
            opacity: 0;
            transition: all 0.3s ease;
            z-index: 1000;
        }

        .auto-save-indicator.show {
            opacity: 1;
        }

        .reset-notice {
            background: linear-gradient(135deg, rgba(244,
175, 201, 0.3), rgba(252, 213, 229, 0.3));
            border: 1px solid rgba(162, 24, 85, 0.2);
            border-radius: 10px;
```

```
    padding: 1rem;
    margin-bottom: 2rem;
    text-align: center;
    color: #6E3DA9;
}

@media (max-width: 768px) {
    .container {
        padding: 1rem;
    }

    .week-nav {
        flex-wrap: wrap;
        gap: 0.5rem;
    }

    .day-tab {
        flex: 1;
        min-width: 100px;
        text-align: center;
        font-size: 0.8rem;
    }

    .form-grid {
        grid-template-columns: 1fr;
    }

    .header-content {
        flex-direction: column;
        gap: 1rem;
    }
```

```
            .controls {
                flex-wrap: wrap;
            }
        }
    </style>
</head>
<body>
    <div class="header">
        <div class="header-content">
            <div class="logo">
                <div class="logo-icon">🪷</div>
                <h1>Practice Homebase</h1>
            </div>
            <div class="controls">
                <button class="btn btn-secondary"
onclick="exportData()">📊 Export</button>
                <button class="btn btn-secondary"
onclick="printDay()">🖨️ Print</button>
                <button class="btn btn-primary"
onclick="clearAllData()">🔄 Reset Week</button>
            </div>
        </div>
    </div>

    <div class="container">
        <div class="reset-notice">
            📅 Data automatically resets every Sunday at
midnight
        </div>

        <div class="week-nav">
            <button class="day-tab day-off active" data-
```

```
day="sunday">Sunday</button>
          <button class="day-tab day-off" data-
day="monday">Monday</button>
          <button class="day-tab patient-day" data-
day="tuesday">Tuesday</button>
          <button class="day-tab patient-day" data-
day="wednesday">Wednesday</button>
          <button class="day-tab day-off" data-
day="thursday">Thursday</button>
          <button class="day-tab patient-day" data-
day="friday">Friday</button>
          <button class="day-tab patient-day" data-
day="saturday">Saturday</button>
       </div>

       <!-- Sunday -->
       <div class="day-content active" id="sunday">
          <div class="day-header">
             <h2 class="day-title">Sunday</h2>
             <p class="day-type">Day Off - Planning &
Personal</p>
          </div>
          <div class="todo-section">
             <div class="section-header">
                <h3 class="section-title">📝 To-Do List</h3>
             </div>
             <div id="sunday-todos"></div>
             <button class="add-todo"
onclick="addTodo('sunday')">+ Add Task</button>
          </div>
       </div>
```

```
        <!-- Monday -->
        <div class="day-content" id="monday">
            <div class="day-header">
                <h2 class="day-title">Monday</h2>
                <p class="day-type">Day Off - Administrative
Tasks</p>
            </div>
            <div class="todo-section">
                <div class="section-header">
                    <h3 class="section-title">📋 Administrative
Tasks</h3>
                </div>
                <div id="monday-todos"></div>
                <button class="add-todo"
onclick="addTodo('monday')">+ Add Task</button>
            </div>
        </div>

        <!-- Tuesday -->
        <div class="day-content" id="tuesday">
            <div class="day-header">
                <h2 class="day-title">Tuesday</h2>
                <p class="day-type">Patient Day</p>
            </div>
            <div class="patients-section">
                <div class="section-header">
                    <h3 class="section-title">👥 Patients</h3>
                    <button class="add-patient-btn"
onclick="addPatient('tuesday')">+ Add Patient</button>
                </div>
                <div id="tuesday-patients"></div>
        </div>
      </div>

      <!-- Wednesday -->
      <div class="day-content" id="wednesday">
        <div class="day-header">
          <h2 class="day-title">Wednesday</h2>
          <p class="day-type">Patient Day</p>
        </div>
        <div class="patients-section">
          <div class="section-header">
            <h3 class="section-title">👥 Patients</h3>
            <button class="add-patient-btn"
onclick="addPatient('wednesday')">+ Add Patient</
button>
          </div>
          <div id="wednesday-patients"></div>
        </div>
      </div>

