PM Request Contract - Zoho CRM EmbeddedApp

A React TypeScript application for managing contract products in Zoho CRM. This EmbeddedApp allows users to select products from deal records and tag them as contract items with single-click functionality.

🎯 Purpose

This application implements a production-ready Zoho CRM EmbeddedApp that:

- Fetches products from Deal record's `Subform_1` field
- Allows users to select products to tag as contract items (single selection only)
- Updates the `Is_Contract` field for selected products using Zoho CRM API
- Provides real-time feedback and error handling
- Automatically triggers Zoho workflows upon successful updates

🚀 Tech Stack

- **React 19.1** with TypeScript
- **Tailwind CSS** for modern UI styling
- **Vite** for fast development and building
- **Axios** for HTTP requests
- **Zoho EmbeddedApp SDK** for CRM integration
- **FormData API** for immediate form submission

📋 Features

- ✅ **Deal Product Management** - Displays all products from deal subforms with detailed information
- ✅ **Single Product Selection** - Radio button interface for selecting one product at a time
- ✅ **Real-time API Updates** - Immediate Zoho CRM record updates via API
- ✅ **Loading States** - Visual feedback during API operations
- ✅ **Error Handling** - Comprehensive error messages and recovery
- ✅ **Success Indicators** - Visual confirmation of successful updates
- ✅ **TypeScript Safety** - Full type coverage for Zoho SDK and data structures
- ✅ **FormData Integration** - Modern React 19.1 form handling
- ✅ **Workflow Triggers** - Automatic Zoho workflow execution on updates
- ✅ **Responsive Design** - Mobile-friendly interface

🏗️ Project Structure

```
src/
├── components/
│   ├── ContractProduct.tsx    # Main contract product selection component
│   └── ZohoPageLoad.tsx       # Zoho SDK event handler
├── types/
│   └── zoho.ts               # TypeScript interfaces for Zoho data
├── utils/
│   └── zohoApi.ts            # Zoho CRM API integration functions
├── App.tsx                   # Application entry point
└── main.tsx                  # React initialization with Zoho SDK
index.html                    # Includes Zoho EmbeddedApp SDK script
```

🛠️ Installation & Setup

Prerequisites

- Node.js 18+
- Access to Zoho CRM Developer Console
- Zoho CRM organization with Deal records containing product subforms
- Understanding of Zoho EmbeddedApp development

Step 1: Clone and Install Dependencies

```
# Navigate to project directory
cd pm_request_contract
# Install dependencies
npm install
```

Step 2: Development Server

```
# Start the development server
npm run dev
```

Open http://localhost:5173 to view the application.

Step 3: Build for Production

```
# Build the application
npm run build

# Preview the production build
npm run preview
```

Step 4: Zoho CRM Setup

1. **Create EmbeddedApp in Zoho CRM:**
    - Go to Setup → Developer Hub → EmbeddedApps
    - Create new EmbeddedApp with your app details
    - Set the hosting URL to your development/production URL
2. **Configure App Settings:**
    - **App Type:** EmbeddedApp
    - **Hosting:** External (your domain)
    - **Display:** Deal records (where the product subforms exist)
    - **Permissions:** CRM scope with read/write access
3. **Install in Zoho CRM:**
    - Install the EmbeddedApp in your Zoho CRM instance
    - Navigate to a Deal record to trigger the widget

🎮 Usage

How It Works

1. **Widget Loads** - When you open a Deal record in Zoho CRM, the widget automatically loads
2. **Data Capture** - The widget captures the deal data including all products from `Subform_1`
3. **Product Display** - All products are displayed with their details (name, type, quantity, pricing, etc.)
4. **Contract Selection** - Users can select a product using radio buttons to tag as contract item
5. **API Update** - Selection immediately triggers Zoho CRM API to update the `Is_Contract` field
6. **Visual Feedback** - Loading states, success indicators, and error messages provide user feedback
7. **Workflow Trigger** - Successful updates automatically trigger configured Zoho workflows

User Interface

The widget displays:

- **Deal Information**: Deal name, account, and stage
- **Product List**: All products from the deal's subform with:
  - Product name and type
  - Quantity and terms
  - Unit price and total pricing
  - Current contract status
  - Main product indicators
- **Selection Interface**: Radio buttons for single product selection
- **Status Indicators**: Loading, success, and error states
- **Summary**: Total products count and combined value

