```
<!DOCTYPE html>
<html lang="pl">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Genealog Indexer v6.0 - Field Entry Mode</title>
  <script
src="https://openseadragon.github.io/openseadragon/openseadragon.min.js"></script>
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css">

  <style>
    /* ===== RESET & ZMIENNE ===== */
    :root {
      --primary: #0078d4;
      --success: #10b981;
      --warning: #FFE66D;
      --danger: #c42b1c;
      --bg-dark: #0a0a0a;
      --bg-light: #1a1a1a;
      --bg-lighter: #252525;
      --border: #2a2a2a;
      --text-primary: #ddd;
      --text-secondary: #888;
    }

    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
    }

    html, body {
      width: 100%;
      height: 100%;
      font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', sans-serif;
      background-color: var(--bg-dark);
      color: var(--text-primary);
      overflow: hidden;
    }

    #app {
      display: flex;
      flex-direction: column;
      height: 100vh;
    }

    /* ===== TOOLBAR ===== */
```

```
.toolbar {
  background: linear-gradient(180deg, #1a1a1a 0%, #151515 100%);
  border-bottom: 1px solid var(--border);
  padding: 8px 12px;
  display: flex;
  align-items: center;
  gap: 8px;
  min-height: 44px;
  box-shadow: 0 2px 8px rgba(0,0,0,0.3);
  z-index: 100;
  flex-wrap: wrap;
}

.toolbar-title {
  font-size: 12px;
  color: var(--primary);
  font-weight: 600;
  text-transform: uppercase;
  letter-spacing: 0.5px;
  margin-right: auto;
}

.toolbar-btn {
  background: var(--bg-lighter);
  border: 1px solid var(--border);
  color: var(--text-secondary);
  padding: 6px 12px;
  border-radius: 4px;
  cursor: pointer;
  font-size: 11px;
  display: flex;
  align-items: center;
  gap: 6px;
  transition: all 0.15s ease;
  white-space: nowrap;
}

.toolbar-btn:hover {
  background: #2d2d2d;
  border-color: var(--primary);
  color: var(--primary);
}

.toolbar-btn.active {
  background: var(--primary);
  border-color: var(--primary);
  color: white;
  box-shadow: 0 0 12px rgba(0,120,212,0.4);
```

```
}

.toolbar-sep {
   width: 1px;
   height: 20px;
   background: var(--border);
   margin: 0 4px;
}

/* ===== MAIN CONTENT ===== */
.main-content {
   flex: 1;
   display: flex;
   gap: 0;
   overflow: hidden;
}

/* ===== VIEWER PANEL ===== */
.viewer-panel {
   flex: 1;
   background: #000;
   position: relative;
   display: flex;
   flex-direction: column;
   overflow: hidden;
}

.viewer-container {
   flex: 1;
   position: relative;
   overflow: hidden;
}

#viewer {
   width: 100%;
   height: 100%;
   cursor: grab;
}

#viewer.grabbing {
   cursor: grabbing;
}

#drawingCanvas {
   position: absolute;
   top: 0;
   left: 0;
   cursor: crosshair;
```

```
    z-index: 50;
    pointer-events: none;
}

#drawingCanvas.active {
    pointer-events: auto;
}

/* ===== FIELD ENTRY OVERLAY ===== */
.field-entry-overlay {
    position: absolute;
    top: 12px;
    left: 12px;
    background: rgba(10, 10, 10, 0.96);
    border: 2px solid var(--primary);
    border-radius: 8px;
    padding: 18px;
    max-width: 420px;
    color: white;
    z-index: 101;
    font-family: inherit;
    box-shadow: 0 12px 40px rgba(0, 0, 0, 0.6);
}

.overlay-header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 16px;
    font-weight: 600;
    font-size: 14px;
    color: var(--primary);
    border-bottom: 1px solid var(--border);
    padding-bottom: 10px;
}

.overlay-controls {
    display: flex;
    gap: 4px;
}

.overlay-controls button {
    padding: 4px 10px;
    background: var(--primary);
    border: none;
    color: white;
    border-radius: 3px;
    cursor: pointer;
    font-size: 12px;
    font-weight: 600;
}

.overlay-controls button:hover {
    background: #0060a0;
}

.overlay-field-current {
    margin: 12px 0;
}

.overlay-field-current label {
    display: block;
    font-size: 11px;
    color: var(--text-secondary);
    margin-bottom: 8px;
    text-transform: uppercase;
    font-weight: 600;
    letter-spacing: 0.3px;
}

.overlay-field-current input,
.overlay-field-current textarea {
    width: 100%;
    padding: 10px;
    background: var(--bg-dark);
    border: 1px solid var(--primary);
    border-radius: 4px;
    color: white;
    font-size: 13px;
    font-family: inherit;
    transition: all 0.2s ease;
}

.overlay-field-current input:focus,
.overlay-field-current textarea:focus {
    outline: none;
    border-color: var(--primary);
    box-shadow: 0 0 8px rgba(0, 120, 212, 0.3);
    background: var(--bg-light);
}

.overlay-field-actions {
    display: flex;
    gap: 6px;
    margin-top: 8px;
}
```

```
.overlay-field-actions button {
   padding: 6px 10px;
   font-size: 11px;
   cursor: pointer;
   background: var(--bg-lighter);
   border: 1px solid var(--border);
   color: var(--text-secondary);
   border-radius: 3px;
   transition: all 0.15s ease;
   flex: 1;
}

.overlay-field-actions button:hover {