      <!-- Thursday -->
      <div class="day-content" id="thursday">
        <div class="day-header">
          <h2 class="day-title">Thursday</h2>
          <p class="day-type">Day Off - Personal Time</
p>
        </div>
        <div class="todo-section">
          <div class="section-header">
            <h3 class="section-title">🌸 Personal Tasks</
h3>
          </div>
```

```
            <div id="thursday-todos"></div>
            <button class="add-todo"
onclick="addTodo('thursday')">+ Add Task</button>
        </div>
    </div>

    <!-- Friday -->
    <div class="day-content" id="friday">
        <div class="day-header">
            <h2 class="day-title">Friday</h2>
            <p class="day-type">Patient Day</p>
        </div>
        <div class="patients-section">
            <div class="section-header">
                <h3 class="section-title">👥 Patients</h3>
                <button class="add-patient-btn"
onclick="addPatient('friday')">+ Add Patient</button>
            </div>
            <div id="friday-patients"></div>
        </div>
    </div>

    <!-- Saturday -->
    <div class="day-content" id="saturday">
        <div class="day-header">
            <h2 class="day-title">Saturday</h2>
            <p class="day-type">Patient Day</p>
        </div>
        <div class="patients-section">
            <div class="section-header">
                <h3 class="section-title">👥 Patients</h3>
                <button class="add-patient-btn"
```

```
onclick="addPatient('saturday')">+ Add Patient</button>
                </div>
                <div id="saturday-patients"></div>
            </div>
        </div>
    </div>

    <div class="auto-save-indicator"
id="autoSaveIndicator">
        ✅ Saved
    </div>

    <script>
        const patientDays = ['tuesday', 'wednesday', 'friday',
'saturday'];
        const dayOffDays = ['sunday', 'monday', 'thursday'];
        let currentDay = 'sunday';
        let patientCounters = {};

        // Initialize counters
        patientDays.forEach(day => {
            patientCounters[day] = 0;
        });

        // Initialize the app
        document.addEventListener('DOMContentLoaded',
function() {
            checkForWeeklyReset();
            loadAllData();
            initializeTabs();
        });
        function initializeTabs() {
            const tabs = document.querySelectorAll('.day-
tab');
            tabs.forEach(tab => {
                tab.addEventListener('click', function() {
                    const day = this.getAttribute('data-day');
                    switchDay(day);
                });
            });
        }

        function switchDay(day) {
            // Update tabs
            document.querySelectorAll('.day-tab').forEach(tab
=> tab.classList.remove('active'));
            document.querySelector(`[data-day="$
{day}"]`).classList.add('active');

            // Update content
            document.querySelectorAll('.day-
content').forEach(content =>
content.classList.remove('active'));

document.getElementById(day).classList.add('active');

            currentDay = day;
        }

        function addPatient(day) {
            patientCounters[day]++;
            const patientId = `${day}-patient-$
```

```
{patientCounters[day]}`;

            const patientCard = createPatientCard(patientId,
patientCounters[day]);
            document.getElementById(`${day}-
patients`).appendChild(patientCard);

            saveData();
        }

        function createPatientCard(patientId, patientNumber)
{
            const card = document.createElement('div');
            card.className = 'patient-card';
            card.innerHTML = `
                <div class="patient-header">
                    <span class="patient-number">Patient $
{patientNumber}</span>
                    <button class="remove-patient"
onclick="removePatient('${patientId}')">×</button>
                </div>
                <div class="form-grid">
                    <div class="form-group">
                        <label class="form-label">Visit Type</label>
                        <div class="checkbox-group">
                            <div class="checkbox-item">
                                <input type="checkbox" id="$
{patientId}-sud" onchange="saveData()">
                                <label for="${patientId}-sud">SUD</
label>
                            </div>
                            <div class="checkbox-item">
                <input type="checkbox" id="$
{patientId}-menopause" onchange="saveData()">
                <label for="${patientId}-
menopause">Menopause</label>
              </div>
            </div>
          </div>

          <div class="form-group">
            <label class="form-label">Visit Format</
label>
            <div class="checkbox-group">
              <div class="checkbox-item">
                <input type="checkbox" id="$
{patientId}-inperson" onchange="saveData()">
                <label for="${patientId}-inperson">In-
Person</label>
              </div>
              <div class="checkbox-item">
                <input type="checkbox" id="$
{patientId}-telehealth" onchange="saveData()">
                <label for="${patientId}-
telehealth">Telehealth</label>
              </div>
            </div>
          </div>

