# Email Sender Program – System Design Overview

## Overview

This document outlines the proposed design for an Email Sender Program, which is responsible for sending various types of emails. The service is designed to support dynamic email templates, logging, and retry mechanisms for failed deliveries. The architecture focuses on being flexible, scalable, and reliable, making it suitable for different clients such as web and mobile applications.

## Key Architectural Components

### Client Applications

These include web and mobile apps that communicate with the Email Sender Service through a well-defined API.

### Micro Services

To promote scalability and separation of concerns, the system is divided into smaller services:

* Email Notifier Service
* SMTP Service
* Customer Service
* Order Service
* Voucher Service
* Template Service

Each of these handles a specific part of the overall functionality.

### API Gateway

An API Gateway sits in front of the microservices. It handles all incoming requests such as creating, updating, or triggering an email notification. It then routes these requests to the appropriate service.

### Layered Architecture (for each Microservice)

Each microservice follows a layered structure to keep the system maintainable and well-organized:

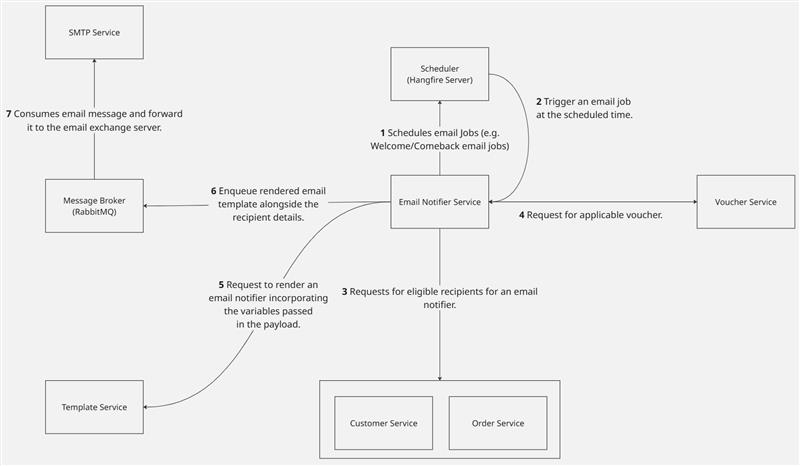
**API Layer (Controllers):** This layer exposes HTTP endpoints and handles incoming API calls.

**Application Layer (Services/Use Cases):** It manages business workflows and coordinates between the domain and infrastructure layers.

**Domain Layer (Core Logic):** Contains the core business rules and models. This layer remains independent of any frameworks or external systems.

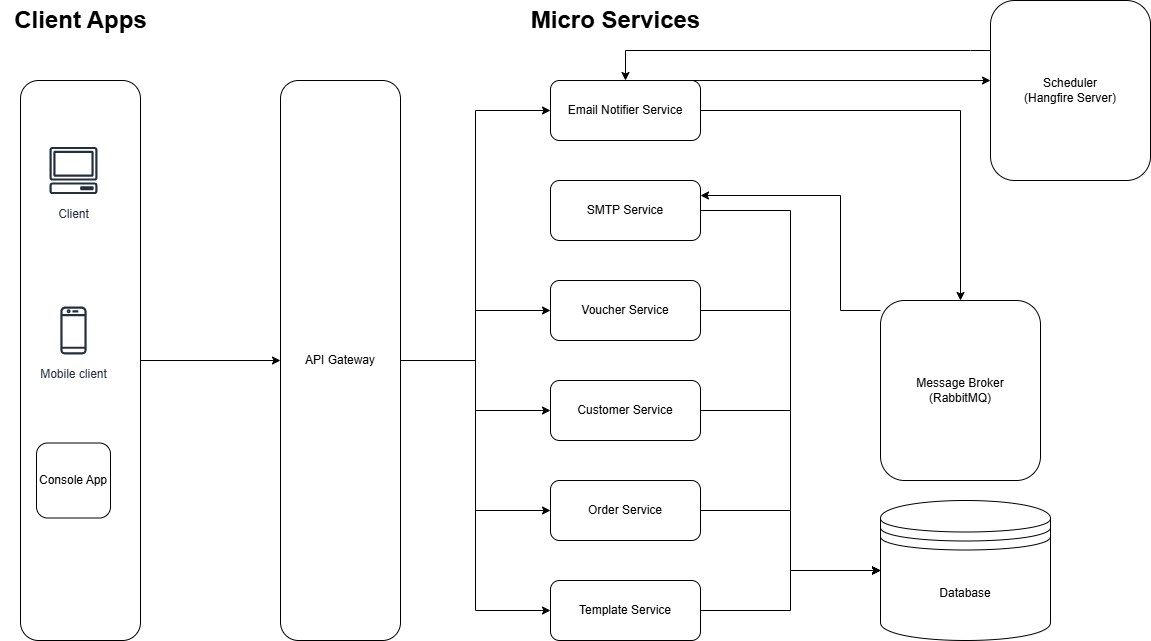
**Infrastructure Layer (Adapters):** Handles the technical details such as database access, external APIs, SMTP clients, and message brokers.

## Flow Diagram



## 

## Architecture Diagram



## Special Techniques and Tools

### MassTransit with RabbitMQ:

We can use MassTransit along with RabbitMQ to implement retry mechanisms. MassTransit provides an easy way to configure delayed retries in case of failures, which improves the reliability of the system.

### Hangfire for Background Jobs:

A dedicated background job scheduler (like Hangfire) is used to manage background tasks. Running this as a separate server helps ensure that scheduled jobs are not executed multiple times in a scaled-out environment.

### Scriban for Template Rendering:

To support dynamic email templates, we use Scriban, a lightweight templating engine. This makes it easy to inject variable content into email templates and allows for flexible updates as new data requirements arise.