Bhotch CRM - Complete System Handbook

Table of Contents

System Overview

Purpose

Bhotch CRM is a comprehensive customer relationship management system designed specifically for roofing sales businesses. It provides tools for lead management, job counting, property visualization, canvassing, scheduling, and communications.

Key Features

- **Lead Management**: Track and manage sales leads with filtering, sorting, and detailed views
- **Job Count Tracking**: Record property measurements and generate quotes
- **Interactive Map**: Visualize leads geographically with Google Maps integration
- **Calendar Integration**: Sync with Google Calendar for appointment management
- **Communications Hub**: Unified interface for calls, SMS, and email via Google Voice
- **360° Property Designer**: Professional property visualization and design tool
- **Canvassing Tools**: Territory management and route optimization
- **Dashboard Analytics**: Real-time statistics and performance metrics

Technology Stack

- **Frontend**: React 18.2.0
- **Styling**: Tailwind CSS 3.4.17
- **Icons**: Lucide React 0.469.0
- **Maps**: Google Maps JavaScript API
- **State Management**: Zustand
- **Backend**: Google Apps Script (Code.gs)
- **Database**: Google Sheets (Bhotchleads, Job Count)
- **Hosting**: Vercel
- **Version Control**: GitHub

System Requirements

Development Environment

- **Node.js**: v14.x or higher (v18.x recommended)
- **npm**: v6.x or higher
- **Operating System**: Windows 10/11, macOS 10.15+, or Linux
- **RAM**: Minimum 4GB, 8GB recommended
- **Disk Space**: Minimum 500MB for dependencies

Browser Requirements

- **Chrome**: Version 90+ (recommended)
- **Firefox**: Version 88+
- **Safari**: Version 14+
- **Edge**: Version 90+

Required Accounts & API Keys

1. **Google Cloud Platform** (for Maps API)
   - Maps JavaScript API
   - Geocoding API
2. **Google Workspace** (brandon@rimehq.net)
   - Google Sheets API access
   - Google Calendar API access
   - Google Voice integration
3. **Vercel** Account for hosting
4. **GitHub** Account for version control

Setup & Installation

Initial Setup

```
# 1. Clone the repository
git clone https://github.com/Bhotch/bhotch-crm.git
cd bhotch-crm

# 2. Install dependencies
npm install

# 3. Create environment file
cp .env.example .env

# 4. Configure environment variables (see below)
# Edit .env with your API keys

# 5. Start development server
npm start

# 6. Build for production
npm run build
```

Environment Configuration

Create a `.env` file in the root directory:

```
# Google Maps API
REACT_APP_GOOGLE_MAPS_API_KEY=your_google_maps_api_key_here

# Google Sheets Configuration
REACT_APP_GOOGLE_SHEETS_API_URL=https://script.google.com/macros/s/YOUR_SCRIPT_ID/exec

# Google Calendar
REACT_APP_GOOGLE_CALENDAR_EMAIL=brandon@rimehq.net

# Firebase (if using authentication)
REACT_APP_FIREBASE_API_KEY=your_firebase_api_key
REACT_APP_FIREBASE_AUTH_DOMAIN=your_project.firebaseapp.com
REACT_APP_FIREBASE_PROJECT_ID=your_project_id
```

Google Apps Script Backend Setup

1. **Open Google Apps Script**:
    - Go to https://script.google.com
    - Create new project named "Bhotch CRM Backend"
2. **Copy Code.gs**:
    - Copy contents of `Code.gs` from repository
    - Paste into Code.gs file in Apps Script editor
3. **Configure Spreadsheets**:
    - Update `BHOTCHLEADS_SHEET_ID` with your Bhotchleads sheet ID
    - Update `JOB_COUNT_SHEET_ID` with your Job Count sheet ID
4. **Deploy as Web App**:
    - Click "Deploy" > "New deployment"
    - Type: Web app
    - Execute as: Your account
    - Who has access: Anyone
    - Copy deployment URL
5. **Update Frontend**:
    - Add deployment URL to `.env` as `REACT_APP_GOOGLE_SHEETS_API_URL`