          <div class="form-group">
            <label class="form-label">Date of Last
Visit</label>
            <input type="text" class="form-input" id="$
{patientId}-lastvisit" placeholder="MM/DD/YYYY"
```

```
oninput="saveData()">
            </div>

            <div class="form-group">
              <label class="form-label">New Meds
Started?</label>
              <div class="checkbox-group">
                <div class="checkbox-item">
                  <input type="checkbox" id="$
{patientId}-newmeds-yes" onchange="saveData()">
                  <label for="${patientId}-newmeds-
yes">Yes</label>
                </div>
                <div class="checkbox-item">
                  <input type="checkbox" id="$
{patientId}-newmeds-no" onchange="saveData()">
                  <label for="${patientId}-newmeds-
no">No</label>
                </div>
              </div>
            </div>

            <div class="form-group">
              <label class="form-label">Previous Top-3
Complaints</label>
              <textarea class="form-textarea" id="$
{patientId}-complaints" placeholder="Enter previous
complaints..." oninput="saveData()"></textarea>
            </div>

            <div class="form-group">
              <label class="form-label">Current
```

```
Medication and Dose</label>
                    <textarea class="form-textarea" id="$
{patientId}-medication" placeholder="Enter current
medications..." oninput="saveData()"></textarea>
                </div>

                <div class="form-group">
                    <label class="form-label">Need UDS?</
label>
                    <div class="checkbox-group">
                        <div class="checkbox-item">
                            <input type="checkbox" id="$
{patientId}-uds-yes" onchange="saveData()">
                            <label for="${patientId}-uds-
yes">Yes</label>
                        </div>
                        <div class="checkbox-item">
                            <input type="checkbox" id="$
{patientId}-uds-no" onchange="saveData()">
                            <label for="${patientId}-uds-no">No</
label>
                        </div>
                    </div>
                </div>

                <div class="form-group">
                    <label class="form-label">Previous
Referral</label>
                    <div class="checkbox-group">
                        <div class="checkbox-item">
                            <input type="checkbox" id="$
{patientId}-referral-yes" onchange="saveData()">
```

```
                        <label for="${patientId}-referral-
yes">Yes</label>
                    </div>
                    <div class="checkbox-item">
                        <input type="checkbox" id="$
{patientId}-referral-no" onchange="saveData()">
                        <label for="${patientId}-referral-
no">No</label>
                    </div>
                </div>
                <input type="text" class="form-input" id="$
{patientId}-referral-details" placeholder="Referral
details..." oninput="saveData()" style="margin-top:
0.5rem;">
            </div>

            <div class="form-group">
                <label class="form-label">Memorable
About Last Visit</label>
                <textarea class="form-textarea" id="$
{patientId}-memorable" placeholder="Notes about last
visit..." oninput="saveData()"></textarea>
            </div>
        </div>
    `;

    card.id = patientId;
    return card;
}

function removePatient(patientId) {
    const card = document.getElementById(patientId);
```

```
        if (card) {
            card.remove();
            saveData();
        }
    }

    function addTodo(day) {
        const todoContainer =
document.getElementById(`${day}-todos`);
        const todoId = `${day}-todo-${Date.now()}`;

        const todoItem = document.createElement('div');
        todoItem.className = 'todo-item';
        todoItem.id = todoId;
        todoItem.innerHTML = `
            <input type="checkbox" class="todo-checkbox"
onchange="toggleTodo('${todoId}')">
            <input type="text" class="todo-input"
placeholder="Enter task..." oninput="saveData()">
            <button class="remove-patient"
onclick="removeTodo('${todoId}')">×</button>
        `;

        todoContainer.appendChild(todoItem);
        todoItem.querySelector('.todo-input').focus();
        saveData();
    }

    function removeTodo(todoId) {
        const todo = document.getElementById(todoId);
        if (todo) {
            todo.remove();
```

```
                saveData();
            }
        }

        function toggleTodo(todoId) {
            const todo = document.getElementById(todoId);
            const checkbox = todo.querySelector('.todo-
checkbox');
            const input = todo.querySelector('.todo-input');

            if (checkbox.checked) {
                input.classList.add('completed');
            } else {
                input.classList.remove('completed');
            }

            saveData();
        }

        function saveData() {
            const data = {
                patients: {},
                todos: {},
                lastSaved: new Date().toISOString()
            };

