**Email Notification Module Design Document**

**1. Introduction**

This document outlines the design of an **Email Notification Module** for a **MERN-based shopping application** that sells subscription-based eSIMs. The module enables the admin to configure email templates for different events and provides a function to send emails based on event triggers.

**2. Requirements**

**2.1 Functional Requirements**

* Admin can create, update, delete email templates for predefined events.
* Admin can specify:
  + Email Subject
  + Email Body (HTML/Text)
  + Attachments (optional)
  + Associated Event (from a predefined list: ['On Purchase', 'On Cancel'])
* Store email templates in the database.
* Expose a backend function sendEmail(eventName, params, recipient) that:
  + Fetches the corresponding email template from the database.
  + Populates dynamic placeholders with data from params.
  + Sends an email to the recipient.
* Ability to trigger emails from both frontend and backend.

**2.2 Non-Functional Requirements**

* Secure API endpoints for email management.
* Scalable architecture for handling large volumes of emails.
* Logging for email delivery status.

**3. System Design**

**3.1 Architecture**

* **Frontend (React.js + Redux)**: Provides an admin interface for managing email templates.
* **Backend (Node.js + Express + MongoDB)**: Stores email templates and processes email requests.
* **Email Service (Nodemailer + SMTP/Third-party API)**: Handles email delivery.

**3.2 Database Schema (MongoDB)**

**email\_templates Collection**

{

"\_id": ObjectId,

"eventName": "On Purchase",

"subject": "Your Purchase Confirmation",

"body": "Hello {{name}}, thank you for purchasing!",

"attachments": ["invoice.pdf"],

"createdAt": ISODate,

"updatedAt": ISODate

}

**4. API Design**

**4.1 Admin Endpoints**

**Create Email Template**

**POST /api/email-templates**

{

"eventName": "On Purchase",

"subject": "Your Purchase Confirmation",

"body": "Hello {{name}}, thank you for purchasing!",

"attachments": ["invoice.pdf"]

}

**Fetch Templates**

**GET /api/email-templates**

**4.2 Email Sending Function**

**sendEmail(eventName, params, recipient)**

const sendEmail = async (eventName, params, recipient) => {

const template = await EmailTemplate.findOne({ eventName });

if (!template) throw new Error("Template not found");

let emailBody = template.body;

for (const key in params) {

emailBody = emailBody.replace(`{{${key}}}`, params[key]);

}

await emailService.send({

to: recipient,

subject: template.subject,

body: emailBody,

attachments: template.attachments

});

};

**5. Frontend Component (React.js)**

* **Admin UI:**
  + Form to create/update email templates.
  + List of existing templates.
  + Option to preview email before saving.

**6. Conclusion**

This module provides a structured approach for sending emails on predefined events in a MERN-based application. It ensures flexibility, scalability, and easy management for the admin.