# BCombs REST API Documentation
# Complete Reference

Vardhan Patel, Tilak Patel, Dipen Patel, Devin Bhatt, Isaiah Benzoni

December 6, 2024

## Contents

# 1 Overview

This REST API uses JSON payloads. All requests and responses define the Content-Type header as "application/json". Authentication is implemented via JWT tokens with a 1-hour expiration time. All requests except authentication and account creation endpoints require authentication.

## 1.1 Authentication Requirements

All authenticated requests must include:

```
Authorization: Bearer <jwt_token>
Content-Type: application/json
```

## 1.2 Common HTTP Status Codes

- 200: Success
- 201: Created successfully
- 400: Bad Request - Missing or invalid parameters
- 401: Unauthorized - Invalid or missing authentication
- 403: Forbidden - Insufficient privileges
- 404: Not Found - Resource doesn't exist
- 409: Conflict - Resource already exists
- 500: Server Error

# 2 Data Models

## 2.1 User Object

```
{
  "id": int,
  "email": string,
  "firstname": string,
  "lastname": string,
  "created_at": datetime,
  "updated_at": datetime,
  "registration_status": string
}
```

### 2.2 Organization Object

```
{
  "id": int,
  "name": string,
  "type_id": int,
  "industry": string,
  "primary_address": string,
  "phone_number": string,
  "user_id": int,
  "role_in_company": string,
  "audience": string,
  "pain_points": string,
  "get_started_preference": string,
  "responsibilities": string,
  "goals": string,
  "organization_code": string
}
```

### 2.3 Form Object

```
{
  "id": int,
  "user_id": int,
  "organization_id": int,
  "form_type": string,
  "categories": string,
  "association": string,
  "title": string,
  "description": string,
  "copyright_protected": boolean,
  "created_at": datetime,
  "updated_at": datetime,
  "fields": [
    {
      "type": string,
      "label": string,
      "placeholder": string,
      "required": boolean,
      "options": array,
      "position": int
    }
  ]
}
```

### 2.4 Group Object

```
{
  "id": int,
  "organization_id": int,
  "name": string,
  "subgroups": string,
  "user_emails": string,
  "form_type": string,
  "capacity": int
}
```

## 3 API Endpoints

### 3.1 Authentication Endpoints (2)

#### 3.1.1 1. Login

**POST /api/auth/login**

| **Request** | **Response** |
|---|---|
| **Route parameters:** None<br><br>**Query parameters:** None<br><br>**Body:**<br>{<br>”email”: string,<br>”password”: string<br>} | **Status:**<br>200 if successful<br>401 if invalid credentials<br><br>**Body:**<br>{<br>”token”: string,<br>”user”: User object<br>} |

#### 3.1.2 2. Get CSRF Token

**GET /api/auth/csrf-token**

| **Request** | **Response** |
|---|---|
| **Route parameters:** None<br>**Query parameters:** None<br>**Body:** None | **Status:** 200<br>**Body:**<br>{”csrfToken”: string} |

### 3.2 User Management Endpoints (6)

#### 3.2.1 1. Create Account

**POST /api/create-account**

| **Request** | **Response** |
|---|---|
| **Route parameters:** None<br><br>**Query parameters:** None<br><br>**Body:**<br>{<br>    ”email”: string,<br>    ”password”: string,<br>    ”firstname”: string,<br>    ”lastname”: string<br>} | **Status:**<br>201 if created<br>409 if email exists<br><br>**Body:**<br>{<br>    ”message”: string,<br>    ”userId”: int<br>} |

### 3.2.2 2. Get User Details by Email

**GET /api/get-user-details**

| **Request** | **Response** |
|---|---|
| **Route parameters:** None<br>**Query parameters:**<br>email: string<br>**Body:** None | **Status:**<br>200 if found<br>404 if not found<br>**Body:** {<br>    ”message”: ”User found!”,<br>    ”user”: User object<br>} |

### 3.2.3 3. Get User Details by ID

**GET /api/get-user-details-by-id/:userId**

| **Request** | **Response** |
|---|---|
| **Route parameters:**<br>userId: string<br>**Query parameters:** None<br>**Body:** None | **Status:**<br>200 if found<br>404 if not found<br>**Body:** {<br>    ”message”: ”User found!”,<br>    ”user”: User object<br>} |

### 3.2.4 4. Update User Role

**PUT /api/update-user-role**

| Request | Response |
|---|---|
| **Route parameters:** None<br>**Query parameters:** None<br>**Body:**<br>{<br>"userId": int,<br><br>"userTypeId": int<br>} | **Status:**<br>200 if updated<br>404 if not found<br>**Body:** {<br>"message": "User role updated successfully"<br>} |

#### 3.2.5 5. Delete User

**DELETE /api/delete-user/:userId**

| Request | Response |
|---|---|
| **Route parameters:**<br>userId: string<br>**Query parameters:** None<br>**Body:** None | **Status:**<br>200 if deleted<br>404 if not found<br>**Body:** {<br>"message": "User deleted successfully"<br>} |

