**AdminApp - Comprehensive Project Documentation**

**Table of Contents**

1. Project Overview
2. Architecture & Technology Stack
3. Project Structure
4. Core Features
5. Authentication & Authorization
6. State Management
7. Services & APIs
8. Component Library
9. Database Schema
10. Routing
11. Styling & UI
12. Installation & Setup
13. Development Guidelines
14. API Integration
15. Testing
16. Deployment
17. Troubleshooting

**Project Overview**

**Description**

AdminApp is a comprehensive Customer Relationship Management (CRM) system built with React.js. It provides businesses with tools for managing customers, leads, invoicing, social media scheduling, and various administrative tasks.

**Key Capabilities**

* **Customer Management**: Lead tracking, customer segmentation, and relationship management
* **Invoicing System**: Invoice creation, templates, and automated billing
* **Social Media Management**: Post scheduling across multiple platforms
* **Order Management**: Shopify integration for order processing
* **Template Management**: Email and invoice template creation and management
* **Dashboard Analytics**: Real-time business insights and reporting
* **User Management**: Role-based access control and permissions

**Architecture & Technology Stack**

**Frontend Technologies**

* **React.js** (v18+) - Main framework
* **React Router** - Client-side routing
* **Redux Toolkit** - State management
* **Ant Design** - UI component library
* **Bootstrap** - CSS framework
* **Axios** - HTTP client
* **React Quill** - Rich text editor
* **Chart.js** - Data visualization

**Backend Services**

* **Firebase** - Primary database and authentication
  + Firestore - NoSQL database
  + Firebase Auth - Authentication
  + Firebase Storage - File storage
* **External APIs** - CRM API integration

**Build Tools**

* **Create React App** - Build configuration
* **npm** - Package management

**Key Libraries**

* **jsPDF** - PDF generation
* **html2canvas** - HTML to canvas conversion
* **XLSX** - Excel file handling
* **react-toastify** - Notifications

**Project Structure**

src/

├── App.js                    # Main application component

├── index.js                  # Application entry point

├── App.css                   # Global styles

├── index.css                 # Base styles

│

├── components/               # Reusable UI components

│   ├── MainLayout.js        # Main application layout

│   ├── ProtectedRoute.jsx   # Route protection wrapper

│   ├── CustomInput.js       # Custom form inputs

│   ├── CustomModal.js       # Modal components

│   ├── PostForm.jsx         # Social media post form

│   ├── DashboardCards/      # Dashboard widgets

│   ├── DashboardNavbar/     # Navigation components

│   └── EmailTemplateEditor/ # Template editing tools

│

├── pages/                   # Page components

│   ├── Dashboard.js         # Main dashboard

│   ├── Login.js            # Authentication

│   ├── Signup.js           # User registration

│   ├── Templates/          # Template management

│   ├── TemplateEditor/     # Template creation/editing

│   ├── Invoicing/          # Invoice management

│   ├── ShopifyManagement/  # Order management

│   ├── SchedulePosts/      # Social media scheduling

│   ├── UserManagement/     # User administration

│   └── CustomerSegments/   # Customer segmentation

│

├── Services/               # API and business logic

│   ├── api.js             # Main API configuration

│   ├── axiosConfig.js     # HTTP client setup

│   ├── firebaseService.js # Firebase operations

│   ├── invoiceService.js  # Invoice management

│   ├── automationService.js # Business automation

│   ├── pdfService.js      # PDF generation

│   └── templateService.js # Template operations

│

├── features/              # Redux slices

│   ├── invoice/          # Invoice state management

│   ├── blogs/            # Blog management

│   └── coupon/           # Coupon system

│

├── utils/                # Utility functions

│   ├── permissions.js    # Role-based access control

│   ├── localstorage.js  # Local storage helpers

│   ├── invoiceHelpers.js # Invoice calculations

│   └── excelExport.js   # Excel export functionality

│

├── firebase/             # Firebase configuration

│   └── Config.js        # Firebase setup

│

└── assets/              # Static assets

    └── images/          # Image files

**Core Features**

**1. Dashboard Analytics**

* **Location**: [Dashboard.js](vscode-file://vscode-app/c:/Users/PMYLS/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)
* **Components**:
  + [OpportunityValueDashboard](vscode-file://vscode-app/c:/Users/PMYLS/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)
  + [LeadSourceReport](vscode-file://vscode-app/c:/Users/PMYLS/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)
  + [GoogleAnalyticsReport](vscode-file://vscode-app/c:/Users/PMYLS/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)

