Bhotch CRM - Developer Guide

Complete Coding Reference - Tab-by-Tab Breakdown

Table of Contents

---

1. Dashboard Tab

File Structure

```
src/features/dashboard/
└── DashboardView.jsx  (Main component)
```

Component API

```
function DashboardView({ stats, leads, jobCounts, onNavigateToTab })
```

Props:

Prop	Type	Description
stats	Object	Aggregated statistics
leads	Array	Array of lead objects
jobCounts	Array	Array of job count objects
onNavigateToTab	Function	Callback to switch tabs

Stats Object Structure

```
const stats = {
  totalLeads: Number,          // Total number of leads
  hotLeads: Number,            // Leads with quality "Hot"
  quotedLeads: Number,         // Leads with disposition "Quoted"
  totalQuoteValue: Number,     // Sum of all quotes
  totalJobCounts: Number,      // Total job counts
  totalSqFt: Number           // Sum of all square footage
};
```

Key Functions

formatCurrency(value)

```
const formatCurrency = (value) => {
  return new Intl.NumberFormat('en-US', {
    style: 'currency',
    currency: 'USD',
    minimumFractionDigits: 0,
    maximumFractionDigits: 0
  }).format(value);
};
```

formatNumber(value)

```
const formatNumber = (value) => {
  return new Intl.NumberFormat('en-US').format(value);
};
```

Calculated Metrics

```
// In component body
const scheduledLeads = leads.filter(l => l.disposition === 'Scheduled').length;
const followUpLeads = leads.filter(l => l.disposition === 'Follow Up').length;
const insuranceLeads = leads.filter(l => l.disposition === 'Insurance').length;
const closedSold = leads.filter(l => l.disposition === 'Closed Sold').length;
const conversionRate = stats.totalLeads > 0
  ? ((closedSold / stats.totalLeads) * 100).toFixed(1)
  : '0.0';
```

Component Sections

1. Primary Stats Grid

```
<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-6">
  <StatCard title="Total Leads" value={stats.totalLeads} icon={Users} color="blue" />
  <StatCard title="Hot Leads" value={stats.hotLeads} icon={TrendingUp} color="red" />
  <StatCard title="Quoted Leads" value={stats.quotedLeads} icon={Target} color="green" />
  <StatCard title="Total Quote Value" value={formatCurrency(stats.totalQuoteValue)} icon={DollarSign} color="purple" />
</div>
```

2. Secondary Stats

```
<div className="grid grid-cols-2 md:grid-cols-4 gap-4">
  {/* Scheduled, Follow Up, Insurance, Conversion */}
</div>
```

3. Job Count Metrics (Gradient Cards)

```
<div className="grid grid-cols-1 md:grid-cols-3 gap-6">
  <div className="bg-gradient-to-br from-blue-500 to-blue-600 rounded-lg shadow-lg p-6 text-white">
    {/* Total Job Counts */}
  </div>
</div>
```

4. Quick Actions Panel

```
<div className="grid grid-cols-2 md:grid-cols-4 gap-3">
  <button onClick={() => onNavigateToTab('leads')}>View Leads</button>
  <button onClick={() => onNavigateToTab('jobcount')}>Job Counts</button>
  <button onClick={() => onNavigateToTab('map')}>Map View</button>
  <button onClick={() => onNavigateToTab('calendar')}>Calendar</button>
</div>
```

5. Recent Activity (Two-Column)

```
<div className="grid grid-cols-1 lg:grid-cols-2 gap-6">
  {/* Recent Leads */}
  <div>...</div>
  {/* Recent Job Counts */}
  <div>...</div>
</div>
```

6. Sales Pipeline Overview

```
<div className="grid grid-cols-2 sm:grid-cols-3 md:grid-cols-6 gap-4">
  {/* New, Scheduled, Insurance, Quoted, Follow Up, Closed Sold */}
</div>
```

Styling Patterns

Gradient Cards:

```
className="bg-gradient-to-br from-blue-500 to-blue-600 rounded-lg shadow-lg p-6 text-white"
```

Hover Effects:

```
className="hover:bg-gray-100 transition-colors"
```

Responsive Text:

```
className="text-2xl font-bold text-gray-900"
```

---

2. Leads Tab

File Structure

```
src/features/leads/
├── LeadsView.jsx           (Main table)
├── LeadFormModal.jsx       (Add/Edit form)
├── LeadDetailModal.jsx     (Detail view)
├── CommunicationModal.jsx  (Communication log)
└── HouseVisualization.jsx  (3D preview)
```

LeadsView Component

```
function LeadsView({ leads, onAddLead, onEditLead, onDeleteLead, onRefreshLeads, onSelectLead })
```

State Management

```
const [searchTerm, setSearchTerm] = useState('');
const [filterDate, setFilterDate] = useState('');
const [sortConfig, setSortConfig] = useState({ key: null, direction: 'asc' });
const [columnFilters, setColumnFilters] = useState({});
const [showFilters, setShowFilters] = useState(false);
const [currentPage, setCurrentPage] = useState(1);
const [itemsPerPage, setItemsPerPage] = useState(25);
const [visibleColumns, setVisibleColumns] = useState(loadSavedColumns());
```