            // Save patient data
            patientDays.forEach(day => {
                const patients = [];
                const patientCards =
document.querySelectorAll(`#${day}-patients .patient-
card`);
```

```
            patientCards.forEach(card => {
                const patientData = {};
                const inputs = card.querySelectorAll('input,
textarea');

                inputs.forEach(input => {
                    if (input.type === 'checkbox') {
                        patientData[input.id] = input.checked;
                    } else {
                        patientData[input.id] = input.value;
                    }
                });

                patients.push({
                    id: card.id,
                    data: patientData
                });
            });

            data.patients[day] = patients;
        });

        // Save todo data
        [...patientDays, ...dayOffDays].forEach(day => {
            const todos = [];
            const todoItems =
document.querySelectorAll(`#${day}-todos .todo-item`);

            todoItems.forEach(item => {
                const checkbox = item.querySelector('.todo-
checkbox');
```

```
                const input = item.querySelector('.todo-
input');

                todos.push({
                    id: item.id,
                    text: input.value,
                    completed: checkbox.checked
                });
            });

            data.todos[day] = todos;
        });

        localStorage.setItem('practiceHomebaseData',
JSON.stringify(data));
        showAutoSaveIndicator();
    }

    function loadAllData() {
        const data =
JSON.parse(localStorage.getItem('practiceHomebaseDat
a') || '{}');

        if (data.patients) {
            // Load patient data
            patientDays.forEach(day => {
                const patients = data.patients[day] || [];
                const container =
document.getElementById(`${day}-patients`);
                container.innerHTML = '';

                patients.forEach((patient, index) => {
                    patientCounters[day] =
Math.max(patientCounters[day], index + 1);
                    const card = createPatientCard(patient.id,
index + 1);
                    container.appendChild(card);

                    // Restore form data
                    Object.keys(patient.data).forEach(inputId =>
{
                        const input =
document.getElementById(inputId);
                        if (input) {
                            if (input.type === 'checkbox') {
                                input.checked = patient.data[inputId];
                            } else {
                                input.value = patient.data[inputId];
                            }
                        }
                    });
                });
            });
        }

        if (data.todos) {
            // Load todo data
            [...patientDays, ...dayOffDays].forEach(day => {
                const todos = data.todos[day] || [];
                const container =
document.getElementById(`${day}-todos`);
                container.innerHTML = '';

                todos.forEach(todo => {
                    const todoItem =
document.createElement('div');
                    todoItem.className = 'todo-item';
                    todoItem.id = todo.id;
                    todoItem.innerHTML = `
                        <input type="checkbox" class="todo-
checkbox" onchange="toggleTodo('${todo.id}')" $
{todo.completed ? 'checked' : ''}>
                        <input type="text" class="todo-input $
{todo.completed ? 'completed' : ''}" value="${todo.text}"
oninput="saveData()">
                        <button class="remove-patient"
onclick="removeTodo('${todo.id}')">×</button>
                    `;
                    container.appendChild(todoItem);
                });
            });
        }
    }

    function checkForWeeklyReset() {
        const lastReset =
localStorage.getItem('practiceHomebaseLastReset');
        const now = new Date();
        const currentSunday = new Date(now);
        currentSunday.setDate(now.getDate() -
now.getDay());
        currentSunday.setHours(0, 0, 0, 0);

        if (!lastReset || new Date(lastReset) <
currentSunday) {
```

```
localStorage.removeItem('practiceHomebaseData');

localStorage.setItem('practiceHomebaseLastReset',
currentSunday.toISOString());

                // Reset counters
                patientDays.forEach(day => {
                    patientCounters[day] = 0;
                });
            }
        }

        function showAutoSaveIndicator() {
            const indicator =
document.getElementById('autoSaveIndicator');
            indicator.classList.add('show');

            setTimeout(() => {
                indicator.classList.remove('show');
            }, 2000);
        }

        function exportData() {
            const data =
JSON.parse(localStorage.getItem('practiceHomebaseDat
a') || '{}');
            const exportData = {
                ...data,
                exportDate: new Date().toISOString(),
                weekOf: getWeekOf()
            };
```