**Key Metrics**:

* Revenue tracking and forecasting
* Lead conversion rates
* Customer acquisition analytics
* Social media performance

**2. Invoice Management**

* **Service**: [invoiceService](vscode-file://vscode-app/c:/Users/PMYLS/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ")
* **Helpers**: [invoiceHelpers](vscode-file://vscode-app/c:/Users/PMYLS/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ")
* **State**: [invoiceSlice](vscode-file://vscode-app/c:/Users/PMYLS/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ")

**Features**:

* Invoice creation and editing
* Template-based invoice generation
* PDF export functionality
* Email delivery
* Payment tracking
* Automated invoice generation from orders

**3. Template System**

* **Pages**: [Templates](vscode-file://vscode-app/c:/Users/PMYLS/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html), [TemplateEditor](vscode-file://vscode-app/c:/Users/PMYLS/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ")
* **API**: [emailTemplateApi](vscode-file://vscode-app/c:/Users/PMYLS/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ")

**Capabilities**:

* Drag-and-drop template editor
* Template categorization with folders
* Email and invoice templates
* Template preview functionality
* Dynamic variable substitution

**4. Social Media Management**

* **Components**: [PostForm](vscode-file://vscode-app/c:/Users/PMYLS/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o "), [ReviewPost](vscode-file://vscode-app/c:/Users/PMYLS/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ")
* **Pages**: SchedulePosts

**Features**:

* Multi-platform posting (Facebook, Instagram, LinkedIn, Twitter, Pinterest)
* Post scheduling and calendar view
* Media upload and management
* Account linking and management

**5. Customer & Lead Management**

* **Service**: [firebaseService](vscode-file://vscode-app/c:/Users/PMYLS/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ")
* **Utils**: [excelExport](vscode-file://vscode-app/c:/Users/PMYLS/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ")

**Features**:

* Lead capture and tracking
* Customer segmentation
* Bulk data import/export
* Customer communication history

**Authentication & Authorization**

**Authentication Flow**

**Implementation**: [Login.js](vscode-file://vscode-app/c:/Users/PMYLS/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)

*// Firebase Authentication*

const handleLogin = async (email, password) => {

  const userCredentials = await signInWithEmailAndPassword(auth, email, password);

  const user = userCredentials.user;

*// Fetch user data from Firestore*

  const userQuery = query(

    collection(db, "Users"),

    where("uid", "==", user.uid)

  );

*// Store user and permissions*

  await fetchAndStoreRolePermissions();

};

**Role-Based Access Control**

**Implementation**: [permissions.js](vscode-file://vscode-app/c:/Users/PMYLS/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)

export const fetchAndStoreRolePermissions = async () => {

  const rolesSnapshot = await getDocs(collection(db, "userRole"));

  const rolePermissions = {};

  roles.forEach((role) => {

    rolePermissions[role.roleName] = role.routes || [];

  });

  localStorage.setItem("rolePermissions", JSON.stringify(rolePermissions));

};

**Protected Routes**

**Component**: [ProtectedRoute](vscode-file://vscode-app/c:/Users/PMYLS/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)

Routes are protected based on user roles and permissions stored in Firestore.

**State Management**

**Redux Store Configuration**

**Location**: [store.js](vscode-file://vscode-app/c:/Users/PMYLS/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)

**Key Slices**

1. **Invoice Management**: [invoiceSlice](vscode-file://vscode-app/c:/Users/PMYLS/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ")
2. **Blog Management**: blogSlice
3. **Coupon System**: couponSlice

**State Structure**

{

  invoice: {

    invoices: [],

    currentInvoice: null,

    isLoading: false,

    isError: false,

    message: ""

  },

*// Other slices...*

}

**Services & APIs**

**Firebase Services**

**Location**: [firebaseService.js](vscode-file://vscode-app/c:/Users/PMYLS/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)