Column Configuration

```
const availableColumns = [
  // Basic Information
  { key: 'id', label: 'ID', type: 'text', category: 'Basic' },
  { key: 'date', label: 'Date', type: 'date', category: 'Basic' },
  { key: 'customerName', label: 'Customer Name', type: 'text', category: 'Basic' },
  { key: 'phoneNumber', label: 'Phone Number', type: 'text', category: 'Basic' },
  { key: 'email', label: 'Email', type: 'text', category: 'Basic' },
  { key: 'address', label: 'Address', type: 'text', category: 'Basic' },

  // Lead Information
  { key: 'quality', label: 'Quality', type: 'text', category: 'Lead Info' },
  { key: 'disposition', label: 'Disposition', type: 'text', category: 'Lead Info' },
  { key: 'leadSource', label: 'Lead Source', type: 'text', category: 'Lead Info' },

  // Measurements
  { key: 'sqFt', label: 'SQ FT', type: 'number', category: 'Measurements' },
  { key: 'ridgeLf', label: 'Ridge LF', type: 'number', category: 'Measurements' },
  { key: 'valleyLf', label: 'Valley LF', type: 'number', category: 'Measurements' },

  // Financial
  { key: 'dabellaQuote', label: 'DaBella Quote', type: 'number', category: 'Financial' },

  // ... 40+ more columns
];
```

Key Functions

1. handleSort(key)

```
const handleSort = useCallback((key) => {
  setSortConfig(prevConfig => ({
    key,
    direction: prevConfig.key === key && prevConfig.direction === 'asc' ? 'desc' : 'asc'
  }));
}, []);
```

2. filteredAndSortedLeads

```
const filteredAndSortedLeads = useMemo(() => {
  let filtered = leads.filter(lead => {
    // Global search
    const matchesSearch = !searchTerm || [
      lead.firstName,
      lead.lastName,
      lead.customerName,
      lead.phoneNumber?.toString(),
      lead.address
    ].some(field => field?.toLowerCase().includes(searchTerm.toLowerCase()));

    // Date filter
    const matchesDate = !filterDate || lead.date === filterDate;

    // Column filters
    const matchesColumnFilters = Object.entries(columnFilters).every(([column, filterValue]) => {
      // ... complex filtering logic
    });

    return matchesSearch && matchesDate && matchesColumnFilters;
  });

  // Sorting
  if (sortConfig.key) {
    filtered.sort((a, b) => {
      const aValue = getSortValue(a, sortConfig.key);
      const bValue = getSortValue(b, sortConfig.key);
      // ... sorting logic
    });
  }

  return filtered;
}, [leads, searchTerm, filterDate, columnFilters, sortConfig]);
```

3. paginatedLeads

```
const paginatedLeads = useMemo(() => {
  const startIndex = (currentPage - 1) * itemsPerPage;
  const endIndex = startIndex + itemsPerPage;
  return filteredAndSortedLeads.slice(startIndex, endIndex);
}, [filteredAndSortedLeads, currentPage, itemsPerPage]);
```

4. Column Visibility (localStorage)

```
const loadSavedColumns = () => {
  try {
    const saved = localStorage.getItem('leadsVisibleColumns');
    if (saved) return JSON.parse(saved);
  } catch (error) {
    console.error('Error loading saved columns:', error);
  }
  return defaultVisibleColumns;
};

const handleSaveColumns = useCallback(() => {
  try {
    localStorage.setItem('leadsVisibleColumns', JSON.stringify(tempVisibleColumns));
    setVisibleColumns(tempVisibleColumns);
    setShowColumnSettings(false);
  } catch (error) {
    console.error('Error saving columns:', error);
  }
}, [tempVisibleColumns]);
```

Components

SortableHeader

```
const SortableHeader = ({ sortKey, children, className = "" }) => {
  const hasFilter = columnFilters[sortKey];
  return (
    <th className={`px-6 py-4 text-left text-xs font-medium text-gray-500 uppercase ${className}`}>
      <div className="space-y-2">
        <div onClick={() => handleSort(sortKey)} className="cursor-pointer">
          <span className={hasFilter ? 'text-blue-600 font-semibold' : ''}>{children}</span>
          {/* Sort icons */}
        </div>
        {showFilters && <ColumnFilter column={sortKey} />}
      </div>
    </th>
  );
};
```

ColumnFilter

```
const ColumnFilter = ({ column, value, onChange, type }) => {
  return (
    <input
      type={type === 'number' ? 'number' : type === 'date' ? 'date' : 'text'}
      value={value}
      onChange={(e) => onChange(e.target.value)}
      placeholder={type === 'number' ? 'Min value...' : 'Filter...'}
      className="w-full px-2 py-1 text-xs border border-gray-300 rounded"
    />
  );
};
```

LeadFormModal Component

```
function LeadFormModal({ lead, isOpen, onClose, onSave })
```