Product Selection Process

1. **View Products** - All products from the deal are displayed
2. **Select Product** - Click radio button next to desired product
3. **Immediate Update** - Selection triggers immediate API call to Zoho CRM
4. **Status Update** - Product is marked as "Contract Item" with green badge
5. **Workflow Execution** - Any configured workflows are automatically triggered

🏗️ Implementation Details

Core Architecture

The application follows a clean, modular architecture:

1. Main App Component (`src/App.tsx`)

- Receives Zoho PageLoad data
- Manages overall application state
- Renders the ContractProduct component

2. Contract Product Component (`src/components/ContractProduct.tsx`)

- Displays product list with selection interface
- Handles FormData for immediate form submission
- Manages loading states and error handling
- Calls Zoho API functions for updates

3. Zoho API Utils (`src/utils/zohoApi.ts`)

- Async/await functions for Zoho CRM API calls
- Proper error handling and logging
- Notification system integration

4. TypeScript Types (`src/types/zoho.ts`)

- Complete type definitions for Zoho SDK
- Deal and product data structures
- API response interfaces

Key Features Implementation

FormData Integration (React 19.1):

```
const handleProductSelection = async (event: React.ChangeEvent<HTMLInputElement>) => {
  const formData = new FormData(event.currentTarget.form!)
  const selectedProductIndex = formData.get('selectedProduct')
  const dealId = formData.get('dealId')
  // Immediate API call triggered
}
```

Single Product Selection:

- Radio buttons instead of checkboxes
- Prevents multiple selections
- Auto-submits on selection change
- Disabled state for already-contracted products

Real-time API Updates:

```
const response = await updateProductContractStatus(
  dealId as string,
  productIndex,
  selectedProduct,
  products
)
```

Error Handling:

- Comprehensive try-catch blocks
- User-friendly error messages
- Loading state management
- Network error recovery **Error Handling:**
- Comprehensive try-catch blocks
- User-friendly error messages
- Loading state management
- Network error recovery

Zoho SDK Integration

Initialization Pattern:

```
// src/main.tsx
useEffect(() => {
  const initializeZoho = () => {
    if (window.ZOHO?.embeddedApp) {
      window.ZOHO.embeddedApp.on("PageLoad", function(data) {
        console.log('PageLoad event received:', data)
        setZohoData(data)
        setIsInitialized(true)
      })

      window.ZOHO.embeddedApp.init()
      console.log('Zoho EmbeddedApp initialized')
    }
  }

  if (window.ZOHO) {
    initializeZoho()
  } else {
    const checkZoho = setInterval(() => {
      if (window.ZOHO) {
        clearInterval(checkZoho)
        initializeZoho()
      }
    }, 100)
  }
}, [])
```

API Update Structure:

```
// Correct Zoho API payload structure
const apiConfig = {
  Entity: "Deals",
  APIData: {
    id: dealId,
    Subform_1: [updatedProduct] // Only the updated product record
  },
  Trigger: ["workflow"] // Automatically trigger workflows
}
```

Data Flow Architecture

1. **PageLoad Event** → Deal data with product subforms captured
2. **Component Render** → Products displayed in user interface
3. **User Selection** → Radio button triggers FormData capture
4. **API Call** → Zoho CRM updateRecord API called
5. **Response Handling** → Success/error feedback displayed
6. **Workflow Trigger** → Automatic Zoho workflow execution

TypeScript Integration

Complete Type Safety:

```
// All Zoho data structures are fully typed
interface ZohoProductSubform {
  Is_Contract: boolean
  Main_Product: boolean
  Products: ZohoProduct
  Product_Type2: string
  Quantity: number
  Terms: string
  Pricing: number
  Total_Pricing: string
  Vendor: string | null
}

// SDK integration with proper typing
declare global {
  interface Window {
    ZOHO: {
      embeddedApp: {
        on: (event: string, callback: (data: ZohoPageLoadData) => void) => void
        init: () => void
      }
      CRM: {
        API: {
          updateRecord: (options: ZohoUpdateOptions) => Promise<ZohoUpdateResponse>
        }
      }
    }
  }
}
```

🔧 API Documentation

Core API Functions

`updateProductContractStatus()`

Updates a product's contract status in Zoho CRM.

