CRM Backend API Guide (Phase 1)

Deployed Backend URL: https://crm-be-deployed.onrender.com/api-docs

This document explains how the backend APIs work, what endpoints are available, and how your frontend should interact with them.
The system is built with Node.js, deployed on **Render** (backend + Postgres DB).

Authentication & Roles

Login

- **POST /auth/login**
- Input: { email, password }
- Output: { accessToken, user }
- This token must be sent in the Authorization: Bearer <token> header for all other requests.

Signup

- Disabled for normal users.
- The **Owner** is added directly to the database by us.
- The **Owner** then invites other team members (see Invite flow below).

Roles

- **Owner** → Full access, can invite managers/staff, manage everything.
- **Manager** → Can see everything in the organization, manage workflows, but can't delete company/billing.
- **Staff** → Can only see their own contacts, deals, and tasks.

User Invitations (Instead of Signup)

Send Invite

- **POST /auth/invite**
- Owner/Manager enters an email + role.
- The system creates an invitation in the DB and sends an email with a link.

Accept Invite

- **POST /auth/accept-invitation**
- New user clicks email link → fills form (name + password).
- The system creates their account with the pre-assigned role.
- The invitation token expires after 7 days and can only be used once.

Dashboard APIs

Get KPIs

- **GET /dashboard/kpis**
- Shows quick stats:
  - Total contacts
  - Total deals
  - Tasks due today
  - Recent activities

This is what you show on the dashboard cards.

Contacts APIs

Get Contacts

- **GET /contacts**
- Supports filters: search, status, tags, pagination.
- Staff → only their contacts.
- Managers/Owners → all organization contacts.

Get Contact Details

- **GET /contacts/{id}**

Create Contact

- **POST /contacts**

Update Contact

- **PATCH /contacts/{id}**

Delete Contact

- **DELETE /contacts/{id}** (soft delete → just marks as inactive).

Timeline

- **GET /contacts/{id}/timeline**
- Shows notes, calls, emails, updates.
- New timeline entries are auto-created when something changes (e.g., contact updated, task completed).

Deals APIs

Get Deals

- **GET /deals**
- Returns sales pipeline view with stages.

Create Deal

- **POST /deals**
- Links to a contact.

Move Deal Between Stages

- **POST /pipelines/{id}/stages**
- Used for drag & drop in pipeline view.

Get Pipeline

- **GET /pipelines**
- Shows all pipelines + stages + stats.

Tasks APIs

Get Tasks

- **GET /tasks**

Create Task

- **POST /tasks**

Update Task

- **PATCH /tasks/{id}**

Complete Task

- When a task is marked completed, a timeline entry is automatically created for the related contact.

Reminders

Use Cases:

- Dashboard notification badges
- Daily reminder emails

Permission Notes:

- Staff users can only see their own task reminders
- Managers and owners can view any user's reminders using `assigneeId` parameter
- Default behavior returns current user's tasks only

Integration Tips:

- Use for notification systems and dashboard alerts
- Combine with real-time updates for live notifications
- Cache results for performance in high-traffic scenarios
- Consider implementing browser notifications for overdue tasks

Role-Based Security

- Every request must include a valid **JWT access token**.
- Role determines what data you see:
    - **Owner:** all data + invite others.
    - **Manager:** all data, can invite staff.
    - **Staff:** only their own assigned data.

How It All Fits Together

- **Users** own **Contacts**.
- **Contacts** have **Deals** + **Tasks**.
- **Every change** (like updating contact, moving deal, completing task) → creates a **Timeline entry**.
- **Dashboard** pulls summary stats from all this data.

How Role-Based Login & Access Works

1. **No Public Signup**
    - We don't allow random people to sign up.
    - The **Owner account** is manually added in the database.
2. **Invitation System Instead**
    - Only the Owner (and in some cases, Managers) can invite new users.
    - An invitation email is sent with a link.
    - The invited user accepts the invite, sets their password, and joins with the pre-assigned role.
3. **Role-Based Access Control (RBAC)**
    - Every API checks the user's role before returning data.
    - Staff → only their own contacts/deals/tasks.
    - Managers/Owners → all organization data.
4. **Testing & Request Bodies**
    - All API request schemas are available in **Swagger UI**.
    - You can test the full flow (login, contacts, deals, tasks, etc.) directly in Swagger with the correct request body examples.