Form Structure:

```
<form onSubmit={handleSubmit}>
  {/* Basic Information */}
  <section>
    <input name="customerName" />
    <input name="phoneNumber" />
    <input name="email" />
    <input name="address" />
  </section>

  {/* Lead Details */}
  <section>
    <select name="quality">
      <option value="Hot">Hot</option>
      <option value="Warm">Warm</option>
      <option value="Cold">Cold</option>
    </select>
    <select name="disposition">
      <option value="New">New</option>
      <option value="Scheduled">Scheduled</option>
      {/* ... more */}
    </select>
  </section>

  {/* Measurements */}
  <section>
    <input name="sqFt" type="number" />
    <input name="ridgeLf" type="number" />
    {/* ... more */}
  </section>

  <button type="submit">Save Lead</button>
</form>
```

Integration with Backend

```
// Via googleSheetsService.js
export async function addLead(leadData) {
  const response = await fetch(GOOGLE_SHEETS_API_URL, {
    method: 'POST',
    mode: 'no-cors',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ action: 'addLead', data: leadData })
  });
  return response.json();
}
```

---

3. Job Count Tab

File Structure

```
src/features/jobcount/
├── JobCountView.jsx           (Main table)
├── JobCountFormModal.jsx      (Add/Edit form)
└── JobCountDetailModal.jsx    (Detail view)
```

Component Similarity to Leads

Job Count tab follows the same architecture as Leads tab with these differences:

Unique Fields:

```
const jobCountSpecificFields = [
  'sqFt',              // Square Footage
  'ridgeLf',           // Ridge Linear Feet
  'valleyLf',          // Valley Linear Feet
  'eavesLf',           // Eaves Linear Feet
  'ridgeVents',        // Ridge Vent count
  'turbine',           // Turbine count
  'rimeFlow',          // Rime Flow count
  'pipes1Half',        // 1.5" pipes
  'pipes2',            // 2" pipes
  'pipes3',            // 3" pipes
  'pipes4',            // 4" pipes
  'gables',            // Gable count
  'turtleBacks',       // Turtle Back count
  'chimney',           // Chimney count
  'solar',             // Solar panel count
  'guttersLf',         // Gutters Linear Feet
  'downspouts',        // Downspout count
  'permanentLighting'  // Permanent Lighting LF
];
```

Backend Integration - Automatic Lead Creation

When a job count is saved, it's automatically duplicated to Bhotchleads:

```
// Code.gs lines 419-460
function addJobCount(data) {
  try {
    // Add to Job Count sheet
    const jobCountSheet = getSheetById(JOB_COUNT_SHEET_ID);
    jobCountSheet.appendRow([/* ... job count data ... */]);

    // Automatically create lead
    const leadData = transformJobCountToLead(data);
    duplicateJobCountToLeads(leadData);

    return { success: true, message: 'Job count added and synced to leads' };
  } catch (error) {
    return { success: false, error: error.message };
  }
}

function duplicateJobCountToLeads(jobCountData) {
  const leadSheet = getSheetById(BHOTCHLEADS_SHEET_ID);

  // Check if lead already exists (by address or phone)
  const existingLead = findExistingLead(jobCountData);

  if (existingLead) {
    // Update existing lead with measurements
    updateLead(existingLead.id, jobCountData);
  } else {
    // Create new lead
    leadSheet.appendRow([
      jobCountData.date,
      jobCountData.customerName,
      jobCountData.firstName,
      jobCountData.lastName,
      jobCountData.phoneNumber,
      jobCountData.email,
      jobCountData.address,
      jobCountData.latitude,
      jobCountData.longitude,
      'Warm',  // Default quality
      'New',   // Default disposition
      'Job Count',  // Lead source
      jobCountData.sqFt,
      jobCountData.ridgeLf,
      jobCountData.valleyLf,
      // ... all measurements
    ]);
  }
}
```

---

4. Map Tab

File Structure

```
src/features/map/
└── MapView.jsx  (Main component)
```

Component Structure

```
function MapView({ leads })
```

State Management

```
const [map, setMap] = useState(null);
const [markers, setMarkers] = useState([]);
const [selectedLead, setSelectedLead] = useState(null);
const [filterQuality, setFilterQuality] = useState('all');
const [searchAddress, setSearchAddress] = useState('');
const [userLocation, setUserLocation] = useState(null);
```

Google Maps Integration

Initialize Map

```
useEffect(() => {
  if (!window.google) {
    console.error('Google Maps JavaScript API not loaded');
    return;
  }

  const mapInstance = new window.google.maps.Map(mapRef.current, {
    center: { lat: 39.7392, lng: -104.9903 }, // Denver, CO
    zoom: 11,
    mapTypeControl: true,
    streetViewControl: true,
    fullscreenControl: true,
    zoomControl: true
  });

  setMap(mapInstance);
}, []);
```