*// Lead Management*

export const fetchCustomLeads = async () => {

  const leadsCollection = collection(db, "customLeads");

  const snapshot = await getDocs(leadsCollection);

  return snapshot.docs.map(doc => ({ id: doc.id, ...doc.data() }));

};

*// Bulk Operations*

export const addBulkCustomLeads = async (leadsArray) => {

  const batch = [];

  const leadsCollection = collection(db, "customLeads");

  for (const lead of leadsArray) {

    const docRef = await addDoc(leadsCollection, {

      ...lead,

      createdAt: new Date(),

      source: lead.source || 'imported'

    });

    batch.push({ id: docRef.id, ...lead });

  }

  return batch;

};

**Invoice Services**

**Location**: [invoiceService.js](vscode-file://vscode-app/c:/Users/PMYLS/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)

Key operations:

* CRUD operations for invoices
* Order-to-invoice conversion
* PDF generation
* Email delivery
* Payment tracking

**PDF Generation**

**Location**: [pdfService.js](vscode-file://vscode-app/c:/Users/PMYLS/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)

export const pdfService = {

  generatePDFFromElement: async (element, options = {}) => {

    const canvas = await html2canvas(element, {

      scale: 2,

      useCORS: true,

      allowTaint: true,

      backgroundColor: '#ffffff'

    });

    const pdf = new jsPDF('portrait', 'mm', 'a4');

    const imgData = canvas.toDataURL('image/png');

*// PDF generation logic...*

  }

};

**Automation Services**

**Location**: [automationService.js](vscode-file://vscode-app/c:/Users/PMYLS/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)

Features:

* Business rule automation
* Order processing workflows
* Email automation
* Invoice generation triggers

**Component Library**

**Layout Components**

* **MainLayout**: Main application shell with sidebar navigation
* **DashboardNavbar**: Top navigation bar
* **ProtectedRoute**: Route access control wrapper

**Form Components**

* **CustomInput**: Standardized form inputs
* **CustomModal**: Reusable modal dialogs
* **PostForm**: Social media post creation form

**Dashboard Components**

* **OpportunityValueDashboard**: Revenue tracking widget
* **LeadSourceReport**: Lead analytics dashboard
* **GoogleAnalyticsReport**: Website analytics display

**Utility Components**

* **RouteLoader**: Loading states for route transitions
* **EmailTemplateEditor**: Rich text template editor

**Database Schema**

**Firebase Collections**

**Users Collection**

{

  uid: "firebase\_user\_id",

  userName: "John Doe",

  userEmail: "john@example.com",

  Role: "Admin", *// Admin, User, Manager*

  profileImageUrl: "https://...",

  createdAt: timestamp

}

**User Roles Collection**

{

  roleName: "Admin",

  routes: [

    "Dashboard",

    "Invoicing",

    "Templates",

    "Settings",

*// ... other permitted routes*

  ],

  permissions: {

    canCreate: true,

    canEdit: true,

    canDelete: true,

    canView: true

  }

}

**Invoices Collection**

{

  invoiceNumber: "INV-20241201-123456",

  customerId: "customer\_id",

  customerName: "Customer Name",

  customerEmail: "customer@example.com",

  items: [

    {

      name: "Product Name",

      quantity: 1,

      price: 100.00,

      total: 100.00

    }

  ],

  subtotal: 100.00,

  tax: 10.00,

  total: 110.00,

  status: "draft", *// draft, sent, paid, overdue*

  createdAt: timestamp,

  dueDate: timestamp

}

**Custom Leads Collection**

{

  firstName: "John",

  lastName: "Doe",

  email: "john@example.com",

  phone: "+1234567890",

  company: "Company Name",

  source: "website", *// website, social, referral, import*

  leadType: "hot", *// hot, warm, cold*

  createdAt: timestamp,

  notes: "Additional information"

}

**Automation Rules Collection**

{

  name: "Auto Invoice Generation",

  triggerType: "new\_order", *// new\_order, payment\_received, etc.*

  conditions: [

    {

      field: "order.total",

      operator: "greater\_than",

      value: 100

    }

  ],

  actions: [

    {

      type: "generate\_invoice",

      templateId: "template\_id"