Phase 2

Templates, Campaigns & Settings Routes

Authentication: All requests require Bearer token in header:

Authorization: Bearer YOUR_JWT_TOKEN

TEMPLATES API

Templates are reusable content for emails, SMS, invoices, proposals, and contracts.

Get All Templates

GET /templates

What it does: Fetch all templates with filtering and pagination

Query Parameters:

- **type (optional): Filter by type (email, sms, invoice, proposal, contract)**
- **category (optional): Filter by category name**
- **search (optional): Search in template name and content**
- **isActive (optional): Filter active/inactive templates (true/false)**
- **page (optional): Page number (default: 1)**
- **pageSize (optional): Items per page (default: 25, max: 100)**

Example Request:

GET /templates?type=email&search=welcome&page=1&pageSize=10

Response:


```
{
  "success": true,
  "data": {
    "templates": [
      {
        "id": "uuid",
        "name": "Welcome Email Template",
```

```
      "type": "email",
      "subject": "Welcome to {{company_name}}",
      "content": "Hello {{first_name}}...",
      "isDefault": false,
      "isActive": true,
      "usage": 5,
      "creator": {
        "firstName": "John",
        "lastName": "Doe"
      }
    }
  ],
  "meta": {
    "pagination": {
      "page": 1,
      "total": 25,
      "totalPages": 3
    }
  }
 }
}
```

Get Single Template

GET /templates/:id

What it does: Get detailed information about a specific template

Response: Single template object with full details including creator info.

Create Template

POST /templates

What it does: Create a new template

Required Fields:

- name: Template name (1-200 characters)
- type: Template type (email, sms, invoice, proposal, contract)
- content: Template content (required)

Optional Fields:

- subject: Email subject line (max 500 characters)
- htmlContent: HTML version of content
- variables: Array of variable names like ["first_name", "company_name"]
- category: Category name (max 100 characters)
- tags: Array of tags like ["welcome", "onboarding"]
- isDefault: Set as default template for this type
- settings: Object with template settings

Example Request:

```
{
  "name": "Welcome Email Template",
  "type": "email",
  "subject": "Welcome to {{company_name}}",
  "content": "Hello {{first_name}}, welcome to our platform!",
  "variables": ["first_name", "company_name"],
  "category": "onboarding",
  "tags": ["welcome", "email"]
}
```

Update Template

PATCH /templates/:id

What it does: Update an existing template (cannot update system templates)

Fields: Same as create, all optional

Delete Template

DELETE /templates/:id

What it does: Soft delete a template (marks as inactive). Cannot delete system templates.

Duplicate Template

POST /templates/:id/duplicate

What it does: Create a copy of an existing template

Required Fields:

- name: New name for the duplicated template

Track Template Usage

POST /templates/:id/use

What it does: Increment usage counter when template is used (call this when sending emails/campaigns using template)

CAMPAIGNS API

Campaigns are for sending bulk emails or SMS to multiple contacts.

Get All Campaigns

GET /campaigns

What it does: Fetch all campaigns with filtering and statistics

Query Parameters:

- **status (optional): Filter by status (draft, scheduled, sending, sent, paused, cancelled)**
- **type (optional): Filter by type (email, sms)**
- **search (optional): Search in campaign name and subject**
- **page, pageSize: Pagination**

Response: Campaigns with statistics (open rates, click rates, etc.)

Get Single Campaign

GET /campaigns/:id

What it does: Get detailed campaign information including template details

Create Campaign

POST /campaigns

What it does: Create a new campaign

Required Fields:

- **name: Campaign name (1-200 characters)**
- **type: Campaign type (email or sms)**

- subject: Email subject line (required, max 500 characters)

Optional Fields:

- templateId: Use existing template (UUID)
- content: Campaign content (if not using template)
- htmlContent: HTML version (for emails)
- scheduledAt: Schedule for later (ISO date string)
- tags: Array of tags
- settings: Campaign settings object

Example Request:

```
{
  "name": "Q4 Product Launch Campaign",
  "type": "email",
  "subject": "Introducing Our New Product Line",
  "templateId": "template-uuid-here",
  "scheduledAt": "2024-12-25T10:00:00Z",
  "tags": ["product-launch", "q4"]
}
```