Add Markers

```
useEffect(() => {
  if (!map) return;

  // Clear existing markers
  markers.forEach(marker => marker.setMap(null));

  // Filter leads
  const filteredLeads = leads.filter(lead => {
    const qualityMatch = filterQuality === 'all' || lead.quality === filterQuality;
    const hasCoordinates = lead.latitude && lead.longitude;
    return qualityMatch && hasCoordinates;
  });

  // Create new markers
  const newMarkers = filteredLeads.map(lead => {
    const marker = new window.google.maps.Marker({
      position: { lat: parseFloat(lead.latitude), lng: parseFloat(lead.longitude) },
      map: map,
      title: lead.customerName || lead.address,
      icon: {
        url: getMarkerIcon(lead.quality),
        scaledSize: new window.google.maps.Size(32, 32)
      }
    });

    // Add click listener for info window
    marker.addListener('click', () => {
      setSelectedLead(lead);
      const infoWindow = new window.google.maps.InfoWindow({
        content: createInfoWindowContent(lead)
      });
      infoWindow.open(map, marker);
    });

    return marker;
  });

  setMarkers(newMarkers);
}, [map, leads, filterQuality]);
```

Marker Icons

```
function getMarkerIcon(quality) {
  const icons = {
    'Hot': 'http://maps.google.com/mapfiles/ms/icons/red-dot.png',
    'Warm': 'http://maps.google.com/mapfiles/ms/icons/orange-dot.png',
    'Cold': 'http://maps.google.com/mapfiles/ms/icons/blue-dot.png'
  };
  return icons[quality] || icons['Cold'];
}
```

Info Window Content

```
function createInfoWindowContent(lead) {
  return `
    <div style="max-width: 300px;">
      <h3 style="margin: 0 0 10px 0; font-size: 16px; font-weight: bold;">
        ${lead.customerName || 'Unknown'}
      </h3>
      <p style="margin: 5px 0;"><strong>Address:</strong> ${lead.address}</p>
      <p style="margin: 5px 0;"><strong>Phone:</strong> ${lead.phoneNumber || 'N/A'}</p>
      <p style="margin: 5px 0;"><strong>Quality:</strong>
        <span style="padding: 2px 8px; background: ${getQualityColor(lead.quality)}; border-radius: 4px; color: white;">
          ${lead.quality}
        </span>
      </p>
      <p style="margin: 5px 0;"><strong>Disposition:</strong> ${lead.disposition}</p>
      ${lead.dabellaQuote ? `<p style="margin: 5px 0;"><strong>Quote:</strong> $${formatNumber(lead.dabellaQuote)}</p>` : ''}
      <button onclick="window.open('https://www.google.com/maps/dir/?api=1&destination=${encodeURIComponent(lead.address)}',
'_blank')"
        style="margin-top: 10px; padding: 8px 12px; background: #4285F4; color: white; border: none; border-radius: 4px; cursor:
pointer;">
        Get Directions
      </button>
    </div>
  `;
}
```

Geocoding (Address to Coordinates)

```
function geocodeAddress(address, callback) {
  const geocoder = new window.google.maps.Geocoder();

  geocoder.geocode({ address }, (results, status) => {
    if (status === 'OK' && results[0]) {
      const location = results[0].geometry.location;
      callback({
        lat: location.lat(),
        lng: location.lng(),
        formattedAddress: results[0].formatted_address
      });
    } else {
      console.error('Geocoding failed:', status);
      callback(null);
    }
  });
}
```

User Location (Geolocation)

```
function getCurrentLocation() {
  if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(
      (position) => {
        const location = {
          lat: position.coords.latitude,
          lng: position.coords.longitude
        };
        setUserLocation(location);

        // Center map on user location
        map.setCenter(location);
        map.setZoom(13);

        // Add user marker
        new window.google.maps.Marker({
          position: location,
          map: map,
          title: 'Your Location',
          icon: {
            url: 'http://maps.google.com/mapfiles/ms/icons/blue-dot.png',
            scaledSize: new window.google.maps.Size(40, 40)
          }
        });
      },
      (error) => {
        console.error('Error getting location:', error);
      }
    );
  }
}
```

---

5. Calendar Tab

File Structure

```
src/features/calendar/
└── CalendarView.jsx  (Main component)
```

Component Structure

```
function CalendarView()
```

Configuration

```
const GOOGLE_CALENDAR_EMAIL = 'brandon@rimehq.net';

const getCalendarEmbedUrl = () => {
  const baseUrl = 'https://calendar.google.com/calendar/embed';
  const params = new URLSearchParams({
    src: GOOGLE_CALENDAR_EMAIL,
    ctz: 'America/Denver',  // Mountain Time Zone
    mode: calendarView,      // MONTH, WEEK, AGENDA
    showTitle: '0',
    showNav: '1',
    showDate: '1',
    showPrint: '0',
    showTabs: '1',
    showCalendars: '0',
    showTz: '0',
    bgcolor: '%23ffffff',
    color: '%232952A3'
  });
  return `${baseUrl}?${params.toString()}`;
};
```

State Management

```
const [calendarView, setCalendarView] = useState('MONTH'); // MONTH, WEEK, AGENDA
const [iframeError, setIframeError] = useState(false);
```

View Switching

```
<div className="flex items-center bg-gray-100 rounded-lg p-1">
  <button onClick={() => setCalendarView('MONTH')}>Month</button>
  <button onClick={() => setCalendarView('WEEK')}>Week</button>
  <button onClick={() => setCalendarView('AGENDA')}>Agenda</button>
</div>
```

