# **CRM Module Documentation**

## **Table of Contents**

1. Overview
2. System Architecture
3. Installation & Setup
4. API Reference
5. Frontend Components
6. Database Schema
7. Security Guidelines
8. Testing
9. Deployment

## **Overview**

The CRM (Customer Relationship Management) module is a comprehensive solution designed for managing client relationships, streamlining onboarding processes, and enhancing customer communication. This module is part of a larger SaaS platform and focuses on improving customer engagement through automated workflows and data-driven insights.

### **Key Features**

* Client onboarding workflow management
* Automated communication system
* Analytics dashboard
* Task management
* Email integration
* Real-time activity tracking

## **System Architecture**

### **Tech Stack**

* Backend: Python/Flask
* Frontend: React.js
* Database: PostgreSQL
* Authentication: JWT
* Email Service: SMTP Integration
* Analytics: Custom implementation with data warehousing

### **Component Diagram**

**[Frontend (React)] ←→ [API Gateway]**

**↑ ↑**

**| |**

**[Analytics Dashboard] [Backend Services]**

**↑ ↑**

**| |**

**[Data Layer] ←→ [PostgreSQL]**

## **Installation & Setup**

### **Prerequisites**

* Python 3.9+
* Node.js 16+
* PostgreSQL 13+
* Docker (optional)

### **Local Development Setup**

1. Clone the repository:

**git clone https://github.com/mrigankraj/crm-module-saas.git**

**cd crm-module-saas**

1. Set up Python virtual environment:

**python -m venv venv**

**source venv/bin/activate *# Unix***

**.\venv\Scripts\activate *# Windows***

**pip install -r requirements.txt**

1. Configure environment variables:

**cp .env.example .env**

***# Edit .env with your configuration***

1. Set up the database:

**psql -U postgres -f db/schema.sql**

**psql -U postgres -f db/seed.sql**

1. Start the backend server:

**python src/backend/app.py**

1. Set up the frontend:

**cd frontend**

**npm install**

**npm start**

## **API Reference**

### **Authentication Endpoints**

## **POST /api/auth/login**

Authenticates a user and returns a JWT token.

Request:

**{**

**"email": "user@company.com",**

**"password": "securepassword123"**

**}**

Response:

**{**

**"token": "eyJhbGciOiJIUzI1NiIs...",**

**"user": {**

**"id": 1,**

**"email": "user@company.com",**

**"company\_name": "Acme Inc"**

**}**

**}**

### **Onboarding Endpoints**

## **POST /api/onboarding/register**

Registers a new company in the system.

Request:

**{**

**"company\_name": "Acme Inc",**

**"email": "user@company.com",**

**"password": "securepassword123",**

**"contact\_person": "John Doe",**

**"phone": "+1-555-0123",**

**"industry": "Technology"**

**}**

Response:

**{**

**"success": true,**

**"user\_id": 1,**

**"message": "Registration successful"**

**}**

## **GET /api/onboarding/progress**

Retrieves onboarding progress for the current user.

Response:

**{**

**"progress": 75,**

**"completed\_tasks": [**

**{**

**"id": 1,**

**"title": "Complete Profile",**

**"completed\_at": "2025-01-08T10:00:00Z"**

**}**

**],**

**"pending\_tasks": [**

**{**

**"id": 2,**

**"title": "Set Communication Preferences",**

**"due\_date": "2025-01-15T00:00:00Z"**

**}**

**]**

**}**

### **Communication Endpoints**

## **POST /api/communication/send**

Sends a communication to a client.

Request:

**{**

**"type": "email",**

**"recipient\_id": 1,**

**"subject": "Welcome to Our Platform",**

**"content": "Dear valued customer...",**

**"priority": "high",**

**"schedule\_time": null *// Optional, for scheduled communications***

**}**

Response:

**{**

**"message\_id": 123,**

**"status": "sent",**

**"timestamp": "2025-01-08T10:05:00Z"**

**}**

### **Analytics Endpoints**

## **GET /api/analytics/engagement**

Retrieves engagement metrics for clients.