   border-color: var(--primary);
   color: var(--primary);
}

.overlay-fields-preview {
   border-top: 1px solid var(--border);
   padding-top: 10px;
   margin-top: 12px;
   font-size: 11px;
   color: var(--text-secondary);
   max-height: 100px;
   overflow-y: auto;
}

.preview-field {
   padding: 6px 0;
   border-bottom: 1px solid rgba(42, 42, 42, 0.5);
}

.preview-field:last-child {
   border-bottom: none;
}

.overlay-actions {
   display: flex;
   gap: 6px;
   margin-top: 12px;
   border-top: 1px solid var(--border);
   padding-top: 10px;
}

.overlay-actions button {
   flex: 1;
   padding: 10px;
```

```
    background: var(--primary);
    border: none;
    color: white;
    border-radius: 4px;
    cursor: pointer;
    font-size: 11px;
    font-weight: 600;
    transition: all 0.15s ease;
}

.overlay-actions button:hover {
    background: #0060a0;
    transform: translateY(-1px);
}

/* ===== RIGHT PANEL ===== */
.right-panel {
    width: 360px;
    background: linear-gradient(180deg, #1a1a1a 0%, #151515 100%);
    border-left: 1px solid var(--border);
    display: flex;
    flex-direction: column;
    overflow: hidden;
    box-shadow: -4px 0 16px rgba(0,0,0,0.3);
}

.right-panel.collapsed {
    width: 0;
    border: none;
}

.panel-content {
    flex: 1;
    display: flex;
    flex-direction: column;
    overflow-y: auto;
}

.section-title {
    font-size: 10px;
    color: var(--text-secondary);
    font-weight: 600;
    text-transform: uppercase;
    letter-spacing: 0.5px;
    padding: 12px 12px 8px 12px;
}

/* ===== CONFIG SECTION ===== */
```

```
.config-section {
   padding: 12px;
   border-bottom: 1px solid var(--border);
}

.form-group {
   margin-bottom: 12px;
}

.form-group label {
   display: block;
   font-size: 10px;
   color: var(--text-secondary);
   margin-bottom: 5px;
   font-weight: 600;
   text-transform: uppercase;
   letter-spacing: 0.3px;
}

.form-group input,
.form-group select {
   width: 100%;
   padding: 8px 10px;
   background: var(--bg-dark);
   border: 1px solid var(--border);
   color: var(--text-primary);
   border-radius: 4px;
   font-size: 12px;
   font-family: inherit;
   transition: all 0.15s ease;
}

.form-group input:focus,
.form-group select:focus {
   border-color: var(--primary);
   outline: none;
   box-shadow: 0 0 0 3px rgba(0,120,212,0.1);
}

.form-group-inline {
   display: flex;
   gap: 8px;
}

.form-group-inline input {
   flex: 1;
}
```

```css
.form-group-inline button {
    padding: 8px 16px;
    background: var(--success);
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    font-size: 11px;
    font-weight: 600;
    transition: all 0.15s ease;
    white-space: nowrap;
}

.form-group-inline button:hover {
    background: #0d6a0d;
    transform: translateY(-1px);
}

/* ===== ACT BUTTONS SECTION ===== */
.act-buttons-section {
    padding: 12px;
    border-bottom: 1px solid var(--border);
}

.act-buttons-grid {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    gap: 8px;
    margin: 10px 0;
}

.act-button {
    width: 100%;
    aspect-ratio: 1;
    border-radius: 6px;
    border: 2px solid var(--border);
    background: var(--bg-lighter);
    color: var(--text-secondary);
    font-weight: 600;
    cursor: pointer;
    position: relative;
    transition: all 0.2s ease;
    display: flex;
    align-items: center;
    justify-content: center;
    font-size: 16px;
}
```

```
.act-button:hover {
   border-color: var(--primary);
   transform: scale(1.05);
}

.act-button.empty {
   background: var(--bg-dark);
   color: #555;
}

.act-button.selected {
   background: var(--primary);
   border-color: var(--primary);
   color: white;
   box-shadow: 0 0 12px rgba(0, 120, 212, 0.6);
}

.act-button.has-roi {
   background: var(--success);
   border-color: var(--success);
   color: white;
   box-shadow: 0 0 12px rgba(16, 185, 129, 0.4);
}

.act-button-counter {
   position: absolute;
   bottom: 2px;
   right: 2px;
   background: rgba(0, 120, 212, 0.8);
   color: white;
   font-size: 7px;
   padding: 2px 4px;
   border-radius: 2px;
   font-weight: 600;
}

/* ===== ACT SETTINGS ===== */
.act-settings {
   background: var(--bg-dark);
   padding: 10px;
   border-radius: 4px;
   font-size: 11px;
}

.act-settings label {
   display: flex;
   align-items: center;
   gap: 6px;
```

```
    margin: 6px 0;
    color: var(--text-secondary);
    cursor: pointer;
    transition: color 0.15s ease;
}

.act-settings label:hover {
    color: var(--text-primary);
}

.act-settings input[type="checkbox"] {
    cursor: pointer;
    width: 14px;
    height: 14px;
}

/* ===== THUMBNAILS BAR ===== */
.thumbnails-bar {
    height: 100px;
    background: var(--bg-dark);
    border-top: 1px solid var(--border);
    display: flex;
    align-items: center;
    padding: 8px;
    gap: 8px;
    overflow-x: auto;
    overflow-y: hidden;
    transition: height 0.3s ease;
    box-shadow: 0 -2px 8px rgba(0,0,0,0.3);
}

.thumbnails-bar.collapsed {