Refresh Functionality

```
const handleRefresh = () => {
  const iframe = document.getElementById('google-calendar-iframe');
  if (iframe) {
    const currentSrc = iframe.src;
    iframe.src = '';
    iframe.src = currentSrc;
    setIframeError(false);
  }
};
```

Open in Google Calendar

```
const handleOpenInGoogle = () => {
  window.open(
    `https://calendar.google.com/calendar/u/0/r?cid=${GOOGLE_CALENDAR_EMAIL}`,
    '_blank'
  );
};
```

Iframe Embed

```
<iframe
  id="google-calendar-iframe"
  src={getCalendarEmbedUrl()}
  style={{ border: 0 }}
  width="100%"
  height="700"
  frameBorder="0"
  scrolling="no"
  title="Google Calendar"
  className="w-full"
></iframe>
```

---

6. Communications Tab

File Structure

```
src/features/communications/
└── CommunicationsView.jsx  (Main component)
```

Component API

```
function CommunicationsView({ leads, jobCounts, communications = [] })
```

Data Merging

```
const allCustomers = useMemo(() => {
  const leadCustomers = leads.map(lead => ({
    id: `lead-${lead.id}`,
    type: 'lead',
    name: lead.customerName || `${lead.firstName} ${lead.lastName}`.trim(),
    address: lead.address,
    phone: lead.phone,
    email: lead.email,
    quality: lead.quality,
    disposition: lead.disposition,
    lastContact: lead.lastContact,
    source: 'Bhotchleads'
  }));

  const jobCountCustomers = jobCounts.map(job => ({
    id: `job-${job.id}`,
    type: 'jobcount',
    name: job.customerName || `${job.firstName} ${job.lastName}`.trim(),
    address: job.address,
    phone: job.phone,
    email: job.email,
    sqFt: job.sqFt,
    source: 'Job Count'
  }));

  return [...leadCustomers, ...jobCountCustomers];
}, [leads, jobCounts]);
```

Google Voice Integration

Phone Calls

```
function handleCall(customer) {
  const phoneNumber = customer.phone.replace(/\D/g, '');
  const url = `https://voice.google.com/u/0/calls?a=nc,%2B1${phoneNumber}&authuser=brandon@rimehq.net`;
  window.open(url, '_blank');
}
```

SMS Messages

```
function handleSMS(customer) {
  const phoneNumber = customer.phone.replace(/\D/g, '');
  const url = `https://voice.google.com/u/0/messages?itemId=t.%2B1${phoneNumber}&authuser=brandon@rimehq.net`;
  window.open(url, '_blank');
}
```

SMS Outcome Tracking

```
const smsOutcomes = [
  { id: 'sent', label: 'Sent SMS', icon: SendHorizontal, color: 'green' },
  { id: 'received', label: 'Received SMS', icon: MessageCircleReply, color: 'teal' },
  { id: 'follow-up', label: 'Follow-up Needed', icon: CalendarClock, color: 'orange' },
  { id: 'not-interested', label: 'Not Interested', icon: UserX, color: 'red' }
];

function handleLogSMS(customerId, outcome, notes) {
  const communication = {
    customerId,
    type: 'sms',
    outcome,
    notes,
    timestamp: new Date().toISOString()
  };

  // Save to backend
  saveCommunication(communication);
}
```

Email Composition

```
function handleEmail(customer, subject, body) {
  const mailtoLink = `mailto:${customer.email}?subject=${encodeURIComponent(subject)}&body=${encodeURIComponent(body)}`;
  window.location.href = mailtoLink;
}
```

Communication History

```
const customerHistory = useMemo(() => {
  if (!selectedCustomer) return [];
  return communications
    .filter(c => c.customerId === selectedCustomer.id)
    .sort((a, b) => new Date(b.timestamp) - new Date(a.timestamp));
}, [selectedCustomer, communications]);
```

Communication Stats

```
const stats = useMemo(() => {
  const history = selectedCustomer ? customerHistory : communications;
  return {
    totalCalls: history.filter(c => c.type === 'call').length,
    totalSMS: history.filter(c => c.type === 'sms').length,
    totalEmails: history.filter(c => c.type === 'email').length,
    lastContact: history.length > 0 ? history[0].timestamp : null
  };
}, [selectedCustomer, customerHistory, communications]);
```

---

7. 360° Designer Tab

File Structure

```
src/features/visualization360/
├── DesignerView.jsx     (New modern designer)
└── Visualization360.jsx (Legacy 3D viewer)
```

Component Structure

```
function DesignerView()
```

State Management

```
const [selectedProperty, setSelectedProperty] = useState(null);
const [selectedLayer, setSelectedLayer] = useState('roof');
const [selectedColor, setSelectedColor] = useState('#334155');
const [selectedMaterial, setSelectedMaterial] = useState('asphalt-shingle');
const [showLeftPanel, setShowLeftPanel] = useState(true);
const [showRightPanel, setShowRightPanel] = useState(true);
const [zoom, setZoom] = useState(100);
const [showGrid, setShowGrid] = useState(true);
```