Response:

**{**

**"overall\_engagement\_rate": 78.5,**

**"interaction\_metrics": {**

**"total\_interactions": 1250,**

**"average\_response\_time": "2h 15m",**

**"popular\_channels": [**

**{"channel": "email", "usage": 65},**

**{"channel": "chat", "usage": 25},**

**{"channel": "phone", "usage": 10}**

**]**

**},**

**"trend\_data": {**

**"daily\_interactions": [...],**

**"response\_rates": [...],**

**"satisfaction\_scores": [...]**

**}**

**}**

## **Frontend Components**

### **Core Components**

1. OnboardingProgress
   * Displays onboarding progress
   * Manages task completion
   * Shows next steps
2. CommunicationDashboard
   * Message composition
   * Communication history
   * Template management
   * Channel selection
3. AnalyticsDashboard
   * Engagement metrics
   * Interactive charts
   * Export functionality

### **Component Usage**

***// Example usage of OnboardingProgress***

**import { OnboardingProgress } from './components';**

**function OnboardingPage() {**

**return (**

**<div className="container">**

**<OnboardingProgress**

**userId={123}**

**onTaskComplete={handleTaskComplete}**

**/>**

**</div>**

**);**

**}**

## **Database Schema**

### **Users Table**

**CREATE TABLE users (**

**id SERIAL PRIMARY KEY,**

**company\_name VARCHAR(255) NOT NULL,**

**email VARCHAR(255) UNIQUE NOT NULL,**

**password\_hash VARCHAR(255) NOT NULL,**

**contact\_person VARCHAR(255),**

**phone VARCHAR(20),**

**industry VARCHAR(100),**

**created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,**

**onboarding\_status VARCHAR(50) DEFAULT 'pending'**

**);**

### **Interactions Table**

**CREATE TABLE interactions (**

**id SERIAL PRIMARY KEY,**

**user\_id INTEGER REFERENCES users(id),**

**type VARCHAR(50) NOT NULL,**

**subject VARCHAR(255),**

**content TEXT,**

**created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,**

**status VARCHAR(50)**

**);**

## **Security Guidelines**

### **Authentication**

* JWT-based authentication
* Token expiration: 24 hours
* Refresh token mechanism
* Rate limiting implemented

### **Data Protection**

* All passwords hashed using bcrypt
* TLS/SSL encryption for all API endpoints
* Data encryption at rest
* Regular security audits

### **Best Practices**

1. Input validation on all endpoints
2. XSS protection
3. CSRF tokens implemented
4. SQL injection prevention
5. Regular dependency updates

## **Testing**

### **Backend Testing**

***# Run all tests***

**python -m pytest**

***# Run specific test suite***

**python -m pytest tests/test\_onboarding.py**

### **Frontend Testing**

***# Run all tests***

**npm test**

***# Run with coverage***

**npm test -- --coverage**

## **Deployment**

### **Docker Deployment**

***# Build and run with Docker Compose***

**docker-compose up --build**

***# Scale specific services***

**docker-compose up --scale backend=3**

### 

### 

### **Manual Deployment**

1. Set up production environment
2. Configure reverse proxy (Nginx recommended)
3. Set up SSL certificates
4. Configure environment variables
5. Run database migrations
6. Start application servers

### 

### 

### **Monitoring**

* Prometheus metrics exposed
* Grafana dashboards available
* Error tracking via Sentry
* Performance monitoring via New Relic

## 

## 

## 

## 

## 

## **Troubleshooting**

### **Common Issues**

1. Database Connection Issues
   * Check connection string
   * Verify network access
   * Check credentials
2. Email Service Issues
   * Verify SMTP credentials
   * Check rate limits
   * Validate email templates

### **Support**

For technical support:

* Contact: mrigankraj2003@gmail.com