    height: 0;
    padding: 0;
    border: none;
    overflow: hidden;
}

.thumbnail {
    width: 84px;
    height: 84px;
    border: 2px solid var(--border);
    border-radius: 4px;
    cursor: pointer;
    overflow: hidden;
    background: var(--bg-light);
    flex-shrink: 0;
    transition: all 0.2s ease;
    position: relative;
}

.thumbnail img {
    width: 100%;
    height: 100%;
    object-fit: cover;
}

.thumbnail:hover {
    border-color: var(--primary);
    transform: scale(1.05);
}

.thumbnail.active {
    border-color: var(--primary);
    box-shadow: 0 0 16px rgba(0,120,212,0.6);
}

.thumbnail-number {
    position: absolute;
    top: 2px;
    left: 2px;
    background: rgba(0,0,0,0.7);
    color: #fff;
    font-size: 10px;
    padding: 2px 5px;
    border-radius: 2px;
    font-weight: 600;
}

/* ===== NOTIFICATIONS ===== */
.notification {
    position: fixed;
    bottom: 24px;
    right: 24px;
    padding: 12px 20px;
    background: var(--primary);
    color: white;
    border-radius: 6px;
    font-size: 12px;
    font-weight: 500;
    z-index: 1000;
    animation: slideIn 0.3s ease;
    box-shadow: 0 4px 20px rgba(0,0,0,0.4);
}

.notification.success {
    background: var(--success);
}

.notification.error {
    background: var(--danger);
}

.notification.warning {
    background: var(--warning);
    color: black;
}

@keyframes slideIn {
    from {
        transform: translateX(400px);
        opacity: 0;
    }
    to {
        transform: translateX(0);
        opacity: 1;
    }
}

/* ===== SCROLLBAR ===== */
::-webkit-scrollbar {
    width: 8px;
    height: 8px;
}

::-webkit-scrollbar-track {
    background: var(--bg-dark);
}

::-webkit-scrollbar-thumb {
    background: var(--border);
    border-radius: 4px;
}

::-webkit-scrollbar-thumb:hover {
    background: #4a4a4a;
}

/* ===== MODAL ===== */
.modal {
    display: none;
    position: fixed;
    z-index: 1000;
    left: 0;
    top: 0;
    width: 100%;
    height: 100%;
    background: rgba(0, 0, 0, 0.7);
    animation: fadeIn 0.3s ease;
}

.modal.active {
    display: flex;
    align-items: center;
    justify-content: center;
}

.modal-content {
    background: var(--bg-light);
    padding: 24px;
    border-radius: 8px;
    border: 1px solid var(--border);
    max-width: 400px;
    width: 90%;
    color: var(--text-primary);
}

.modal-header {
    font-size: 16px;
    font-weight: 600;
    margin-bottom: 16px;
    color: var(--primary);
    border-bottom: 1px solid var(--border);
    padding-bottom: 12px;
}

.modal-body {
    margin: 16px 0;
    font-size: 13px;
    line-height: 1.6;
}

.modal-actions {
    display: flex;
    gap: 8px;
    justify-content: flex-end;
    margin-top: 20px;
    border-top: 1px solid var(--border);
    padding-top: 16px;
}

.modal-btn {
    padding: 10px 20px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    font-size: 12px;
    font-weight: 600;
    transition: all 0.15s ease;
}

.modal-btn.primary {
    background: var(--primary);
    color: white;
}

.modal-btn.primary:hover {
    background: #0060a0;
}

.modal-btn.secondary {
    background: var(--bg-darker);
    color: var(--text-secondary);
    border: 1px solid var(--border);
}

.modal-btn.secondary:hover {
    border-color: var(--primary);
    color: var(--primary);
}

@keyframes fadeIn {
    from {
        opacity: 0;
    }
    to {
        opacity: 1;
    }
}

@media (max-width: 1024px) {
    .right-panel {
        width: 320px;
    }
}

@media (max-width: 768px) {
    .main-content {
        flex-direction: column;
    }
```

```
            .right-panel {
                width: 100%;
                max-height: 300px;
            }
            .field-entry-overlay {
                max-width: 95%;
            }
        }
    </style>
</head>
<body>
    <div id="app">
        <!-- TOOLBAR -->
        <div class="toolbar">
            <span class="toolbar-title">📚 Genealog Indexer v6.0</span>
            <button class="toolbar-btn" onclick="app.addImages()">
                <i class="fas fa-folder-open"></i> Dodaj obrazy
            </button>
            <div class="toolbar-sep"></div>
            <button class="toolbar-btn" id="actRoiBtn" onclick="app.toggleActROI()">
                <i class="fas fa-rectangle-landscape"></i> ACT ROI
            </button>
            <button class="toolbar-btn" id="fieldRoiBtn" onclick="app.toggleFieldROI()">
                <i class="fas fa-crop"></i> FIELD ROI
            </button>
            <div class="toolbar-sep"></div>
            <button class="toolbar-btn" onclick="app.rotateImage(90)">
                <i class="fas fa-undo"></i> Obrót
            </button>
            <button class="toolbar-btn" onclick="app.zoomReset()">
                <i class="fas fa-expand"></i> Reset
            </button>
            <div class="toolbar-sep"></div>
            <button class="toolbar-btn" onclick="app.exportData()">