Layer Configuration

```
const layers = [
  { id: 'roof', name: 'Roof', icon: Home, visible: true, locked: false },
  { id: 'siding', name: 'Siding', icon: Layers, visible: true, locked: false },
  { id: 'trim', name: 'Trim', icon: Palette, visible: true, locked: false },
  { id: 'gutters', name: 'Gutters', icon: Grid, visible: true, locked: false }
];
```

Material Library

```
const materials = {
  roof: [
    { id: 'asphalt-shingle', name: 'Asphalt Shingle', color: '#334155' },
    { id: 'metal-standing-seam', name: 'Metal Standing Seam', color: '#64748b' },
    { id: 'tile', name: 'Clay Tile', color: '#dc2626' },
    { id: 'slate', name: 'Natural Slate', color: '#1e293b' }
  ],
  siding: [
    { id: 'vinyl', name: 'Vinyl Siding', color: '#f8fafc' },
    { id: 'fiber-cement', name: 'Fiber Cement', color: '#e2e8f0' },
    { id: 'wood', name: 'Wood Siding', color: '#92400e' }
  ],
  trim: [
    { id: 'white', name: 'White', color: '#ffffff' },
    { id: 'black', name: 'Black', color: '#000000' },
    { id: 'brown', name: 'Brown', color: '#78350f' }
  ]
};
```

Color Palette

```
const colorPalette = [
  { name: 'Charcoal', hex: '#334155' },
  { name: 'Slate Gray', hex: '#64748b' },
  { name: 'Weathered Wood', hex: '#92400e' },
  { name: 'Terracotta', hex: '#dc2626' },
  { name: 'Forest Green', hex: '#065f46' },
  { name: 'Colonial Blue', hex: '#1e40af' },
  { name: 'Tan', hex: '#d6bc8a' },
  { name: 'White', hex: '#ffffff' }
];
```

Canvas Rendering

```
<div
  ref={canvasRef}
  className="bg-white rounded-lg shadow-2xl relative"
  style={{
    width: `${zoom}%`,
    maxWidth: '1200px',
    aspectRatio: '16 / 9'
  }}
>
  {/* Grid overlay */}
  {showGrid && (
    <div
      className="absolute inset-0 pointer-events-none"
      style={{
        backgroundImage: 'linear-gradient(#e5e7eb 1px, transparent 1px), linear-gradient(90deg, #e5e7eb 1px, transparent 1px)',
        backgroundSize: '20px 20px'
      }}
    />
  )}

  {/* Property image placeholder */}
  {!selectedProperty ? (
    <div className="absolute inset-0 flex flex-col items-center justify-center">
      <Image className="w-20 h-20 mb-4 text-gray-400" />
      <p>Select or upload a property image to start designing</p>
    </div>
  ) : (
    <img src={selectedProperty.image} alt={selectedProperty.name} />
  )}
</div>
```

Zoom Controls

```
<div className="flex items-center gap-1 bg-gray-100 rounded-lg p-1">
  <button onClick={() => setZoom(Math.max(25, zoom - 25))}>
    <ZoomOut className="w-4 h-4" />
  </button>
  <span className="px-3 text-sm font-medium">{zoom}%</span>
  <button onClick={() => setZoom(Math.min(200, zoom + 25))}>
    <ZoomIn className="w-4 h-4" />
  </button>
</div>
```

---

8. Canvassing Tab

File Structure (Complex)

```
src/features/canvassing/
├── CanvassingView.jsx              (Legacy)
├── CanvassingViewEnhanced.jsx      (Current)
├── components/
│   ├── analytics/
│   │   └── CanvassingDashboard.jsx
│   ├── gamification/
│   │   └── Leaderboard.jsx
│   ├── map/
│   │   └── PropertyMarker.jsx
│   ├── property/
│   │   └── PropertyDetailSheet.jsx
│   ├── route/
│   │   └── RouteOptimizer.jsx
│   └── territory/
│       ├── TerritoryDrawingTool.jsx
│       └── TerritoryManager.jsx
├── hooks/
│   ├── useGeoLocation.js
│   └── useTerritories.js
├── services/
│   └── weatherService.js
├── store/
│   └── canvassingStore.js
└── utils/
    └── geoUtils.js
```

Zustand Store

```
// store/canvassingStore.js
import create from 'zustand';

const useCanvassingStore = create((set, get) => ({
  // State
  territories: [],
  routes: [],
  properties: [],
  currentLocation: null,
  selectedTerritory: null,

  // Actions
  addTerritory: (territory) => set(state => ({
    territories: [...state.territories, territory]
  })),

  deleteTerritory: (territoryId) => set(state => ({
    territories: state.territories.filter(t => t.id !== territoryId)
  })),

  updateTerritory: (territoryId, updates) => set(state => ({
    territories: state.territories.map(t =>
      t.id === territoryId ? { ...t, ...updates } : t
    )
  })),

  optimizeRoute: (waypoints) => {
    // Route optimization logic
    const optimized = calculateOptimalRoute(waypoints);
    set({ routes: [...get().routes, optimized] });
  },

  markPropertyVisited: (propertyId) => set(state => ({
    properties: state.properties.map(p =>
      p.id === propertyId ? { ...p, visited: true, visitedAt: new Date() } : p
    )
  }))
}));
```

