Assignment Description

You have given High-Level Design and Low-Level Design of lite version of typical NewsLetters App. You need to understand the requirement

Object of this assignment is to test your requirement understand knowledge and skill of understanding or generating code using AI tools. You are free to refer code of any of Open-Source NewsLetters App https://medevel.com/10-email-marketing-automation/

Expected Result is…

1. **Back End Code, API, share your repository link**
2. You should implement only CRUD operation with user authentication
3. It is up to you whether you can implement *Nice to have requirement* or not
4. You are free to use AI tool to generate code

This App should provide a basic Back End for managing email campaigns and subscriber lists. Here are some key features:

Key Features and Requirements

1. **List Management**: It should supports managing large subscriber lists, allowing users to add subscribers manually or import them via CSV files. Allow only private list and it should be visible only to user
2. **Custom Fields**: The application enables users to create custom fields for subscribers, which can be utilized in newsletters through merge tags.
3. **Segmentation**: Users can segment their lists based on predefined rules, facilitating targeted email campaigns without the need for multiple lists.
4. **RSS Campaigns**: It should automatically generate campaigns based on new entries in specified RSS feeds, streamlining content distribution. *(Nice to have requirement)*
5. **GPG Encryption**: The application supports GPG public key encryption, allowing secure communication with subscribers. *(Nice to have requirement)*
6. **Click Statistics**: After sending campaigns, users can track click statistics for each link included in the email, providing insights into engagement. There should be appropriate security measure taken in code to prevent unauthorised entry to database *(Nice to have requirement)*
7. **Template Editor**: It should a built-in template editor, allowing users to design newsletters with ease. *(Nice to have requirement)*
8. **Automation**: The platform supports automation triggers, enabling users to send messages based on specific actions. *(Nice to have requirement)*
9. **Multi-Tenancy**: It should be configured to support multiple organizations, ensuring data isolation and security.
10. **Send via Any Provide:** Supports SMTP protocol to send out newsletters e.g. SES AWS SparkPost, SendGrid and Mailgun

High-Level Design

1. Architecture Overview

The architecture of the application can be divided into three main layers:

- **Backend Layer**: Implemented using **NestJS** for handling API requests and business logic.
- **Database Layer**: **PostgreSQL** for data storage and drizzle ORM or **TypeORM** for connectivity, ensuring relational integrity and support for complex queries.

2. Components

- **User Interface (UI)**: *This is only for understanding of requirement*
    - Dashboard for managing campaigns, lists, and statistics.
    - Forms for creating and managing custom fields, lists, and templates.
    - Template editor interface for designing newsletters.
- **API Layer**:
    - RESTful APIs for managing subscribers, lists, campaigns, and templates.
    - Authentication and authorization endpoints for user management.
    - Webhooks for automation triggers and RSS feed monitoring.
- **Database**:
    - Tables for Users, Organizations, Subscribers, Lists, Campaigns, Templates, and Click Stats.
    - Each table will include an ***organization_id*** field to support multi-tenancy.
- **Email Service**:
    - Integration with AWS SES/Gmail or any other E-Mail provide for sending emails and managing email queues.

3. Multi-Tenancy

- Implement multi-tenancy using the ***organization_id*** field in all relevant tables.
- Queries will filter data based on the ***organization_id*** to ensure data isolation.

Low-Level Design

1. Database Schema

Here's a simplified version of the database schema:

Users Table

Column Name	Data Type	Description
id	UUID	Primary Key
email	VARCHAR	User email
password_hash	VARCHAR	Hashed password
organization_id	UUID	Foreign Key to Organizations
role	ENUM	Role (Superadmin, Admin, User)
created_at	TIMESTAMP	Timestamp of account creation

Column Name	Data Type	Description
updated_at	TIMESTAMP	Timestamp of account update

Organizations Table

Column Name	Data Type	Description
id	UUID	Primary Key
name	VARCHAR	Organization name
created_at	TIMESTAMP	Timestamp of organization creation
updated_at	TIMESTAMP	Timestamp of account update

Subscribers Table

Column Name	Data Type	Description
id	UUID	Primary Key
email	VARCHAR	Subscriber email
organization_id	UUID	Foreign Key to Organizations
custom_fields	JSONB	Custom fields in JSON format
gpg_public_key	TEXT	GPG public key for encryption
created_at	TIMESTAMP	Timestamp of subscription

Lists Table

Column Name	Data Type	Description
id	UUID	Primary Key
name	VARCHAR	List name
organization_id	UUID	Foreign Key to Organizations
custom_fields	JSONB	Custom fields in JSON format
created_at	TIMESTAMP	Timestamp of list creation

Campaigns Table

Column Name	Data Type	Description
id	UUID	Primary Key
subject	VARCHAR	Email subject
content	TEXT	Email content
list_id	UUID	Foreign Key to Lists
organization_id	UUID	Foreign Key to Organizations
created_at	TIMESTAMP	Timestamp of campaign creation

Click Stats Table

Column Name	Data Type	Description
id	UUID	Primary Key
campaign_id	UUID	Foreign Key to Campaigns
link	VARCHAR	URL of the clicked link
click_count	INTEGER	Number of clicks
created_at	TIMESTAMP	Timestamp of click

2. API Endpoints

Here are some key API endpoints for the application:

User Management

- POST /api/users/register: Register a new user.
- GET /api/users/:id: Get user details.

Organization Management

- POST /api/organizations: Create a new organization.
- GET /api/organizations: List all organizations

Subscriber Management

- POST /api/subscribers: Add a new subscriber.
- GET /api/subscribers: List subscribers with pagination and filtering.
- PUT /api/subscribers/:id: Update subscriber information.

List Management

- POST /api/lists: Create a new list.
- GET /api/lists: List all lists for the organization.
- PUT /api/lists/:id: Update list details.

Campaign Management

- POST /api/campaigns: Create a new campaign.
- GET /api/campaigns: List all campaigns for the organization.
- POST /api/campaigns/:id/send: Send a campaign.

GPG Encryption

- POST /api/gpg/upload: Upload GPG public key for a subscriber.

3. Frontend Components *(This is only for understanding of requirement)*

- **Dashboard**: Overview of campaigns, lists, and statistics.
- **List Management**: UI for adding, editing, and deleting lists.
- **Subscriber Management**: UI for managing subscribers and custom fields.
- **Campaign Creation**: Interface for designing and sending campaigns.
- **Template Editor**: Integrate GrapeJS and Mosaico for advanced template editing.

4. Automation Triggers *(Optional)*

- Define triggers in the database that can be linked to specific actions (e.g., new subscriber, new RSS feed entry).
- Implement background jobs to process automation tasks.

*****