Parameters:

- `dealId: string` - The Zoho Deal record ID
- `productIndex: number` - Index of the product in the subform array
- `productData: ZohoProductSubform` - Complete product record data
- `allProducts: ZohoProductSubform[]` - All products in the subform

Returns: `Promise<ZohoUpdateResponse>`

Example:

```
const response = await updateProductContractStatus(
  "123456789",
  0,
```

```
  selectedProduct,
  allProducts
)
```

`showNotification()`

Displays notifications and manages popup behavior.

Parameters:

- `message: string` - Notification message
- `type: 'success' | 'error' | 'info'` - Notification type

Example:

```
showNotification(
  "Successfully tagged product as contract item",
  "success"
)
```

API Response Handling

Success Response:

```
{
  "data": [{
    "code": "SUCCESS",
    "details": {
      "Modified_Time": "2025-06-04T10:30:00+00:00",
      "Modified_By": {
        "name": "User Name",
        "id": "123456"
      },
      "id": "deal_record_id"
    },
    "message": "record updated",
    "status": "success"
  }]
}
```

Error Handling:

- Network errors are caught and displayed to users
- API errors include detailed error messages
- Loading states prevent multiple concurrent requests
- User feedback includes both success and failure scenarios

🔧 Build and Deploy

Build for Production

```
# Build for production
npm run build

# Preview the build
npm run preview
```

The build artifacts will be in the `dist` folder.

Deployment Options

1. Static Hosting (Vercel, Netlify, GitHub Pages)

```
# Build the project
npm run build

# Deploy the dist folder to your hosting provider
```

2. Custom Server

```
# Build and serve
npm run build
npx serve dist
```

Zoho CRM Configuration

Zoho CRM Configuration

After deployment, configure your EmbeddedApp in Zoho CRM:

1. **Update App Settings:**
    - Set the hosting URL to your production domain
    - Configure proper CORS settings
    - Set up SSL/HTTPS for security
2. **Test Integration:**
    - Navigate to a Deal record with product subforms
    - Verify the widget loads and displays products
    - Test product selection and contract status updates
    - Confirm workflow triggers are working
3. **Production Deployment:**
    - Publish the EmbeddedApp to all users
    - Monitor error logs and user feedback
    - Set up analytics if needed

🐛 Troubleshooting

Common Issues and Solutions

1. Widget Not Loading

Problem: EmbeddedApp doesn't appear in Zoho CRM **Solutions:**

- Check hosting URL is accessible via HTTPS
- Verify EmbeddedApp is installed and published
- Ensure proper permissions are set in Zoho
- Check browser console for errors

2. PageLoad Event Not Firing

Problem: No data received from Zoho **Solutions:**

```
// Add SDK availability check
useEffect(() => {
  const checkZoho = () => {
    if (window.ZOHO?.embeddedApp) {
      console.log('Zoho SDK available')
      window.ZOHO.embeddedApp.on("PageLoad", handlePageLoad)
      window.ZOHO.embeddedApp.init()
    } else {
      console.log('Zoho SDK not yet available, retrying...')
      setTimeout(checkZoho, 100)
    }
  }
  checkZoho()
}, [])
```

3. API Update Failures

Problem: Contract status not updating in Zoho CRM **Solutions:**

- Verify API permissions include CRM.modules.deals.UPDATE
- Check deal ID and subform structure
- Ensure `Is_Contract` field exists in Zoho CRM
- Review API payload structure matches Zoho requirements

4. TypeScript Errors

Problem: Type errors with Zoho SDK **Solutions:**

- Use `@ts-expect-error` for Zoho global objects
- Ensure all interfaces match actual Zoho data structure
- Update type definitions if Zoho changes API structure

Debug Mode

Enable comprehensive logging for troubleshooting:

```
// Add to src/utils/zohoApi.ts
const DEBUG_MODE = true

if (DEBUG_MODE) {
  console.log('=== DEBUG MODE ENABLED ===')
  console.log('API Config:', apiConfig)
```

```
  console.log('Product Data:', productData)
  console.log('==========================')
}
```

Performance Optimization

Bundle Size Optimization:

```
# Analyze bundle size
npm run build
npx vite-bundle-analyzer dist
```

API Call Optimization:

- Implement request debouncing for rapid selections
- Cache deal data to reduce API calls
- Use loading states to prevent multiple concurrent requests

🔐 Security Considerations

Data Protection

- All API calls use Zoho's secure authentication
- No sensitive data stored in client-side state
- HTTPS required for production deployment