                <i class="fas fa-download"></i> Eksport
            </button>
        </div>

        <!-- MAIN CONTENT -->
        <div class="main-content">
            <!-- VIEWER -->
            <div class="viewer-panel">
                <div class="viewer-container" id="viewerContainer">
                    <div id="viewer"></div>
                    <canvas id="drawingCanvas"></canvas>

                    <!-- FIELD ENTRY OVERLAY -->
                    <div id="fieldEntryOverlay" class="field-entry-overlay" style="display: none;">
```

```
                        <div class="overlay-header">
                            <span id="overlayTitle">AKT #1</span>
                            <div class="overlay-controls">
                                <button onclick="app.fieldNav.previousField()" title="Poprzednie
pole">←</button>
                                <button onclick="app.fieldNav.nextField()" title="Następne
pole">→</button>
                                <button onclick="app.fieldNav.completeEntry()" title="Zakończ
akt">✓</button>
                            </div>
                        </div>

                        <div class="overlay-field-current" id="overlayFieldCurrent">
                            <!-- Dynamicznie generowane -->
                        </div>

                        <div class="overlay-fields-preview" id="overlayFieldsPreview">
                            <!-- Preview następnych pól -->
                        </div>

                        <div class="overlay-actions">
                            <button onclick="app.fieldNav.previousField()">← Wróć</button>
                            <button onclick="app.fieldNav.skipFields()">Pomiń</button>
                            <button onclick="app.fieldNav.completeEntry()">Gotowe</button>
                        </div>
                    </div>
                </div>
            </div>

            <!-- RIGHT PANEL -->
            <div class="right-panel" id="rightPanel">
                <div class="panel-content">
                    <!-- CONFIG SECTION -->
                    <div class="config-section">
                        <div class="section-title">⚙️ Konfiguracja</div>
                        <div class="form-group">
                            <label>Parafia</label>
                            <input type="text" id="parafiaInput" value="Wiśniewa" />
                        </div>
                        <div class="form-group">
                            <label>Rok</label>
                            <input type="number" id="rokInput" value="1825" min="1600" max="2100"
/>
                        </div>
                        <div class="form-group">
                            <label>Typ aktu</label>
                            <select id="typAktuSelect">
                                <option value="birth">Urodzenia</option>
                            <option value="marriage">Małżeństwa</option>
                            <option value="death">Zgonu</option>
                        </select>
                    </div>
                    <div class="form-group">
                        <label>Liczba aktów na skanach</label>
                        <div class="form-group-inline">
                            <input type="number" id="actCountInput" value="1" min="1" max="20" />
                            <button onclick="app.initActs()">Utwórz</button>
                        </div>
                    </div>
                </div>

                <!-- ACT BUTTONS SECTION -->
                <div class="act-buttons-section">
                    <div class="section-title">🎯 Akty</div>
                    <div class="act-buttons-grid" id="actButtonsGrid">
                        <!-- Dynamicznie generowane -->
                    </div>

                    <div class="section-title" style="margin-top: 12px;">⚙️ Ustawienia</div>
                    <div class="act-settings">
                        <label>
                            <input type="checkbox" id="autoZoomAct" checked>
                            Auto-zoom do aktu
                        </label>
                        <label>
                            <input type="checkbox" id="seqFields" checked>
                            Pola sekwencyjnie
                        </label>
                        <label>
                            <input type="checkbox" id="rememberType" checked>
                            Zapamiętaj typ
                        </label>
                    </div>
                </div>
            </div>
        </div>
    </div>

    <!-- THUMBNAILS BAR -->
    <div class="thumbnails-bar collapsed" id="thumbsBar"></div>
</div>

<input type="file" id="fileInput" multiple accept="image/*" style="display: none;">

<script>
    // ===== APP OBJECT =====
    const app = {
        images: [],
        currentImageIdx: 0,
        currentActIdx: null,