Update Campaign

PATCH /campaigns/:id

What it does: Update campaign (cannot update sent/sending campaigns)

Delete Campaign

DELETE /campaigns/:id

What it does: Delete campaign (only draft or cancelled campaigns)

Get Campaign Recipients

GET /campaigns/:id/recipients

What it does: Get list of recipients for a campaign

Query Parameters:

- status (optional): Filter by recipient status (pending, sent, delivered, opened, clicked, bounced, failed)
- page, pageSize: Pagination

Add Recipients to Campaign

POST /campaigns/:id/recipients

What it does: Add contacts to campaign recipient list

Required Fields:

- contactIds: Array of contact UUIDs

Example Request:

```
{
  "contactIds": [
    "contact-uuid-1",
    "contact-uuid-2",
    "contact-uuid-3"
  ]
}
```

Response: Shows how many were added and how many were skipped (already recipients)

Send Campaign

POST /campaigns/:id/send

What it does: Send the campaign to all recipients immediately

Note: Campaign must have recipients and be in sendable status (draft, scheduled, paused)

Pause Campaign

POST /campaigns/:id/pause

What it does: Pause a scheduled or sending campaign

Get Campaign Statistics

GET /campaigns/:id/stats

What it does: Get detailed analytics for a campaign

Response:


```
{
  "success": true,
  "data": {
    "totalRecipients": 1000,
    "sent": 1000,
    "delivered": 950,
    "opened": 380,
    "clicked": 95,
    "bounced": 50,
    "openRate": "40.00",
    "clickRate": "10.00",
    "deliveryRate": "95.00"
  }
}
```


SETTINGS API

Settings manage organization profile, user accounts, and system preferences.

Organization Settings

Get Organization Profile
GET /settings/organization

What it does: Get company/organization details

Update Organization Profile
PATCH /settings/organization

What it does: Update organization settings (Owner/Manager only)

Optional Fields:

- **name: Company name (max 200 characters)**
- **industry: Industry name (max 100 characters)**

- employeeCount: Employee range ("1-10", "11-50", "51-200", "201-500", "500+")
- website: Company website URL
- contactPhone: Company phone (max 20 characters)
- address: Company address (max 500 characters)
- timezone: Timezone string (e.g., "America/New_York")
- currency: Currency code ("USD", "EUR", "GBP", "CAD", "AUD")
- logo: Logo image URL
- settings: Custom settings object

User Profile Settings

Get Current User Profile
GET /settings/profile

What it does: Get logged-in user's profile information

Update User Profile
PATCH /settings/profile

What it does: Update current user's profile

Optional Fields:

- firstName: First name (max 100 characters)
- lastName: Last name (max 100 characters)
- phone: Phone number (max 20 characters)
- avatar: Profile picture URL
- timezone: User's timezone
- language: Language preference ("en", "es", "fr", "de", "pt")
- preferences: User preferences object

Change Password
PATCH /settings/password

What it does: Change user's password

Required Fields:

- currentPassword: Current password
- newPassword: New password (min 8 characters)
- confirmPassword: Confirm new password

Team Management (Owner Only)

Get Team Members
GET /settings/team

What it does: Get list of all team members in organization

Update Team Member
PATCH /settings/team/:userId

What it does: Update another user's role or status (Owner only)

Optional Fields:

- **role: User role ("owner", "manager", "staff")**
- **status: User status ("active", "inactive")**

Note: Cannot modify your own account

Remove Team Member
DELETE /settings/team/:userId

What it does: Remove team member (marks as inactive)

Note: Cannot remove yourself

System Information (Owner Only)

Get System Info
GET /settings/system

What it does: Get system statistics and information

Response: System version, uptime, organization stats, and feature flags

Variable Replacement in Templates

When using templates, variables like `{{first_name}}` and `{{company_name}}` need to be replaced with actual values before sending. The template's `variables` array tells you which variables are available.

INVOICES API

Invoices are for billing contacts with itemized services/products and processing payments via Stripe integration.