    }

  ],

  isActive: true,

  createdAt: timestamp

}

**Routing**

**Route Structure**

**Location**: [App.js](vscode-file://vscode-app/c:/Users/PMYLS/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)

*// Main Routes*

<Route path="/admin" element={<MainLayout />}>

  <Route index element={<Dashboard />} />

  // Authentication Routes

  <Route path="profile" element={<Profile />} />

  // Business Management

  <Route path="Contacts" element={<Contacts />} />

  <Route path="CustomerSegments" element={<CustomerSegments />} />

  // Social Media

  <Route path="SchedulePosts/Create-Post" element={<CreatePosts />} />

  <Route path="SchedulePosts/Calendar" element={<Calendar />} />

  // Templates

  <Route path="Templates" element={<Templates />} />

  <Route path="template-editor" element={<TemplateEditor />} />

  <Route path="template-editor/:id" element={<TemplateEditor />} />

  // Invoicing

  <Route path="invoicing/\*" element={<Invoicing />} />

  // Shopify Integration

  <Route path="Shopify/OrderList" element={<OrderList />} />

  <Route path="Shopify/Delivered-Orders" element={<DeliveredOrders />} />

  // Administration

  <Route path="Settings" element={<Settings />} />

  <Route path="UserManagement" element={<UserManagement />} />

</Route>

**Route Protection**

All routes are protected using the [ProtectedRoute](vscode-file://vscode-app/c:/Users/PMYLS/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) component, which checks user permissions against the required route access level.

**Styling & UI**

**CSS Architecture**

* **Global Styles**: [index.css](vscode-file://vscode-app/c:/Users/PMYLS/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)
* **Component Styles**: [App.css](vscode-file://vscode-app/c:/Users/PMYLS/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)
* **Page-specific Styles**: Individual CSS files per page/component

**Design System**

* **Primary Framework**: Bootstrap 5
* **Component Library**: Ant Design
* **Icons**: Font Awesome
* **Fonts**: Roboto, Poppins

**Theme Configuration**

:root {

  --color-red:  #dc3545;

  --color-212529:  #212529;

  --color-c3d4da:  #c3d4da;

  --bg-main:  #1d1d1d;

}

**Responsive Design**

* Mobile-first approach
* Breakpoints: 450px, 850px, 1200px
* Flexible grid system using Bootstrap

**Installation & Setup**

**Prerequisites**

* Node.js (v14 or higher)
* npm or yarn
* Firebase account
* Git

**Local Development Setup**

1. **Clone Repository**

git clone <repository-url>

cd AdminApp

1. **Install Dependencies**

npm install

1. **Environment Configuration** Create [.env](vscode-file://vscode-app/c:/Users/PMYLS/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) file:

REACT\_APP\_API\_URL=https://crmapi.alayaarts.com/api

REACT\_APP\_BEARER\_TOKEN=your\_bearer\_token\_here

1. **Firebase Configuration** Update [Config.js](vscode-file://vscode-app/c:/Users/PMYLS/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) with your Firebase credentials:

const firebaseConfig = {

  apiKey: "your\_api\_key",

  authDomain: "your\_project.firebaseapp.com",

  projectId: "your\_project\_id",

  storageBucket: "your\_project.appspot.com",

  messagingSenderId: "your\_sender\_id",

  appId: "your\_app\_id"

};

1. **Start Development Server**

npm start

**Production Build**

npm run build

**Development Guidelines**

**Code Standards**

* **JavaScript**: ES6+ features, functional components with hooks
* **Naming Conventions**:
  + Components: PascalCase
  + Functions: camelCase
  + Constants: UPPER\_SNAKE\_CASE
* **File Organization**: Group related files in directories
* **Import Order**: External libraries, internal services, components, utils

**Component Structure**

*// Standard component template*

import React, { useState, useEffect } from 'react';

import { useDispatch, useSelector } from 'react-redux';

import './ComponentName.css';

const ComponentName = ({ prop1, prop2 }) => {

  const [localState, setLocalState] = useState('');

  const dispatch = useDispatch();

  const globalState = useSelector(state => state.slice);

  useEffect(() => {

*// Side effects*

  }, []);

  const handleAction = () => {

*// Event handlers*

  };

  return (

    <div className="component-container">

      {*/\* JSX content \*/*}

    </div>

  );