        currentFieldIdx: 0,

        viewer: null,
        actROIMode: false,
        fieldROIMode: false,

        drawingCanvas: null,
        drawingCtx: null,
        drawingROI: false,
        roiStartX: 0,
        roiStartY: 0,

        currentImage: {
            file: null,
            parafia: "Wiśniewa",
            rok: 1825,
            typ: "birth",
            acts: []
        },

        templates: {},

        settings: {
            autoZoomAct: true,
            seqFields: true,
            rememberType: true
        },

        // ===== FIELD NAVIGATION =====
        fieldNav: {
            previousField() {
                if (app.currentFieldIdx > 0) {
                    app.fieldNav.saveField();
                    app.currentFieldIdx--;
                    app.renderFieldEntryOverlay();
                }
            },

            nextField() {
                app.fieldNav.saveField();
                const template = app.templates[app.currentImage.typ];
                if (app.currentFieldIdx < template.fields.length - 1) {
                    app.currentFieldIdx++;
                    app.renderFieldEntryOverlay();
```

```
          } else {
            app.fieldNav.completeEntry();
          }
        },

        skipFields() {
          if (confirm('Pominąć pozostałe pola w tym akcie?')) {
            app.fieldNav.completeEntry();
          }
        },

        completeEntry() {
          app.fieldNav.saveField();
          const act = app.currentImage.acts[app.currentActIdx];
          const template = app.templates[app.currentImage.typ];

          const filledCount = Object.keys(act.data).filter(k => act.data[k]).length;
          act.status = filledCount > 0 ? (filledCount === template.fields.length ? 'complete'
: 'partial') : 'empty';

          document.getElementById('fieldEntryOverlay').style.display = 'none';
          app.renderActButtons();
          app.saveToStorage();

          if (app.currentActIdx < app.currentImage.acts.length - 1) {
            const nextActNum = app.currentImage.acts[app.currentActIdx + 1].number;
            if (confirm(`Przejść do aktu #${nextActNum}?`)) {
              app.selectAct(app.currentActIdx + 1);
            }
          } else {
            app.notify(`✅ Ukończono wszystkie akty!`, 'success');
          }
        },

        saveField() {
          const template = app.templates[app.currentImage.typ];
          const field = template.fields[app.currentFieldIdx];
          const input = document.getElementById('fieldInput');
          if (input && app.currentActIdx !== null) {
            const value = input.value.trim();
            if (value) {
              app.currentImage.acts[app.currentActIdx].data[field.id] = value;
            }
          }
        },

        drawROI(fieldId) {
          app.fieldROIMode = true;
          app.drawingCanvas.classList.add('active');
          app.notify(`📍 Zaznacz ROI dla pola: ${fieldId}`, 'warning');
        }
      },

      // ===== INITIALIZATION =====
      init() {
        this.loadTemplates();
        this.initViewer();
        this.setupDrawingCanvas();
        this.setupEvents();
        this.loadFromStorage();
        this.notify('System gotowy', 'success');
      },

      loadTemplates() {
        this.templates = {
          birth: {
            id: "birth",
            name: "Akt Urodzenia",
            fields: [
              {id: "birth_date", label: "Data urodzenia", type: "date", required: true},
              {id: "child_first_name", label: "Imię dziecka", type: "text", required: true},
              {id: "child_last_name", label: "Nazwisko dziecka", type: "text", required:
true},
              {id: "birth_place", label: "Miejsce urodzenia", type: "text", required: false},
              {id: "father_first_name", label: "Imię ojca", type: "text", required: false},
              {id: "mother_first_name", label: "Imię matki", type: "text", required: false},
              {id: "notes", label: "Uwagi", type: "textarea", required: false}
            ]
          },
          marriage: {
            id: "marriage",
            name: "Akt Małżeństwa",
            fields: [
              {id: "groom_first_name", label: "Imię pana młodego", type: "text", required:
true},
              {id: "groom_last_name", label: "Nazwisko pana młodego", type: "text",
required: true},
              {id: "bride_first_name", label: "Imię panny młodej", type: "text", required:
true},
              {id: "bride_last_name", label: "Nazwisko panny młodej", type: "text",
required: true},
              {id: "marriage_date", label: "Data małżeństwa", type: "date", required:
false},
              {id: "marriage_place", label: "Miejsce małżeństwa", type: "text", required:
false},
              {id: "notes", label: "Uwagi", type: "textarea", required: false}
```

```
                        ]
                    },
                    death: {