Access Control

- Proper Zoho CRM permissions required
- Widget only accessible to authorized users
- API calls respect Zoho's security model

Best Practices

- Input validation for all user interactions
- Error messages don't expose sensitive information
- Proper logging without including personal data

📝 Development Notes

Code Standards

- **TypeScript**: Full type coverage required
- **ESLint**: Follow configured linting rules
- **Formatting**: Use Prettier for consistent code style
- **Comments**: Document complex business logic

Testing Strategy

- **Manual Testing**: Test all user interactions
- **Integration Testing**: Verify Zoho API integration
- **Error Testing**: Test error scenarios and recovery
- **Cross-browser**: Test in different browsers

Version Control

- **Git Workflow**: Feature branches with pull requests
- **Commit Messages**: Clear, descriptive commit messages
- **Tagging**: Tag releases for production deployments

🚀 Future Enhancements

Potential Features

1. **Bulk Product Selection** - Select multiple products at once
2. **Contract Templates** - Pre-defined contract product categories
3. **Approval Workflow** - Require approval for contract selections
4. **Reporting Dashboard** - Analytics for contract product usage
5. **Email Notifications** - Notify stakeholders of contract updates

Technical Improvements

1. **Unit Testing** - Add comprehensive test coverage
2. **Caching** - Implement client-side caching for better performance
3. **Offline Support** - Handle offline scenarios gracefully
4. **Real-time Updates** - WebSocket integration for live updates

📚 Resources

Zoho Documentation

- Zoho CRM EmbeddedApp Guide
- Zoho CRM API Reference
- EmbeddedApp Best Practices

React & TypeScript

- React 19 Documentation
- TypeScript Handbook
- Vite Guide

Development Tools

- Tailwind CSS
- ESLint Configuration
- VS Code Extensions

📄 License

🤝 Contributing

1. Fork the repository
2. Create a feature branch (`git checkout -b feature/amazing-feature`)
3. Commit your changes (`git commit -m 'Add some amazing feature'`)
4. Push to the branch (`git push origin feature/amazing-feature`)
5. Open a Pull Request

📞 Support

For support and questions:

- Review the troubleshooting section above
- Check Zoho CRM documentation
- Review the codebase for implementation examples
- Create an issue for bugs or feature requests

Built with ❤️ for Zoho CRM integration

- Navigate to records in Zoho CRM
- Verify PageLoad events are being captured
- Check browser console for any errors

🐛 Troubleshooting

Common Issues

1. ZOHO SDK Not Loading

```
Error: Cannot read property 'embeddedApp' of undefined
```

Solution: Ensure the Zoho SDK script is loaded before React app initialization.

2. PageLoad Event Not Firing

Solution:

- Verify the app is properly installed in Zoho CRM
- Check that you're navigating to records (not just viewing lists)
- Ensure the app is displayed in the correct module

3. CORS Issues

Solution:

- Ensure your domain is whitelisted in Zoho CRM app settings
- Use HTTPS in production
- Configure proper CORS headers if using custom server

Debug Mode

Enable detailed logging by opening browser console:

```
// The app automatically logs PageLoad data to console
// Look for: console.log(data) output
```

📚 Resources

Zoho Documentation

- EmbeddedApp Documentation
- SDK Reference
- Best Practices

Development Tools

- Zoho Developer Console

- CRM Setup Guide

🤝 Contributing

1. Fork the repository
2. Create a feature branch (`git checkout -b feature/your-feature`)
3. Commit your changes (`git commit -m 'Add some feature'`)
4. Push to the branch (`git push origin feature/your-feature`)
5. Open a Pull Request

📄 License

This project is licensed under the MIT License - see the LICENSE file for details.

🙋 Support

For questions or issues:

1. Check the Troubleshooting section
2. Review Zoho Documentation
3. Open an issue in this repository

Built with ❤️ for Zoho CRM integration

)}

````
### Using Zoho Hooks

```tsx
import { useZohoSDK } from '@/hooks/core/useZohoSDK'
import { useZohoContacts } from '@/hooks/data/useZohoContacts'
import { useZohoActions } from '@/hooks/actions/useZohoActions'

function MyComponent() {
  const { isInitialized, error } = useZohoSDK()
  const contacts = useZohoContacts()
  const { createContact } = useZohoActions()