};

export default ComponentName;

**State Management Patterns**

* Use Redux for global state
* Use local state for component-specific data
* Use useEffect for side effects
* Use custom hooks for reusable logic

**Error Handling**

try {

  const result = await apiCall();

  toast.success('Operation successful');

} catch (error) {

  console.error('Error:', error);

  toast.error('Operation failed');

}

**API Integration**

**External API Configuration**

**Location**: [axiosConfig.js](vscode-file://vscode-app/c:/Users/PMYLS/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)

**API Endpoints**

* **Base URL**: https://crmapi.alayaarts.com/api
* **Authentication**: Bearer token
* **Content Type**: application/json

**Key API Services**

1. **Email Templates**: CRUD operations for email templates
2. **Folders**: Template organization
3. **Invoices**: Invoice management
4. **Automation**: Business rule automation

**Error Handling**

const apiRequest = async (method, endpoint, data = null) => {

  try {

    const response = await axios({

      method,

      url: `${API\_URL}${endpoint}`,

      headers: {

        'Authorization': `Bearer ${BEARER\_TOKEN}`,

        'Content-Type': 'application/json'

      },

      data

    });

    return response.data;

  } catch (error) {

    console.error(`API Error (${method} ${endpoint}):`, error);

    throw new Error(error.response?.data?.message || 'API request failed');

  }

};

**Testing**

**Testing Strategy**

* Unit tests for utility functions
* Integration tests for services
* Component testing with React Testing Library
* End-to-end testing for critical user flows

**Test File Structure**

src/

├── \_\_tests\_\_/           # Test files

├── utils/

│   ├── invoiceHelpers.js

│   └── \_\_tests\_\_/

│       └── invoiceHelpers.test.js

└── Services/

    ├── invoiceService.js

    └── \_\_tests\_\_/

        └── invoiceService.test.js

**Running Tests**

npm test                 *# Run all tests*

npm test -- --watch     *# Watch mode*

npm test -- --coverage  *# Coverage report*

**Deployment**

**Build Process**

npm run build

**Deployment Options**

1. **Firebase Hosting**: Integrated with Firebase backend
2. **Netlify**: Static site deployment
3. **Vercel**: React application hosting
4. **Traditional Web Server**: Serve static files

**Environment Variables**

Ensure all production environment variables are configured:

* API endpoints
* Firebase configuration
* Authentication tokens
* Feature flags

**Build Optimization**

* Code splitting with React.lazy()
* Image optimization
* Bundle analysis with webpack-bundle-analyzer
* Performance monitoring

**Troubleshooting**

**Common Issues**

**Authentication Problems**

*// Debug authentication state*

const debugAuth = () => {

  const user = getUserFromLocalStorage();

  console.log('Current user:', user);

  console.log('Role permissions:', localStorage.getItem('rolePermissions'));

};

**Firebase Connection Issues**

*// Debug Firebase connection*

import { debugFirebaseData } from './utils/debugFirebase';

*// Available in browser console*

window.debugFirebaseData();

**Route Access Issues**

Check role permissions and ensure user has appropriate access level for the route.

**Performance Issues**

* Monitor bundle size
* Optimize images
* Implement lazy loading
* Check for memory leaks in useEffect hooks

**Development Tools**

* **React Developer Tools**: Browser extension for debugging React
* **Redux DevTools**: State management debugging
* **Firebase Console**: Database and authentication management
* **Network Tab**: API request monitoring

**Logging and Monitoring**

*// Comprehensive error logging*

const logError = (error, context) => {

  console.error('Error occurred:', {

    error: error.message,

    stack: error.stack,

    context,

    timestamp: new Date().toISOString(),

    userAgent: navigator.userAgent

  });

};

**Contributing**

**Code Review Process**

1. Create feature branch from main
2. Implement changes following coding standards
3. Write/update tests
4. Submit pull request
5. Code review and approval
6. Merge to main branch

**Documentation Updates**

* Update this documentation for new features
* Add inline code comments for complex logic
* Update API documentation for new endpoints
* Maintain changelog for version releases

**Version Control**

* Use semantic versioning (MAJOR.MINOR.PATCH)
* Tag releases appropriately
* Maintain clean commit history
* Use conventional commit messages