                        id: "death",
                        name: "Akt Zgonu",
                        fields: [
                            {id: "deceased_first_name", label: "Imię zmarłego", type: "text", required:
true},
                            {id: "deceased_last_name", label: "Nazwisko zmarłego", type: "text",
required: true},
                            {id: "death_date", label: "Data zgonu", type: "date", required: false},
                            {id: "death_place", label: "Miejsce zgonu", type: "text", required: false},
                            {id: "age", label: "Wiek", type: "number", required: false},
                            {id: "notes", label: "Uwagi", type: "textarea", required: false}
                        ]
                    }
                };
            },

            initViewer() {
                this.viewer = OpenSeadragon({
                    id: 'viewer',
                    prefixUrl: 'https://openseadragon.github.io/openseadragon/images/',
                    minZoomLevel: 0.5,
                    maxZoomLevel: 20,
                });
            },

            setupDrawingCanvas() {
                this.drawingCanvas = document.getElementById('drawingCanvas');
                this.drawingCtx = this.drawingCanvas.getContext('2d', { willReadFrequently: true
});

                const container = document.getElementById('viewerContainer');
                const resize = () => {
                    this.drawingCanvas.width = container.offsetWidth;
                    this.drawingCanvas.height = container.offsetHeight;
                };
                resize();
                window.addEventListener('resize', resize);
            },

            setupEvents() {
                document.getElementById('fileInput').addEventListener('change', (e) => {
                    this.handleFiles(e.target.files);
                });

                document.addEventListener('keydown', (e) => {
```

```
            const overlay = document.getElementById('fieldEntryOverlay');
            if (overlay.offsetParent === null) return;

            if (e.key === 'Tab') {
                e.preventDefault();
                e.shiftKey ? this.fieldNav.previousField() : this.fieldNav.nextField();
            } else if (e.key === 'Enter') {
                e.preventDefault();
                this.fieldNav.nextField();
            } else if (e.key === 'Escape') {
                overlay.style.display = 'none';
            }
        });

        // Drag & drop
        const container = document.getElementById('viewerContainer');
        container.addEventListener('dragover', (e) => {
            e.preventDefault();
            container.style.borderColor = 'var(--primary)';
        });
        container.addEventListener('dragleave', () => {
            container.style.borderColor = '';
        });
        container.addEventListener('drop', (e) => {
            e.preventDefault();
            container.style.borderColor = '';
            this.handleFiles(e.dataTransfer.files);
        });
    },

    addImages() {
        document.getElementById('fileInput').click();
    },

    handleFiles(files) {
        const imageFiles = Array.from(files).filter(f => f.type.startsWith('image/'));
        if (imageFiles.length === 0) {
            this.notify('Nie znaleziono obrazów', 'error');
            return;
        }

        let loaded = 0;
        imageFiles.forEach(file => {
            const reader = new FileReader();
            reader.onload = (e) => {
                this.images.push({
                    name: file.name,
                    data: e.target.result
                });
                loaded++;
                this.renderThumbnails();
                if (this.images.length === 1) this.selectImage(0);
                if (loaded === imageFiles.length) {
                    this.notify(`✅ Wczytano ${imageFiles.length} obraz(y)`, 'success');
                }
            };
            reader.readAsDataURL(file);
        });
    },

    selectImage(idx) {
        if (idx < 0 || idx >= this.images.length) return;

        this.currentImageIdx = idx;
        this.currentImage.file = this.images[idx].name;
        this.currentImage.parafia = document.getElementById('parafiaInput').value;
        this.currentImage.rok = parseInt(document.getElementById('rokInput').value);
        this.currentImage.typ = document.getElementById('typAktuSelect').value;
        this.currentActIdx = null;
        this.currentFieldIdx = 0;

        if (this.viewer && this.images[idx]) {
            this.viewer.open({
                type: 'image',
                url: this.images[idx].data
            });
        }

        document.getElementById('fieldEntryOverlay').style.display = 'none';
        this.renderThumbnails();
        this.renderActButtons();
        this.saveToStorage();
    },

    renderThumbnails() {