  // Your component logic
}
````

Error Handling

```
import { ErrorBoundary } from '@/components/common/ErrorBoundary'

function App() {
  return (
    <ErrorBoundary>
      <ZohoProvider>
        <YourComponents />
      </ZohoProvider>
```
```

```
    </ErrorBoundary>
  )
}
```

🎨 UI Components

This project uses shadcn/ui components with Tailwind CSS v4. Available components:

- **Badge** - Status indicators and labels
- **Card** - Container components with header, content, footer
- **Button** - Various button styles and variants

Adding New shadcn/ui Components

```
npx shadcn@latest add [component-name]
```

Example:

```
npx shadcn@latest add dialog table form
```

🧪 Testing in Zoho CRM

Local Development Testing

1. **Use ngrok for HTTPS tunnel:**

   ```
   # Install ngrok globally
   npm install -g ngrok

   # Start your dev server
   npm run dev

   # In another terminal, create HTTPS tunnel
   ngrok http 5173
   ```

2. **Update EmbeddedApp URL:**
   - Copy the ngrok HTTPS URL
   - Update your EmbeddedApp configuration in Zoho CRM
   - Test the integration

Production Deployment

1. **Build the application:**

   ```
   npm run build
   ```

2. **Deploy to your hosting provider**

3. **Update EmbeddedApp URL in Zoho CRM**

4. **Test in production environment**

🔍 Key Implementation Details

Zoho SDK Initialization Sequence

The correct initialization sequence is crucial for EmbeddedApp functionality:

1. **Subscribe to PageLoad events** before initialization
2. **Initialize the SDK** with proper error handling
3. **Handle initialization results** and update state accordingly

```
// src/hooks/core/useZohoSDK.ts - Key implementation
useEffect(() => {
  const initializeSDK = async () => {
    try {
      // 1. Subscribe to PageLoad FIRST
      await embeddedAppService.subscribeToPageLoad()

      // 2. Then initialize
      const result = await embeddedAppService.initialize()

      // 3. Handle results
      setInitialized(result.success)
    } catch (error) {
      setError(error as Error)
    }
  }

  initializeSDK()
}, [])
```

Service Layer Pattern

The application uses a clean service layer architecture:

- **Services** - Business logic and Zoho SDK interactions
- **Repositories** - Data access and caching
- **Hooks** - React state management and side effects
- **Components** - UI presentation layer

React 19.1 Features

The application leverages React 19.1 features:

- **useActionState** - For form submissions and mutations
- **use() hook** - Planned for data fetching (when stable)
- **Enhanced Suspense** - For loading states
- **Automatic batching** - Improved performance

🐛 Troubleshooting

Common Issues

1. **SDK Not Initializing:**
   - Check HTTPS requirement in production
   - Verify EmbeddedApp configuration in Zoho CRM
   - Ensure proper domain whitelisting
2. **PageLoad Events Not Firing:**
   - Verify event subscription happens before initialization
   - Check browser console for errors
   - Ensure proper cleanup on unmount
3. **Type Errors:**
   - All Zoho SDK types are defined in `src/types/zoho.ts`
   - Update types if using newer Zoho SDK versions
4. **Build Issues:**
   - Ensure all path aliases are properly configured
   - Check TypeScript configuration in `tsconfig.json`

Debug Mode

Enable debug logging by setting:

```
// In your service initialization
const embeddedAppService = new ZohoEmbeddedAppService({
  debug: true,
  // other options
})
```

📚 Documentation Links

- Zoho CRM EmbeddedApp Documentation
- React 19.1 Documentation
- Tailwind CSS v4 Documentation
- shadcn/ui Documentation
- Vite Documentation

🤝 Contributing

1. Fork the repository
2. Create a feature branch
3. Follow the existing architecture patterns
4. Add comprehensive TypeScript types
5. Test thoroughly in Zoho CRM environment
6. Submit a pull request

📄 License

🆘 Support

For issues related to:

- **Zoho CRM Integration** - Check Zoho Developer Documentation
- **React/TypeScript** - Refer to official React documentation
- **shadcn/ui Components** - Visit shadcn/ui documentation
- **Project-specific Issues** - Create an issue in this repository

---

Built with ❤️ using React 19.1, Tailwind CSS v4, and modern development practices. }, })

```
# CPOA-Product-Contract-Utility

# CPOA-Product-Contract-Utility

# CPOA-Product-Contract-Utility
```