useGeoLocation Hook

```
// hooks/useGeoLocation.js
import { useState, useEffect } from 'react';

export function useGeoLocation(options = {}) {
  const [location, setLocation] = useState(null);
  const [error, setError] = useState(null);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    if (!navigator.geolocation) {
      setError('Geolocation not supported');
      setLoading(false);
      return;
    }

    const watchId = navigator.geolocation.watchPosition(
      (position) => {
        setLocation({
          lat: position.coords.latitude,
          lng: position.coords.longitude,
          accuracy: position.coords.accuracy,
          timestamp: position.timestamp
        });
        setLoading(false);
      },
      (err) => {
        setError(err.message);
        setLoading(false);
      },
      {
        enableHighAccuracy: true,
        timeout: 5000,
        maximumAge: 0,
        ...options
      }
    );

    return () => navigator.geolocation.clearWatch(watchId);
  }, []);

  return { location, error, loading };
}
```

Territory Drawing

```jsx
// components/territory/TerritoryDrawingTool.jsx
function TerritoryDrawingTool({ map, onTerritoryCreated }) {
  const [isDrawing, setIsDrawing] = useState(false);
  const [polygon, setPolygon] = useState(null);

  const startDrawing = () => {
    const drawingManager = new google.maps.drawing.DrawingManager({
      drawingMode: google.maps.drawing.OverlayType.POLYGON,
      drawingControl: false,
      polygonOptions: {
        fillColor: '#FF0000',
        fillOpacity: 0.3,
        strokeWeight: 2,
        strokeColor: '#FF0000',
        editable: true,
        draggable: true
      }
    });

    drawingManager.setMap(map);

    google.maps.event.addListener(drawingManager, 'polygoncomplete', (polygon) => {
      const path = polygon.getPath();
      const coordinates = [];

      for (let i = 0; i < path.getLength(); i++) {
        const point = path.getAt(i);
        coordinates.push({ lat: point.lat(), lng: point.lng() });
      }

      onTerritoryCreated({
        id: Date.now().toString(),
        name: `Territory ${Date.now()}`,
        coordinates,
        area: google.maps.geometry.spherical.computeArea(path),
        createdAt: new Date()
      });

      setIsDrawing(false);
      drawingManager.setMap(null);
    });

    setIsDrawing(true);
  };

  return (
    <button onClick={startDrawing} disabled={isDrawing}>
      {isDrawing ? 'Drawing...' : 'Draw Territory'}
    </button>
  );
}
```

Route Optimization

```jsx
// components/route/RouteOptimizer.jsx
async function optimizeRoute(waypoints) {
  const directionsService = new google.maps.DirectionsService();

  const origin = waypoints[0];
  const destination = waypoints[waypoints.length - 1];
  const waypointsForService = waypoints.slice(1, -1).map(wp => ({
    location: new google.maps.LatLng(wp.lat, wp.lng),
    stopover: true
  }));

  return new Promise((resolve, reject) => {
    directionsService.route({
      origin: new google.maps.LatLng(origin.lat, origin.lng),
      destination: new google.maps.LatLng(destination.lat, destination.lng),
      waypoints: waypointsForService,
      optimizeWaypoints: true,
      travelMode: google.maps.TravelMode.DRIVING
    }, (result, status) => {
      if (status === 'OK') {
        resolve({
          route: result.routes[0],
          distance: result.routes[0].legs.reduce((sum, leg) => sum + leg.distance.value, 0),
          duration: result.routes[0].legs.reduce((sum, leg) => sum + leg.duration.value, 0),
          optimizedOrder: result.routes[0].waypoint_order
        });
      } else {
        reject(new Error(`Route optimization failed: ${status}`));
      }
    });
  });
}
```

Weather Service

```
// services/weatherService.js
const WEATHER_API_KEY = process.env.REACT_APP_WEATHER_API_KEY;

export async function getCurrentWeather(lat, lng) {
  try {
    const response = await fetch(
      `https://api.openweathermap.org/data/2.5/weather?lat=${lat}&lon=${lng}&appid=${WEATHER_API_KEY}&units=imperial`
    );
    const data = await response.json();

    return {
      temperature: Math.round(data.main.temp),
      conditions: data.weather[0].main,
      description: data.weather[0].description,
      humidity: data.main.humidity,
      windSpeed: Math.round(data.wind.speed),
      icon: data.weather[0].icon,
      isSuitableForCanvassing: isSuitableWeather(data)
    };
  } catch (error) {
    console.error('Weather fetch error:', error);
    return null;
  }
}

function isSuitableWeather(weatherData) {
  const temp = weatherData.main.temp;
  const conditions = weatherData.weather[0].main.toLowerCase();

  // Too hot/cold
  if (temp < 30 || temp > 95) return false;

  // Rain/snow
  if (conditions.includes('rain') || conditions.includes('snow') || conditions.includes('storm')) {
    return false;
  }

  return true;
}
```