```
          const blob = new Blob([JSON.stringify(exportData,
null, 2)], {type: 'application/json'});
          const url = URL.createObjectURL(blob);
          const a = document.createElement('a');
          a.href = url;
          a.download = `practice-homebase-$
{getWeekOf()}.json`;
          document.body.appendChild(a);
          a.click();
          document.body.removeChild(a);
          URL.revokeObjectURL(url);
      }

      function getWeekOf() {
          const now = new Date();
          const sunday = new Date(now);
          sunday.setDate(now.getDate() - now.getDay());
          return sunday.toISOString().split('T')[0];
      }

      function printDay() {
          window.print();
      }

      function clearAllData() {
          if (confirm('Are you sure you want to reset all data
for this week? This cannot be undone.')) {

localStorage.removeItem('practiceHomebaseData');

              // Reset counters
              patientDays.forEach(day => {
                  patientCounters[day] = 0;
              });

              // Clear all patient containers
              patientDays.forEach(day => {
                  document.getElementById(`${day}-
patients`).innerHTML = '';
              });

              // Clear all todo containers
              [...patientDays, ...dayOffDays].forEach(day => {
                  document.getElementById(`${day}-
todos`).innerHTML = '';
              });

              showAutoSaveIndicator();
          }
      }

      // Print styles
      const printStyles = `
          <style media="print">
              .header, .week-nav, .controls, .add-patient-
btn, .add-todo, .remove-patient {
                  display: none !important;
              }

              .day-content:not(.active) {
                  display: none !important;
              }

              .day-content.active {
```

```
            display: block !important;
        }

        body {
            background: white !important;
            color: black !important;
        }

        .patient-card, .todo-section, .patients-section {
            background: white !important;
            border: 1px solid #ccc !important;
            box-shadow: none !important;
        }

        .day-title {
            color: #333 !important;
        }

        .section-title {
            color: #333 !important;
        }

        .auto-save-indicator, .reset-notice {
            display: none !important;
        }

        @page {
            margin: 0.5in;
        }
    </style>
`;
```

```
        document.head.insertAdjacentHTML('beforeend',
printStyles);

        // Google Calendar Integration
        const GOOGLE_API_KEY =
'AIzaSyCUFF18in3_trYTFWPvm8XqowYR4xudO54';
        const GOOGLE_CLIENT_ID = '656168979425-
uk7kqvrsmisd0l7ilrr9g7vrdk5c6lvj.apps.googleuserconten
t.com';
        const DISCOVERY_DOC = 'https://
www.googleapis.com/discovery/v1/apis/calendar/v3/rest';
        const SCOPES = 'https://www.googleapis.com/auth/
calendar.readonly';

        let gapi;
        let google;
        let tokenClient;
        let gapiInited = false;
        let gisInited = false;

        // Initialize Google APIs
        async function initializeGoogleAPIs() {
            await gapiLoaded();
            await gisLoaded();
        }

        async function gapiLoaded() {
            await new Promise((resolve) => {
                gapi.load('client', resolve);
            });

            await gapi.client.init({
                apiKey: GOOGLE_API_KEY,
                discoveryDocs: [DISCOVERY_DOC],
            });
            gapiInited = true;
            maybeEnableButtons();
        }

        function gisLoaded() {
            tokenClient =
google.accounts.oauth2.initTokenClient({
                client_id: GOOGLE_CLIENT_ID,
                scope: SCOPES,
                callback: '', // defined later
            });
            gisInited = true;
            maybeEnableButtons();
        }

        function maybeEnableButtons() {
            if (gapiInited && gisInited) {
                // Add calendar sync button to header
                addCalendarSyncButton();
            }
        }

        function addCalendarSyncButton() {
            const controls =
document.querySelector('.controls');
            const syncButton =
document.createElement('button');
            syncButton.className = 'btn btn-primary';
            syncButton.innerHTML = '📅 Sync Calendar';
            syncButton.onclick = handleAuthClick;
            controls.insertBefore(syncButton,
controls.firstChild);
        }

        function handleAuthClick() {
            tokenClient.callback = async (resp) => {
                if (resp.error !== undefined) {
                    throw (resp);
                }
                await loadCalendarEvents();
            };

            if (gapi.client.getToken() === null) {
                tokenClient.requestAccessToken({prompt:
'consent'});
            } else {
                tokenClient.requestAccessToken({prompt: ''});
            }
        }

        async function loadCalendarEvents() {
            try {
                // Get events for the current week
                const now = new Date();
                const startOfWeek = new Date(now);
                startOfWeek.setDate(now.getDate() -
now.getDay()); // Sunday
                startOfWeek.setHours(0, 0, 0, 0);