#### 3.2.6 6. Join Organization Without Code

**POST /api/create-account-without-code**

| Request | Response |
|---|---|
| **Route parameters:** None<br>**Query parameters:** None<br>**Body:**<br>{<br>"userId": int<br>} | **Status:**<br>201 if successful<br>404 if user not found<br>**Body:** {<br>"message": string,<br>"userId": int<br>} |

## 3.3 Organization Management Endpoints (3)

#### 3.3.1 1. Complete Organization Setup

**POST /api/complete-organization-setup**

| **Request** | **Response** |
|---|---|
| **Route parameters:** None<br>**Query parameters:** None<br>**Body:**<br>{<br>"userId": int,<br><br>"organizationName": string,<br>"organizationType": int,<br>"industry": string,<br>"roleInCompany": string,<br>"phoneNumber": string,<br>"primaryAddress": string,<br>"audience": string,<br>"painPoints": string,<br>"startingPoint": string,<br>"responsibilities": string,<br>"goals": array<br>} | **Status:**<br>201 if successful<br>404 if user not found<br>**Body:** {<br>"message": "Organization setup complete",<br>"orgId": int<br>} |

### 3.3.2 2. Join Organization

**POST /api/join-organization**

| **Request** | **Response** |
|---|---|
| **Route parameters:** None<br>**Query parameters:** None<br>**Body:**<br>{<br>"userId": int,<br><br>"organizationCode": string<br>} | **Status:**<br>200 if successful<br>404 if not found<br>**Body:** {<br>"message": "User added to organization",<br>"organizationId": int<br>} |

### 3.3.3 3. Get Organization By User

**GET /api/get-organization-by-user/:userId**

| **Request** | **Response** |
|---|---|
| **Route parameters:**<br>userId: string<br>**Query parameters:** None<br>**Body:** None | **Status:**<br>200 if found<br>404 if not found<br>**Body:** {<br>"message": "Organization found!",<br>"organization": Organization object<br>} |

## 3.4 Form Management Endpoints (7)

### 3.4.1 1. Create Form

**POST /api/forms/create**

| **Request** | **Response** |
|---|---|
| **Route parameters:** None<br>**Query parameters:** None<br>**Body:**<br>Form object without id | **Status:**<br>201 if created<br>500 if error<br>**Body:** {<br>"message": "Form created successfully",<br>"form": Form object<br>} |

### 3.4.2 3. Get Form By ID

**GET /api/forms/:formId**

| **Request** | **Response** |
|---|---|
| **Route parameters:**<br>formId: string<br>**Query parameters:** None<br>**Body:** None | **Status:**<br>200 if found<br>404 if not found<br>**Body:** {<br>"form": Form object<br>} |

### 3.4.3 4. Get Forms By User

**GET /api/forms/user/:userId**

| **Request** | **Response** |
|---|---|
| **Route parameters:**<br>userId: string<br>**Query parameters:** None<br>**Body:** None | **Status:**<br>200<br>**Body:** {<br>"forms": Array of Form objects<br>} |

### 3.4.4 5. Get Forms By Organization

**GET /api/forms/organization/:organizationId**

| **Request** | **Response** |
|---|---|
| **Route parameters:**<br>organizationId: string<br>**Query parameters:** None<br>**Body:** None | **Status:**<br>200<br>**Body:** {<br>"forms": Array of Form objects<br>} |

#### 3.4.5 6. Edit Form

**PUT /api/forms/:formId**

| Request | Response |
|---|---|
| **Route parameters:**<br>formId: string<br>**Query parameters:** None<br>**Body:**<br>Form object | **Status:**<br>200 if updated<br>404 if not found<br>**Body:** {<br>"message": "Form updated successfully",<br>"form": Form object<br>} |

#### 3.4.6 7. Delete Form

**DELETE /api/forms/:formId**

| Request | Response |
|---|---|
| **Route parameters:**<br>formId: string<br>**Query parameters:** None<br>**Body:** None | **Status:**<br>200 if deleted<br>404 if not found<br>**Body:** {<br>"message": "Form deleted successfully"<br>} |

### 3.5 Group Management Endpoints (5)

#### 3.5.1 1. Create/Update Group

**POST /api/groups/create**

| Request | Response |
|---|---|
| **Route parameters:** None<br>**Query parameters:** None<br>**Body:**<br>Group object | **Status:**<br>201 if created<br>400 if invalid capacity<br>**Body:** {<br>"message": "Group created successfully"<br>} |

#### 3.5.2 2. Add Users to Group

**POST /api/groups/emails**

| Request | Response |
|---|---|
| **Route parameters:** None<br>**Query parameters:** None<br>**Body:**<br>{<br>”organization$_id$” : $int$,<br><br>”emails”: array of strings<br>} | **Status:**<br>200 if successful<br>404 if group not found<br>**Body:** {<br>”message”: ”Users added to group successfully”<br>} |

