COLLEGE CODE: 8203

COLLEGE: AVC COLLEGE OF ENGINEERING

DEPARTMENT: INFORMATION TECHNOLOGY

STUDENT NM-ID:164F4C9B531C69F8C41498DE7B1C57F0

ROLL NO: 23IT64

DATE:22/09/2025

Completed the project named as Phase 3

TECHNOLOGY PROJECT NAME: Email Remainder System

SUBMITTED BY,

NAME: LOGESH R

MOBILE NO: 9344415970

MVP Implementation

Project Setup

The first step is to establish the foundation and connect all the required services.

Step	Description	Tools Involved
1. Project Initialization & Dependencies	Initialize the Node.js project and install the necessary packages for the backend and email service.	npm init -y, npm install express mongoose nodemailer node-cron dotenv
2. Server and Environment Setup	Create server.js to initialize the **Express.js** application. Set up a .env file to securely store sensitive credentials like the MongoDB URI and Nodemailer SMTP details (host, port, user, password).	**Node.js, Express, dotenv**
3. Database Connection	Use **Mongoose** (the MongoDB ODM) in db.js to establish a connection to the MongoDB instance using the URI from the .env file.	**MongoDB, Mongoose**

Data Storage (Database)

Define the structure to store and manage the reminder data in **MongoDB**.

1. **Mongoose Schemas:**
    - **Reminder Schema:** Define the model to store reminder details and scheduling information.
        - recipientEmail (String, Required)
        - subject (String, Required)
        - body (String, Required)

        - cronString (String, Required): The expression for **node-cron** to trigger the job.
        - isActive (Boolean, Default: true)
        - lastSentAt (Date, No)
    - **Log Schema:** Define a separate model to track email delivery attempts.
        - reminderId (Reference to Reminder Schema)
        - timestamp (Date, Required)
        - status (String, Required: Success or Failure)
        - message (String, Error details or success confirmation)

Core Features Implementation

This involves implementing the logic for the three main system components: API, Scheduler, and Email Sender.

1. **REST API Endpoint (Express):**
    - Create a POST /api/reminders endpoint in the Express router.
    - **Input Validation:** Ensure the request body contains all required fields (recipientEmail, subject, body, cronString) and validates the email format.
    - **Data Persistence:** Upon valid input, use Mongoose to save the new reminder document to **MongoDB**.
2. **Email Sending Service (Nodemailer):**
    - Create a dedicated function (e.g., sendEmail(recipient, subject, body)) that initializes and uses **Nodemailer** to dispatch the email.
    - This function must handle the success and failure states of the email transmission.
3. **Scheduling Service (Node-cron):**
    - Create a startup script (e.g., initScheduler.js) that is called when the server starts.
    - **Initial Load:** Query **MongoDB** for all reminders where isActive: true.
    - **Job Creation:** For each reminder, use node-cron.schedule(reminder.cronString, callback).

- **Callback Function:** The cron job's callback function will:
    - Call the **Nodemailer** service function (`sendEmail`).
    - Record the result in the **Log Schema** in MongoDB (success or failure message).

Testing Core Features

Thorough testing is required to verify the integration between the scheduling, database, and email components.

1. **API Functionality Test:**
    - Use **Postman** to schedule a reminder with a cron string set for a time 2-3 minutes in the future (e.g., `* * * * *` for every minute for a quick test).
    - Verify the new entry appears in the MongoDB reminders collection.
2. **Scheduling and Delivery Test:**
    - Observe the server logs to confirm that **node-cron** successfully triggers the job at the scheduled time.
    - Check the recipient's inbox to confirm the email was delivered with the correct content.
3. **Logging Test:**
    - Verify that an entry is created in the MongoDB log collection showing `status: Success` after successful delivery.
    - Modify the email credentials to intentionally cause a failure, then test the scheduling again to verify a `status: Failure` log entry is created.

Version Control (GitHub)

Maintain project integrity and allow for collaboration.

Repository : https://github.com/LogeshR2005/NM-PROJECT