                const endOfWeek = new Date(startOfWeek);
                endOfWeek.setDate(startOfWeek.getDate() + 7);
                const request = {
                    'calendarId': 'primary',
                    'timeMin': startOfWeek.toISOString(),
                    'timeMax': endOfWeek.toISOString(),
                    'showDeleted': false,
                    'singleEvents': true,
                    'orderBy': 'startTime'
                };

                const response = await
gapi.client.calendar.events.list(request);
                const events = response.result.items;

                if (events && events.length > 0) {
                    populatePatientsFromCalendar(events);
                    showAutoSaveIndicator();
                } else {
                    alert('No events found for this week.');
                }
            } catch (err) {
                console.error('Error loading calendar events:',
err);
                alert('Error loading calendar events. Please try
again.');
            }
        }

        function populatePatientsFromCalendar(events) {
            // Clear existing patients
            patientDays.forEach(day => {
                document.getElementById(`${day}-
patients`).innerHTML = '';
                patientCounters[day] = 0;
            });

            // Group events by day
            const eventsByDay = {
                tuesday: [],
                wednesday: [],
                friday: [],
                saturday: []
            };

            events.forEach(event => {
                if (!event.start || !event.start.dateTime) return;

                const eventDate = new
Date(event.start.dateTime);
                const dayOfWeek = eventDate.getDay();

                // Map day numbers to our patient days
                const dayMapping = {
                    2: 'tuesday',    // Tuesday
                    3: 'wednesday',  // Wednesday
                    5: 'friday',     // Friday
                    6: 'saturday'    // Saturday
                };

                const dayName = dayMapping[dayOfWeek];
                if (dayName && eventsByDay[dayName]) {
                    eventsByDay[dayName].push(event);
                }
            });
```

```
          // Create patient cards for each event
          Object.keys(eventsByDay).forEach(day => {
             eventsByDay[day].forEach((event, index) => {
                patientCounters[day]++;
                const patientId = `${day}-patient-$
{patientCounters[day]}`;

                const patientCard =
createPatientCard(patientId, patientCounters[day]);
                document.getElementById(`${day}-
patients`).appendChild(patientCard);

                // Pre-populate with calendar event info
                const eventTime = new
Date(event.start.dateTime).toLocaleTimeString('en-US', {
                   hour: 'numeric',
                   minute: '2-digit',
                   hour12: true
                });

                // Add event title and time to memorable field
                const memorableField =
document.getElementById(`${patientId}-memorable`);
                if (memorableField) {
                   memorableField.value = `${event.summary ||
'Appointment'} at ${eventTime}`;
                }

                // Try to detect visit type from event title
                const title = (event.summary ||
'').toLowerCase();
```

```
                if (title.includes('sud') ||
title.includes('substance')) {
                    const sudCheckbox =
document.getElementById(`${patientId}-sud`);
                    if (sudCheckbox) sudCheckbox.checked =
true;
                }
                if (title.includes('menopause') ||
title.includes('hormone')) {
                    const menopauseCheckbox =
document.getElementById(`${patientId}-menopause`);
                    if (menopauseCheckbox)
menopauseCheckbox.checked = true;
                }

                // Detect if telehealth
                if (title.includes('telehealth') ||
title.includes('virtual') || title.includes('zoom')) {
                    const telehealthCheckbox =
document.getElementById(`${patientId}-telehealth`);
                    if (telehealthCheckbox)
telehealthCheckbox.checked = true;
                } else {
                    const inPersonCheckbox =
document.getElementById(`${patientId}-inperson`);
                    if (inPersonCheckbox)
inPersonCheckbox.checked = true;
                }
            });
        });

        saveData();
    }

    // Initialize when page loads
    document.addEventListener('DOMContentLoaded',
function() {
        checkForWeeklyReset();
        loadAllData();
        initializeTabs();

        // Initialize Google APIs if keys are provided
        if (GOOGLE_API_KEY !== 'YOUR_API_KEY_HERE'
&& GOOGLE_CLIENT_ID !== 'YOUR_CLIENT_ID_HERE') {
            initializeGoogleAPIs();
        }
    });

    // Google Calendar Integration placeholder
    // You can add your Google Calendar API integration
here
    function loadGoogleCalendarEvents(day) {
        // This function is now implemented above
        console.log(`Loading Google Calendar events for $
{day}`);
    }

    // Auto-save on page unload
    window.addEventListener('beforeunload', saveData);
</script>
</body>
</html>
```