### 3.5.3 3. Get Group Details

**GET /api/groups/:organizationId**

| Request | Response |
|---|---|
| **Route parameters:**<br>organizationId: string<br>**Query parameters:** None<br>**Body:** None | **Status:**<br>200<br>**Body:**<br>Array of Group objects |

### 3.5.4 4. Get All Group Emails

**GET /api/groups/allEmails/:organizationId**

| Request | Response |
|---|---|
| **Route parameters:**<br>organizationId: string<br>**Query parameters:** None<br>**Body:** None | **Status:**<br>200<br>**Body:**<br>Array of email strings |

### 3.5.5 5. Delete Group

**DELETE /api/groups/delete/:id**

| Request | Response |
|---|---|
| **Route parameters:**<br>id: string<br>**Query parameters:** None<br>**Body:** None | **Status:**<br>200 if deleted<br>404 if not found<br>**Body:** {<br>”message”: ”Group deleted successfully”<br>} |

## 3.6 Password Reset Endpoints (4)

### 3.6.1 1. Request Password Reset

**POST /api/reset-password**

| **Request** | **Response** |
|---|---|
| **Route parameters:** None<br>**Query parameters:** None<br>**Body:**<br>{<br>"email": string<br>} | **Status:**<br>200 if email sent<br>404 if email not found<br>**Body:** {<br>"message": "Password reset email sent"<br>} |

#### 3.6.2 2. Resend Reset Code

**POST /api/resend-reset-code**

| **Request** | **Response** |
|---|---|
| **Route parameters:** None<br>**Query parameters:** None<br>**Body:**<br>{<br>"email": string<br>} | **Status:**<br>200 if sent<br>404 if email not found<br>**Body:** {<br>"message": "A new reset code has been sent to your email"<br>} |

#### 3.6.3 3. Verify Reset Code

**POST /api/verify-reset-code**

| **Request** | **Response** |
|---|---|
| **Route parameters:** None<br>**Query parameters:** None<br>**Body:**<br>{<br>"code": string<br>} | **Status:**<br>200 if valid<br>400 if invalid/expired<br>**Body:** {<br>"message": "Code verified",<br>"resetToken": string<br>} |

#### 3.6.4 4. Set New Password

**POST /api/set-new-password**

| **Request** | **Response** |
|---|---|
| **Route parameters:** None<br>**Query parameters:** None<br>**Body:**<br>{<br>"newPassword": string,<br>"resetToken": string<br>} | **Status:**<br>200 if updated<br>400 if invalid token<br>**Body:** {<br>"message": "Password has been updated successfully"<br>} |

## 3.7 Search Endpoints (2)

### 3.7.1 1. Get Events

**GET /api/search/events**

| Request | Response |
|---|---|
| **Route parameters:** None<br>**Query parameters:** None<br>**Body:** None | **Status:**<br>200<br>**Body:**<br>Array of Event objects |

### 3.7.2 2. Get Event Types

**POST /api/search/event-types**

| Request | Response |
|---|---|
| **Route parameters:** None<br>**Query parameters:** None<br>**Body:**<br>{<br>"id": int<br>} | **Status:**<br>200<br>**Body:**<br>Array of event type strings |

## 3.8 Attendance Endpoints (4)

### 3.8.1 1. Get Event Attendance Details

**GET /api/attendance/:id**

| Request | Response |
|---|---|
| **Route parameters:** id (Event ID)<br>**Query parameters:** None<br>**Body:** None | **Status:** 200 if found, 404 if not found<br>**Body:** {<br>"attendanceDetails": Array of attendance objects<br>} |

### 3.8.2 2. Get Event Students

**GET /api/attendance/students**

| Request | Response |
|---|---|
| **Route parameters:** None<br>**Query parameters:** None<br>**Body:** None | **Status:** 200<br>**Body:** {<br>"students": Array of student objects<br>} |

#### 3.8.3 3. Start Event Attendance

**POST /api/attendance-event/start**

| Request | Response |
|---|---|
| **Route parameters:** None<br><br>**Query parameters:** None<br>**Body:** {<br><br>"eventId": int,<br>"date": string,<br>"startTime": string,<br>"endTime": string<br>} | **Status:** 200 if created, 404 if event not found<br>**Body:** {<br>"message": "Attendance record created successfully"<br>} |

#### 3.8.4 4. Save Event Attendance

**POST /api/attendance-event/save**

| Request | Response |
|---|---|
| **Route parameters:** None<br>**Query parameters:** None<br>**Body:** {<br><br>"eventId": int,<br>"attendanceData": Array of attendance updates<br>} | **Status:** 200<br>**Body:** {<br>"message": "Attendance data saved successfully"<br>} |