File Structure

```
bhotch-crm/
├── public/                         # Static files
│   ├── index.html                  # HTML template
│   ├── manifest.json               # PWA manifest
│   └── sw.js                       # Service worker
│
├── src/                            # Source code
│   ├── api/                        # API services
│   │   └── googleSheetsService.js  # Google Sheets integration
│   │
│   ├── components/                 # Shared components
│   │   ├── CommunicationIndicators.jsx
│   │   ├── ConfigErrorDisplay.jsx
│   │   ├── ConnectionStatus.jsx
│   │   ├── ErrorBoundary.jsx
│   │   └── StatCard.jsx
│   │
│   ├── features/                   # Feature modules (tabs)
│   │   ├── auth/                   # Authentication
│   │   │   └── LoginForm.jsx
│   │   │
│   │   ├── dashboard/               # Dashboard Tab
│   │   │   └── DashboardView.jsx
│   │   │
│   │   ├── leads/                   # Leads Tab
│   │   │   ├── LeadsView.jsx
│   │   │   ├── LeadFormModal.jsx
│   │   │   ├── LeadDetailModal.jsx
│   │   │   ├── CommunicationModal.jsx
│   │   │   └── HouseVisualization.jsx
│   │   │
│   │   ├── jobcount/                # Job Count Tab
│   │   │   ├── JobCountView.jsx
│   │   │   ├── JobCountFormModal.jsx
│   │   │   └── JobCountDetailModal.jsx
│   │   │
│   │   ├── map/                     # Map Tab
│   │   │   └── MapView.jsx
│   │   │
│   │   ├── calendar/                # Calendar Tab
│   │   │   └── CalendarView.jsx
│   │   │
│   │   ├── communications/          # Communications Tab
│   │   │   └── CommunicationsView.jsx
│   │   │
│   │   ├── visualization360/        # 360° Designer Tab
│   │   │   ├── DesignerView.jsx     # Main designer interface
│   │   │   ├── Visualization360.jsx  # Legacy 3D viewer
│   │   │   ├── components/          # Designer components
│   │   │   ├── services/            # AI & processing services
│   │   │   ├── store/               # State management
│   │   │   └── utils/               # Utility functions
│   │   │
│   │   └── canvassing/              # Canvassing Tab
│   │       ├── CanvassingView.jsx
│   │       ├── CanvassingViewEnhanced.jsx
│   │       ├── components/          # Canvassing components
│   │       ├── hooks/               # Custom hooks
│   │       ├── services/            # Services (weather, etc.)
│   │       ├── store/               # State management
│   │       └── utils/               # Utility functions
│   │
│   ├── hooks/                       # Custom React hooks
│   │   ├── useLeads.js
│   │   ├── useJobCounts.js
│   │   ├── useCommunications.js
│   │   └── useNotifications.js
│   │
│   ├── services/                    # Application services
│   │   ├── communicationsService.js
│   │   ├── firebase.js
│   │   └── googleMapsService.js
│   │
│   ├── App.jsx                      # Main application component
│   ├── index.js                     # Application entry point
│   ├── index.css                    # Global styles (Tailwind)
│   └── setupTests.js                # Test configuration
│
├── Code.gs                          # Google Apps Script backend
├── .env                             # Environment variables (not in git)
├── .env.example                     # Environment template
├── .gitignore                       # Git ignore rules
├── package.json                     # Dependencies and scripts
├── tailwind.config.js               # Tailwind configuration
├── README.md                        # Project readme
├── CLAUDE.md                        # Development instructions
└── SYSTEM_HANDBOOK.md               # This file
```

Architecture Overview

Frontend Architecture

```
┌──────────────────────────────────────────────┐
│           App.jsx (Main Router)              │
│  - Authentication                            │
│  - Tab Navigation                            │
│  - Global State                              │
└──────────────────────────────────────────────┘
                    │
        ┌───────────┴───────────┐
        │                       │
┌───────▼──────┐        ┌───────▼──────┐
│  UI Layer    │        │  Data Layer  │
│  - Dashboard │        │  - Hooks     │
│  - Leads     │        │  - Services  │
│  - JobCount  │        │  - API       │
│  - Map       │        │  - State     │
│  - Calendar  │        └──────────────┘
│  - Comms     │
│  - 360°      │
│  - Canvassing│
└──────────────┘
```