        const bar = document.getElementById('thumbsBar');
        bar.innerHTML = '';
        this.images.forEach((img, i) => {
            const div = document.createElement('div');
            div.className = `thumbnail ${i === this.currentImageIdx ? 'active' : ''}`;
            div.onclick = () => this.selectImage(i);

            const num = document.createElement('div');
            num.className = 'thumbnail-number';
            num.textContent = i + 1;
        const imgEl = document.createElement('img');
        imgEl.src = img.data;
        imgEl.alt = img.name;

        div.appendChild(num);
        div.appendChild(imgEl);
        bar.appendChild(div);
    });

    if (this.images.length > 0) {
        bar.classList.remove('collapsed');
    }
},

initActs() {
    const count = parseInt(document.getElementById('actCountInput').value) || 1;
    this.currentImage.acts = [];

    for (let i = 0; i < count; i++) {
        this.currentImage.acts.push({
            id: i,
            number: i + 1,
            actROI: null,
            data: {},
            fieldROIs: {},
            status: "empty"
        });
    }

    this.currentActIdx = null;
    this.currentFieldIdx = 0;
    this.renderActButtons();
    this.notify(`✅ Utworzono ${count} aktów`, 'success');
    this.saveToStorage();
},

selectAct(actIdx) {
    if (actIdx < 0 || actIdx >= this.currentImage.acts.length) return;

    this.currentActIdx = actIdx;
    this.currentFieldIdx = 0;
    const act = this.currentImage.acts[actIdx];
    act.status = act.status === 'empty' ? 'partial' : act.status;

    if (this.settings.autoZoomAct && act.actROI) {
        this.zoomToActROI(act.actROI);
    }
```

```
                this.renderActButtons();
                this.renderFieldEntryOverlay();
                document.getElementById('fieldEntryOverlay').style.display = 'block';
                this.saveToStorage();
            },

            renderActButtons() {
                const grid = document.getElementById('actButtonsGrid');
                grid.innerHTML = '';

                if (this.currentImage.acts.length === 0) {
                    grid.innerHTML = '<p style="grid-column: 1/-1; text-align: center; color:
var(--text-secondary); padding: 20px;">Brak aktów. Ustaw liczbę i kliknij "Utwórz"</p>';
                    return;
                }

                this.currentImage.acts.forEach((act, idx) => {
                    const btn = document.createElement('button');
                    btn.className = `act-button ${act.status}`;
                    btn.textContent = act.number;
                    btn.onclick = () => this.selectAct(idx);
                    btn.title = `Akt #${act.number} - ${act.status}`;

                    const counter = document.createElement('div');
                    counter.className = 'act-button-counter';
                    const template = this.templates[this.currentImage.typ];
                    const filled = Object.keys(act.data).filter(k => act.data[k]).length;
                    counter.textContent = `${filled}/${template.fields.length}`;
                    btn.appendChild(counter);

                    grid.appendChild(btn);
                });
            },

            renderFieldEntryOverlay() {
                if (this.currentActIdx === null) return;

                const act = this.currentImage.acts[this.currentActIdx];
                const template = this.templates[this.currentImage.typ];
                const field = template.fields[this.currentFieldIdx];

                document.getElementById('overlayTitle').textContent = `AKT #${act.number} -
${template.name}`;

                const currentFieldHtml = `
                    <label>[${this.currentFieldIdx + 1}/${template.fields.length}]
${field.label}${field.required ? ' *' : ''}</label>
```

```
          <input type="text" id="fieldInput" value="${act.data[field.id] || ''}"
placeholder="Wpisz wartość lub TAB">
          <div class="overlay-field-actions">
            <button onclick="app.fieldNav.drawROI('${field.id}')">📍 ROI</button>
            <button onclick="app.fieldNav.clearField('${field.id}')">×</button>
          </div>
        `;
        document.getElementById('overlayFieldCurrent').innerHTML = currentFieldHtml;

        let previewHtml = '';
        for (let i = this.currentFieldIdx + 1; i < Math.min(this.currentFieldIdx + 3,
template.fields.length); i++) {
          const nextField = template.fields[i];
          previewHtml += `<div class="preview-field">[${i + 1}] ${nextField.label}</div>`;
        }
        document.getElementById('overlayFieldsPreview').innerHTML = previewHtml;

        setTimeout(() => {
          const input = document.getElementById('fieldInput');
          if (input) input.focus();
        }, 100);
      },

      clearField(fieldId) {
        if (this.currentActIdx !== null) {
          delete this.currentImage.acts[this.currentActIdx].data[fieldId];