Map Fix (Critical)

```
// CanvassingView.jsx - Fixed initialization
useEffect(() => {
  // Use requestAnimationFrame to ensure DOM is fully rendered
  requestAnimationFrame(() => {
    requestAnimationFrame(() => {
      const timer = setTimeout(() => {
        if (mapRef.current) {
          initializeMap();
        } else {
          console.warn('[Canvassing] Map ref not ready, retrying...');
          const retryTimer = setTimeout(() => {
            if (mapRef.current) {
              initializeMap();
            } else {
              setError('Map failed to initialize');
              setLoading(false);
            }
          }, 500);
        }
      }, 100);
    });
  });
}, []);
```

---

9. Shared Components

StatCard Component

```jsx
// components/StatCard.jsx
export function StatCard({ title, value, icon: Icon, color = 'blue' }) {
  const colors = {
    blue: 'bg-blue-50 text-blue-600 border-blue-200',
    red: 'bg-red-50 text-red-600 border-red-200',
    green: 'bg-green-50 text-green-600 border-green-200',
    purple: 'bg-purple-50 text-purple-600 border-purple-200'
  };

  return (
    <div className={`rounded-lg p-6 border-2 ${colors[color]} hover:shadow-lg transition-shadow`}>
      <div className="flex items-center justify-between mb-2">
        <Icon className="w-8 h-8" />
      </div>
      <p className="text-sm font-medium text-gray-600">{title}</p>
      <p className="text-3xl font-bold text-gray-900 mt-2">{value}</p>
    </div>
  );
}
```

ErrorBoundary Component

```jsx
// components/ErrorBoundary.jsx
import React from 'react';

export class ErrorBoundary extends React.Component {
  constructor(props) {
    super(props);
    this.state = { hasError: false, error: null };
  }

  static getDerivedStateFromError(error) {
    return { hasError: true, error };
  }

  componentDidCatch(error, errorInfo) {
    console.error('Error caught by boundary:', error, errorInfo);
  }

  render() {
    if (this.state.hasError) {
      return (
        <div className="min-h-screen flex items-center justify-center bg-gray-50">
          <div className="max-w-md p-8 bg-white rounded-lg shadow-lg">
            <h2 className="text-2xl font-bold text-red-600 mb-4">Something went wrong</h2>
            <p className="text-gray-600 mb-4">{this.state.error?.message}</p>
            <button
              onClick={() => window.location.reload()}
              className="px-4 py-2 bg-blue-600 text-white rounded hover:bg-blue-700"
            >
              Reload Application
            </button>
          </div>
        </div>
      );
    }

    return this.props.children;
  }
}
```

---

10. Custom Hooks

useLeads Hook

```
// hooks/useLeads.js
import { useState, useEffect, useCallback } from 'react';
import * as sheetsService from '../api/googleSheetsService';

export function useLeads() {
  const [leads, setLeads] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  const fetchLeads = useCallback(async () => {
    try {
      setLoading(true);
      const data = await sheetsService.getLeads();
      setLeads(data);
      setError(null);
    } catch (err) {
      setError(err.message);
    } finally {
      setLoading(false);
    }
  }, []);

  useEffect(() => {
    fetchLeads();
  }, [fetchLeads]);

  const addLead = async (leadData) => {
    try {
      await sheetsService.addLead(leadData);
      await fetchLeads();
    } catch (err) {
      throw new Error(`Failed to add lead: ${err.message}`);
    }
  };

  const updateLead = async (leadId, updates) => {
    try {
      await sheetsService.updateLead(leadId, updates);
      await fetchLeads();
    } catch (err) {
      throw new Error(`Failed to update lead: ${err.message}`);
    }
  };

  const deleteLead = async (leadId) => {
    try {
      await sheetsService.deleteLead(leadId);
      await fetchLeads();
    } catch (err) {
      throw new Error(`Failed to delete lead: ${err.message}`);
    }
  };

  return {
    leads,
    loading,
    error,
    refreshLeads: fetchLeads,
    addLead,
    updateLead,
    deleteLead
  };
}
```

11. API Services

Google Sheets Service

```
// api/googleSheetsService.js
const API_URL = process.env.REACT_APP_GOOGLE_SHEETS_API_URL;

export async function getLeads() {
  const response = await fetch(`${API_URL}?action=getLeads`);
  const data = await response.json();
  return data.success ? data.data : [];
}

export async function addLead(leadData) {
  const response = await fetch(API_URL, {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ action: 'addLead', data: leadData })
  });
  return response.json();
}

export async function updateLead(leadId, updates) {
  const response = await fetch(API_URL, {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ action: 'updateLead', leadId, data: updates })
  });
  return response.json();
}

export async function deleteLead(leadId) {
  const response = await fetch(API_URL, {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ action: 'deleteLead', leadId })
  });
  return response.json();
}
```

End of Developer Guide

This guide provides complete coding details for every tab and component in the Bhotch CRM system. Each section includes actual code examples, API references, and implementation patterns used in production.