Get All Invoices

GET /invoices

What it does: Fetch all invoices with filtering and pagination

Query Parameters:

- `status` (optional): Filter by status (draft, sent, paid, overdue, cancelled)
- `contactId` (optional): Filter by contact UUID
- `search` (optional): Search in invoice number and notes
- `dateFrom` (optional): Filter invoices from date (ISO format)
- `dateTo` (optional): Filter invoices to date (ISO format)
- `page` (optional): Page number (default: 1)
- `pageSize` (optional): Items per page (default: 25, max: 100)

Example Request:

GET /invoices?status=sent&page=1&pageSize=10

Get Single Invoice

GET /invoices/:id

What it does: Get detailed information about a specific invoice including payment history

Response: Single invoice object with full details including contact info, items, and payments.

Create Invoice

POST /invoices

What it does: Create a new invoice for a contact

Required Fields:

- `contactId`: Contact UUID
- `items`: Array of invoice items

Optional Fields:

- `dueDate`: Due date (ISO format, default: 30 days from now)
- `notes`: Invoice notes
- `terms`: Payment terms
- `paymentTerms`: Payment terms description (default: "Net 30")

Update Invoice

PATCH /invoices/:id

What it does: Update an existing invoice (cannot update paid invoices)

Fields: Same as create, all optional. Can also update `status`.

Delete Invoice

DELETE /invoices/:id

What it does: Delete an invoice (cannot delete paid invoices)

Send Invoice

POST /invoices/:id/send

What it does: Send invoice via email to the contact

Optional Fields:

- `channels`: Array of channels (default: ["email"])
- `customMessage`: Custom message to include in email
- `template`: Email template customization object

Available template variables: `{{contactFirstName}}`, `{{invoiceNumber}}`, `{{organizationName}}`, `{{dueDateFormatted}}`, etc.

Record Manual Payment

POST /invoices/:id/payments

What it does: Record a manual payment for an invoice

Required Fields:

- `amount`: Payment amount (must be > 0)
- `paymentMethod`: Payment method (cash, check, credit_card, bank_transfer, paypal, stripe, other)

Optional Fields:

- `paymentDate`: Payment date (default: today)
- `reference`: Reference number or check number
- `notes`: Payment notes

Create Stripe Payment Intent

POST /invoices/:id/pay

What it does: Create a Stripe payment intent for online payment

Optional Fields:

- `amount`: Custom payment amount (default: remaining invoice amount)
- `currency`: Currency code (default: "usd")

Response:

{

"success": true,
"data": {
"clientSecret": "pi_xxx_secret_xxx",
"paymentIntentId": "pi_xxx",
"amount": 1500.00,
"currency": "usd",
"invoice": {
"id": "invoice-uuid",
"number": "ABC-2024-0001",
"total": 1500.00,
"remainingAmount": 1500.00
}
}
}

Use the `clientSecret` with Stripe.js to complete payment on frontend.

Get Invoice Payments

GET /invoices/:id/payments

What it does: Get all payments for a specific invoice

Response: Array of payment records with summary totals.

Stripe Webhook Handler

POST /invoices/webhook/stripe

What it does: Handle Stripe webhook events for payment confirmations (internal use)

Note: This endpoint processes Stripe webhooks automatically. Configure your Stripe webhook to point to this endpoint.

Invoice Statistics

GET /invoices/stats

What it does: Get invoice statistics and summary data

Query Parameters:

- `dateFrom` (optional): Start date for statistics
- `dateTo` (optional): End date for statistics

Role-Based Access Control

- **Owner/Manager:** Full access to all invoices
- **Staff:** Can only see their own created invoices
- All payment operations require proper role permissions

Integration Notes

- Invoice numbers are auto-generated per organization
- Totals are automatically calculated from items
- Timeline entries are created for all major actions
- Stripe webhooks automatically update payment status

UPLOAD API

Upload User Avatar

POST /upload/avatar/user/:id

What it does: Upload avatar image for a user

Content-Type: `multipart/form-data` Required Field: `avatar` (image file)

Permissions:

- Users can only update their own avatar

- Owners can update any user's avatar

File Requirements:

- Image files only (JPEG, PNG, GIF, WebP)
- Maximum size: 5MB

Example Request:

const formData = new FormData();

formData.append('avatar', imageFile);

Upload Contact Avatar

POST /upload/avatar/contact/:id

What it does: Upload avatar image for a contact

Content-Type: `multipart/form-data` Required Field: `avatar` (image file)

Permissions:

- Staff can only update avatars for their own contacts
- Managers/Owners can update any contact's avatar

File Requirements: Same as user avatar

Upload Organization Logo

POST /upload/logo

What it does: Upload logo for the organization

Content-Type: `multipart/form-data` Required Field: `logo` (image file)

Permissions:

- Only Owners and Managers can upload logo
- Staff users cannot upload logo

File Requirements: Same as avatars

File Management Notes

Automatic Cleanup:

- Old files are automatically deleted when new ones are uploaded
- No manual file deletion needed

File URLs:

- All uploaded files get permanent S3 URLs
- URLs are directly accessible (no authentication needed)
- Files persist across server deployments

Database Integration:

- User avatar URLs are stored in `User.avatar` field
- Contact avatar URLs are stored in `Contact.avatar` field
- Organization logo URLs are stored in `Organization.logo` field
- Existing GET APIs automatically return these URLs

Error Handling: Common error responses:

- `NO_FILE`: No file was uploaded
- `UPLOAD_ERROR`: File type not allowed or size exceeded
- `FORBIDDEN`: User doesn't have permission
- `USER_NOT_FOUND` / `CONTACT_NOT_FOUND`: Invalid ID

Communication (Inbox & Messages)

Frontend Routes

- `/app/inbox` (unified inbox)
- `/app/inbox/[conversationId]` (conversation detail)

API Endpoints

Messages:

- `POST /messages` → Send a new message (Email or SMS) to a contact
- `GET /messages/{id}` → Retrieve message details

- `PATCH /messages/{id}` → Update message status (mark as read/unread)

Inbox:

- `GET /inbox/conversations` → Retrieve inbox conversations with filtering and pagination
- `GET /inbox/conversations/{id}` → Retrieve conversation details with message history
- `POST /inbox/conversations/{id}/messages` → Send a reply message in an existing conversation
- `POST /inbox/conversations/{id}/assign` → Assign conversation to a team member
- `PATCH /inbox/conversations/{id}/tags` → Update conversation tags

Key Features:

- Unified Email & SMS messaging through single interface
- Conversations automatically group related messages by contact and type
- Support for conversation assignment, tagging, and status management
- Message status tracking (pending, sent, delivered, failed)
- Built-in integration points for email providers (SendGrid) and SMS providers (Twilio)

Frontend Routes

- `/app/products` (Products list and management)

API Endpoints

- `GET /products` → List products with filtering, search, and pagination
- `GET /products/{id}` → Get single product details
- `POST /products` → Create new product
- `PATCH /products/{id}` → Update product information
- `DELETE /products/{id}` → Delete product

Query Parameters for GET /products:

- `page`, `pageSize` → Pagination controls
- `search` → Search in name, SKU, description
- `category` → Filter by product category
- `status` → Filter by active/inactive status

Key Features:

- Automatic SKU generation if not provided
- SKU uniqueness validation
- Search across name, SKU, and description
- Category-based organization
- Status management (active/inactive)

Analytics & Reports

Frontend Routes

- `/app/reports` (Overview dashboard)
- `/app/reports/sales` (Sales analytics)
- `/app/reports/marketing` (Marketing metrics)
- `/app/reports/customers` (Customer insights)

API Endpoints

- `GET /reports/overview` → Dashboard KPIs and activity charts
- `GET /reports/sales` → Deal pipeline and revenue analytics
- `GET /reports/marketing` → Campaign performance and message metrics
- `GET /reports/customers` → Customer growth and top customer analysis
- `GET /reports/export` → Export data as CSV download

Query Parameters (all report endpoints):

- `period` → Time range: `7d`, `30d`, `90d`, `1y` (default: `30d`)

Export Parameters:

- `type` → Data type: `contacts`, `deals`, `invoices`, `products`
- `format` → File format: `csv`, `excel` (default: `csv`)

Report Data Structure:

- **Overview**: KPIs (contacts, deals, revenue, invoices), activity charts
- **Sales**: Deal stages, revenue over time, top performers, pipeline metrics
- **Marketing**: Campaign stats, message performance, channel effectiveness
- **Customers**: Growth trends, top customers by value, acquisition sources

Usage Example:

// Export contacts as CSV

```
window.open('/api/reports/export?type=contacts&format=csv');
```