Data Flow

```
User Action
    ↓
Component Event Handler
    ↓
Custom Hook (useLeads, useJobCounts, etc.)
    ↓
API Service (googleSheetsService.js)
    ↓
HTTP Request → Google Apps Script Backend
    ↓
Google Sheets Database
    ↓
Response ← Google Apps Script
    ↓
State Update (React hooks)
    ↓
Component Re-render
    ↓
UI Update
```

State Management Strategy

- **Local State**: Component-specific data (useState)
- **Shared State**: Cross-component data (lifted state in App.jsx)
- **Cached State**: API responses (custom hooks with caching)
- **Zustand Stores**: Complex feature state (Canvassing, Visualization)

Tab-by-Tab Documentation

1. Dashboard Tab

File: `src/features/dashboard/DashboardView.jsx`

Purpose: Central overview of business metrics and quick actions

Key Features:

- Primary statistics (Total Leads, Hot Leads, Quoted Leads, Quote Value)
- Secondary statistics (Scheduled, Follow Up, Insurance, Conversion Rate)
- Job Count metrics (Total Counts, Square Feet, Closed Deals)
- Quick Actions panel
- Recent Leads and Job Counts
- Sales Pipeline overview

Props:

```
{
  stats: {
    totalLeads: Number,
    hotLeads: Number,
    quotedLeads: Number,
    totalQuoteValue: Number,
    totalJobCounts: Number,
    totalSqFt: Number
  },
  leads: Array<Lead>,
  jobCounts: Array<JobCount>,
  onNavigateToTab: Function(tabName)
}
```

State: None (stateless presentation component)

Data Flow:

1. Receives aggregated stats from parent (App.jsx)
2. Displays formatted metrics
3. Triggers navigation via callback

Styling: Tailwind CSS with gradient cards and responsive grid

---

2. Leads Tab

File: `src/features/leads/LeadsView.jsx`

Purpose: Comprehensive lead management and tracking

Key Features:

- Advanced filtering and search
- Sortable columns
- Column visibility management (saved to localStorage)
- Pagination (10/25/50/100 per page)
- Inline editing via modals
- Phone/email quick actions
- Quality and disposition badges

Components:

- `LeadsView.jsx` - Main table view
- `LeadFormModal.jsx` - Add/Edit form
- `LeadDetailModal.jsx` - Detailed view
- `CommunicationModal.jsx` - Communication logging

State Management:

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

Available Columns (40+ fields):

- Basic: ID, Date, Customer Name, Phone, Email, Address
- Lead Info: Quality, Disposition, Lead Source, Status
- Roof Info: Roof Age, Roof Type
- Measurements: SQ FT, Ridge LF, Valley LF, Eaves LF
- Financial: DaBella Quote
- Components: Pipes, Vents, Gutters, etc.

API Integration:

```
// Via useLeads hook
const { leads, loading, error, refreshLeads, addLead, updateLead, deleteLead } = useLeads();

// Backend: Code.gs
doGet(e) - Fetch leads
doPost(e) - Create/Update/Delete leads
```

Column Management:

- Click "Manage Columns" to show/hide columns
- Preferences saved to localStorage as `leadsVisibleColumns`
- Restored on page load

Filtering:

- Global search across all fields
- Date filter
- Per-column filters (text/number)
- Clear all filters button

---

3. Job Count Tab

File: `src/features/jobcount/JobCountView.jsx`

Purpose: Track property measurements and generate quotes

Key Features:

- Similar to Leads tab (filtering, sorting, pagination)
- Measurement tracking (SQ FT, Ridge LF, Valley LF, etc.)
- Quote calculation
- Automatic sync to Bhotchleads sheet

Components:

- `JobCountView.jsx` - Main table
- `JobCountFormModal.jsx` - Add/Edit form
- `JobCountDetailModal.jsx` - Detailed view

Unique Fields:

- Square Footage (SQ FT)
- Ridge Linear Feet
- Valley Linear Feet
- Eaves Linear Feet
- Ventilation counts (Ridge Vents, Turbine, Rime Flow)
- Pipe counts (1.5", 2", 3", 4")
- Features (Gables, Turtle Backs, Satellite, Chimney, Solar, Swamp Cooler)
- Gutters (LF, Downspouts, Gutter Guard LF)
- Permanent Lighting

Backend Sync:

```
// Code.gs lines 419-460
// When job count is created, it's automatically duplicated to Bhotchleads
function duplicateJobCountToLeads(jobCountData) {
  // Transforms job count into lead format
  // Adds to Bhotchleads sheet
  // Returns success/failure
}
```

---

4. Map Tab

File: `src/features/map/MapView.jsx`

Purpose: Geographic visualization of leads and routing

Key Features:

- Google Maps integration
- Lead markers with clustering
- Info windows with lead details
- Filter by quality/disposition
- Search by address
- Directions to property

API Integration:

```
// Google Maps JavaScript API
const map = new google.maps.Map(element, options);
const marker = new google.maps.Marker({ position, map, title });
const infoWindow = new google.maps.InfoWindow({ content });

// Geocoding for address → coordinates
const geocoder = new google.maps.Geocoder();
geocoder.geocode({ address }, callback);
```

Marker Colors:

- 🔴 Red: Hot leads
- 🟠 Orange: Warm leads
- 🔵 Blue: Cold leads

Map Controls:

- Zoom in/out
- Fullscreen toggle
- Street view
- Map/Satellite view

---

5. Calendar Tab

File: `src/features/calendar/CalendarView.jsx`

Purpose: Appointment scheduling and management via Google Calendar

Key Features:

- Embedded Google Calendar (brandon@rimehq.net)
- Month/Week/Agenda views
- Timezone: America/Denver (Mountain Time)
- Refresh functionality
- Open in Google Calendar button

Configuration:

```
const GOOGLE_CALENDAR_EMAIL = 'brandon@rimehq.net';
const getCalendarEmbedUrl = () => {
  return `https://calendar.google.com/calendar/embed?src=${email}&ctz=America/Denver&mode=MONTH`;
};
```

Integration:

- Read-only embed (appointments managed in Google Calendar)
- Events automatically sync and display
- Click "Open in Google" to add/edit appointments

---

6. Communications Tab

File: `src/features/communications/CommunicationsView.jsx`

Purpose: Unified communication hub for calls, SMS, and email

Key Features:

- Customer search and filtering
- Google Voice integration for calls and SMS
- Email composition (opens default mail client)
- Communication history logging
- Quick outcome selection for SMS

Google Voice Integration:

```
// Phone calls
window.open(`https://voice.google.com/u/0/calls?a=nc,%2B1${phoneNumber}&authuser=brandon@rimehq.net`);

// SMS messages
window.open(`https://voice.google.com/u/0/messages?itemId=t.%2B1${phoneNumber}&authuser=brandon@rimehq.net`);
```

Communication Types:

1. **Phone Calls**:
   - Click "📞 Call" button
   - Opens Google Voice in new tab
   - Logs call outcome
2. **SMS Messages**:
   - Select quick outcome (Sent, Received, Follow-up, Not Interested)
   - Enter message content
   - Log to system
   - Optionally open Google Voice
3. **Email**:
   - Compose subject and body
   - Click to open default email client
   - Pre-filled with recipient and content

Communication History:

- Stored in `communications` array
- Filtered by customer
- Displays type, outcome, timestamp, notes
- Color-coded by outcome

---

7. 360° Property Designer Tab

File: `src/features/visualization360/DesignerView.jsx`

Purpose: Professional property visualization and design tool

Key Features:

- Layer-based editing (Roof, Siding, Trim, Gutters)
- Material selection library
- Color palette with custom picker
- Zoom controls (25%-200%)
- Grid overlay
- Photo upload placeholder
- Export and share functionality

Layout:

```
┌─────────────────────────────────────────────────────┐
│  Top Toolbar                                        │
│  [Property] [Zoom] [Grid] [Upload] [Export] [Share]  │
├─────────┬───────────────────────────┬───────────────┤
│  Layers │      Canvas               │  Materials    │
│  - Roof │   [Property Image]        │  Color        │
│  - Siding│  [Zoom 100%]             │  Palette      │
│  - Trim │   [Grid overlay]          │  Custom       │
│  - Gutters                          │               │
└─────────┴───────────────────────────┴───────────────┘
```