          this.renderFieldEntryOverlay();
          this.saveToStorage();
        }
      },

      zoomToActROI(roi) {
        if (!this.viewer || !this.viewer.world.getItemAt(0)) return;

        const item = this.viewer.world.getItemAt(0);
        const itemSize = item.getContentSize();
        const rect = new OpenSeadragon.Rect(
          roi.x * itemSize.x,
          roi.y * itemSize.y,
          roi.w * itemSize.x,
          roi.h * itemSize.y
        );
        const viewportRect = this.viewer.viewport.imageToViewportRectangle(rect);
        this.viewer.viewport.fitBounds(viewportRect, true);
      },

      toggleActROI() {
        this.actROIMode = !this.actROIMode;
        document.getElementById('actRoiBtn').classList.toggle('active', this.actROIMode);
        this.drawingCanvas.classList.toggle('active', this.actROIMode);

        if (this.actROIMode) {
          this.notify('🎨 Zaznaczaj granice aktów (rysuj prostokąty)', 'warning');
        } else {
          this.notify('Wyjście z trybu ACT ROI', 'info');
        }
      },

      toggleFieldROI() {
        this.fieldROIMode = !this.fieldROIMode;
        document.getElementById('fieldRoiBtn').classList.toggle('active',
this.fieldROIMode);
        this.drawingCanvas.classList.toggle('active', this.fieldROIMode);

        if (this.fieldROIMode) {
          this.notify('📍 Zaznaczaj pola formularza', 'warning');
        } else {
          this.notify('Wyjście z trybu FIELD ROI', 'info');
        }
      },

      rotateImage(degrees) {
        if (!this.viewer) return;
        const current = this.viewer.viewport.getRotation();
        this.viewer.viewport.setRotation((current + degrees + 360) % 360);
      },

      zoomReset() {
        if (this.viewer && this.viewer.world.getItemAt(0)) {
          this.viewer.viewport.fitBounds(this.viewer.world.getItemAt(0).getBounds());
        }
      },

      exportData() {
        if (this.images.length === 0 || this.currentImage.acts.length === 0) {
          this.notify('❌ Brak danych do eksportu', 'error');
          return;
        }

        const csv = this.convertToCSV();
        const json = JSON.stringify(this.currentImage, null, 2);

        this.downloadFile(csv, `genealog-${new Date().toISOString().split('T')[0]}.csv`,
'text/csv');
        this.downloadFile(json, `genealog-${new Date().toISOString().split('T')[0]}.json`,
'application/json');
```

```
            this.notify(`✅ Eksportowano ${this.currentImage.acts.length} aktów`, 'success');
        },

        convertToCSV() {
            let csv = 'Lp.,Akt nr,Plik,Parafia,Rok,Typ,Status\n';

            this.currentImage.acts.forEach((act, idx) => {
                const fields = Object.entries(act.data).map(([k, v]) => `"${v}"`).join(',');
                csv += `${idx + 1},${act.number},"${this.currentImage.file}","${this.currentImage.parafia}",${this.currentImage.rok},"${this.templates[this.currentImage.typ].name}","${act.status}"\n`;
            });

            return csv;
        },

        downloadFile(content, filename, mimetype) {
            const blob = new Blob([content], { type: mimetype });
            const link = document.createElement('a');
            link.href = URL.createObjectURL(blob);
            link.download = filename;
            link.click();
            URL.revokeObjectURL(link.href);
        },

        notify(message, type = 'info') {
            const div = document.createElement('div');
            div.className = `notification ${type}`;
            div.textContent = message;
            document.body.appendChild(div);
            setTimeout(() => div.remove(), 3000);
        },

        saveToStorage() {
            localStorage.setItem('genealog_data', JSON.stringify({
                images: this.images.map(img => ({ name: img.name, data: img.data })),
                currentImageIdx: this.currentImageIdx,
                currentImage: this.currentImage
            }));
        },

        loadFromStorage() {
            const stored = localStorage.getItem('genealog_data');
            if (stored) {
                try {
                    const data = JSON.parse(stored);
                    this.images = data.images || [];
```

```
                        this.currentImageIdx = data.currentImageIdx || 0;
                        this.currentImage = data.currentImage || this.currentImage;
                        this.renderThumbnails();
                        if (this.images.length > 0) {
                            this.selectImage(this.currentImageIdx);
                        }
                    } catch (e) {
                        console.error('Błąd ładowania z localStorage:', e);
                    }
                }
            }
        };

        // Initialize
        document.addEventListener('DOMContentLoaded', () => {
            app.init();
        });
    </script>
</body>
</html>
```