Layer Management:

```
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:

- Asphalt Shingle, Metal Standing Seam, Tile, Slate (Roof)

- Vinyl, Fiber Cement, Wood (Siding)
- White, Black, Brown (Trim)

Color Palette:

- 8 preset colors (Charcoal, Slate Gray, Weathered Wood, etc.)
- Custom hex color picker
- Real-time preview

State:

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

Benefits over Legacy Viewer:

- 641KB smaller bundle size
- Faster load times
- More intuitive UI
- Better for sales presentations
- Easier to use for non-technical staff

---

8. Canvassing Tab

File: `src/features/canvassing/CanvassingViewEnhanced.jsx`

Purpose: Door-to-door sales territory management and route optimization

Key Features:

- Territory drawing and management
- Route optimization
- Property tracking
- Weather integration
- Leaderboard and gamification
- Analytics dashboard

Components:

- `CanvassingViewEnhanced.jsx` - Main view
- `TerritoryManager.jsx` - Territory management
- `RouteOptimizer.jsx` - Route planning
- `PropertyDetailSheet.jsx` - Property information
- `CanvassingDashboard.jsx` - Analytics
- `Leaderboard.jsx` - Gamification

State Management (Zustand):

```
// store/canvassingStore.js
const useCanvassingStore = create((set) => ({
  territories: [],
  routes: [],
  properties: [],
  currentLocation: null,
  addTerritory: (territory) => { /* ... */ },
  optimizeRoute: (waypoints) => { /* ... */ },
  markPropertyVisited: (propertyId) => { /* ... */ }
}));
```

Weather Integration:

```
// services/weatherService.js
export async function getCurrentWeather(lat, lng) {
  // Fetches weather data for canvassing decisions
  // Returns temperature, conditions, precipitation
}
```

Territory Drawing:

- Click to draw polygon on map
- Save territory boundaries
- Assign territories to users
- Track coverage

Bug Fix (Completed):

- Fixed map container initialization issue
- Added `requestAnimationFrame` for DOM rendering
- Improved retry logic
- Now loads reliably

Backend Integration

Google Apps Script (Code.gs)

Purpose: Serverless backend for data operations

Key Functions:

1. **doGet(e)** - Handle GET requests

```
function doGet(e) {
  const action = e.parameter.action;
  if (action === 'getLeads') return getLeads();
  if (action === 'getJobCounts') return getJobCounts();
  // ... etc
}
```

2. **doPost(e)** - Handle POST requests

```
function doPost(e) {
  const data = JSON.parse(e.postData.contents);
  const action = data.action;
  if (action === 'addLead') return addLead(data);
  if (action === 'updateLead') return updateLead(data);
  if (action === 'deleteLead') return deleteLead(data);
  // ... etc
}
```

3. **duplicateJobCountToLeads(jobCountData)** - Lines 419-460

```
// When job count is created, automatically create lead
function duplicateJobCountToLeads(jobCountData) {
  const leadSheet = getSheetById(BHOTCHLEADS_SHEET_ID);
  // Transform job count to lead format
  // Add to Bhotchleads sheet
  // Return success
}
```

Data Structure:

Bhotchleads Sheet Columns:

```
A: Date
B: Customer Name
C: First Name
D: Last Name
E: Phone Number
F: Email
G: Address
H: Latitude
I: Longitude
J: Quality (Hot/Warm/Cold)
K: Disposition (New/Scheduled/Insurance/Quoted/Follow Up/Closed Sold/Closed Lost)
L: Lead Source
M-Z: Measurements and features
AA-AZ: Additional fields
```

Job Count Sheet Columns:

```
Similar structure to Bhotchleads
Plus additional measurement fields
SQ FT, Ridge LF, Valley LF, etc.
```

Error Handling:

```
try {
  // Operation
  return ContentService.createTextOutput(JSON.stringify({ success: true, data }));
} catch (error) {
  Logger.log('Error: ' + error);
  return ContentService.createTextOutput(JSON.stringify({ success: false, error: error.message }));
}
```

Deployment Guide

Vercel Deployment

Prerequisites:

- Vercel account
- GitHub repository connected

Steps:

1. **Push to GitHub**:

```
git add .
git commit -m "Deploy to production"
git push origin main
```

2. **Automatic Deployment**:

- Vercel auto-deploys from GitHub pushes

- Build command: `npm run build`
- Output directory: `build`
- Node version: 18.x

3. **Manual Deployment**:

```
vercel --prod
```

4. **Environment Variables**:

- Add environment variables in Vercel dashboard
- Settings → Environment Variables
- Add all variables from `.env`

Deployment URLs:

- Production: https://bhotch-crm.vercel.app (or custom domain)
- Preview: Unique URL per deployment

Build Settings:

```
{
  "buildCommand": "npm run build",
  "outputDirectory": "build",
  "installCommand": "npm install",
  "framework": "create-react-app"
}
```

Performance Optimization

Current Bundle Size:

- Main JS: 225.02 KB (gzipped)
- CSS: 9.49 KB (gzipped)
- Total: ~234 KB

Optimizations Applied:

- Code splitting
- Lazy loading
- Tree shaking
- Minification
- Image optimization
- Service worker caching

Troubleshooting

Common Issues

1. Map Not Loading

```
Issue: Google Maps doesn't render
Solution:
- Check REACT_APP_GOOGLE_MAPS_API_KEY in .env
- Verify API is enabled in Google Cloud Console
- Check browser console for API errors
```

2. Canvassing Map Container Error

```
Issue: "Map container element not found"
Solution:
- Already fixed in latest version
- Uses requestAnimationFrame for DOM readiness
- If persists, refresh page
```

3. Google Sheets API Failing

```
Issue: Data not loading from sheets
Solution:
- Verify Google Apps Script deployment URL
- Check CORS settings in Apps Script
- Ensure "Execute as: Me" in deployment settings
- Verify sheet IDs are correct
```

4. Calendar Not Displaying

```
Issue: 401 Unauthorized or blank calendar
Solution:
- Calendar is private - this is normal
- Click "Open in Google" to manage directly
- Check GOOGLE_CALENDAR_EMAIL matches account
```

5. Build Errors

```
Issue: npm run build fails
Solution:
- Clear cache: rm -rf node_modules package-lock.json
- Reinstall: npm install
- Check Node version: node --version (should be 14+)
```

6. Communications Not Working

```
Issue: Google Voice links not opening
Solution:
- Verify logged into correct Google account (brandon@rimehq.net)
- Check phone number format is valid
- Pop-up blocker may be blocking - allow pop-ups
```

Debug Mode

Enable verbose logging:

```
// In App.jsx
const DEBUG = process.env.NODE_ENV === 'development';

if (DEBUG) {
  console.log('[DEBUG] Leads loaded:', leads);
  console.log('[DEBUG] API response:', response);
}
```

Check Service Worker:

```
// In browser console
navigator.serviceWorker.getRegistrations().then(registrations => {
  console.log('Service Workers:', registrations);
});
```

Maintenance & Updates

Regular Maintenance Tasks

Weekly:

- Check Vercel deployment status
- Review error logs
- Backup Google Sheets data

Monthly:

- Update npm dependencies: `npm update`
- Review and optimize bundle size
- Check for security vulnerabilities: `npm audit`

Quarterly:

- Major dependency updates
- Performance audit
- User feedback review

Version Control

Branching Strategy:

```
main          - Production-ready code
develop       - Development branch
feature/*     - New features
bugfix/*      - Bug fixes
hotfix/*      - Emergency fixes
```

Commit Message Format:

```
feat: Add new feature
fix: Bug fix
docs: Documentation update
style: Formatting changes
refactor: Code restructuring
test: Add tests
chore: Maintenance tasks
```

Support & Contact

Developer: Brandon Hotchkiss
Email: brandon@rimehq.net
GitHub: https://github.com/Bhotch/bhotch-crm
Deployment: https://bhotch-crm.vercel.app

Last Updated: 2025-10-02
Version: 2.0.0
Status: Production Ready ✅