

Step 1: Install Node.js and Nest.js CLI

Install Node.js:

First, check if Node.js is installed by running:

- node -v

If Node.js isn't installed, you can download and install it from the official Node.js website.

Install Nest.js CLI:

With Node.js installed, you can now install the Nest.js CLI globally:

- `npm install -g @nestjs/cli`

Step 2: Create a New Nest.js Project

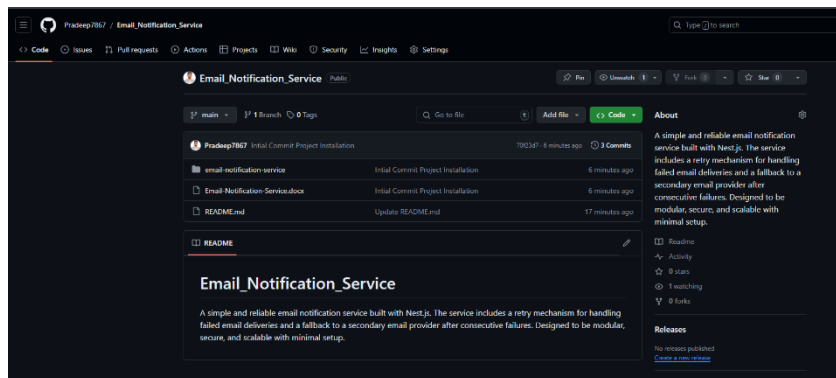
Create the Project:

Open a terminal in VS Code and run the following command to create a new Nest.js project:

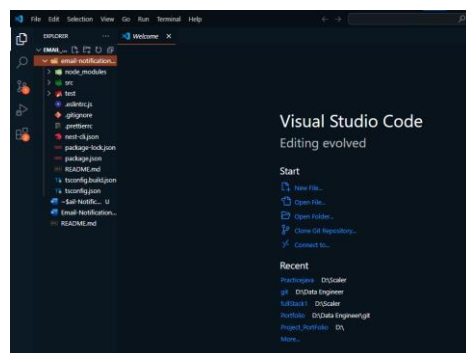
- nest new email-notification-service

Nest.js will prompt you to choose a package manager (either npm or yarn). Choose whichever you're comfortable with (let's assume npm for simplicity).

Step 3: Initialize Git and Create a GitHub Repository



Step 4: Open the Project in VS Code



Step 5: Now Run the Application

Step 6: Key Takeaways from the Skeleton Code:

Project Structure:

The structure outlined in the skeleton code aligns with what we discussed earlier but adds more modularity with channels like email, SMS, and push notifications.

We'll focus on the email channel and ensure it's functioning with basic retry and fallback logic.

Notifme SDK:

The code suggests using the Notifme SDK for handling multiple notification channels. We'll start by setting this up for email notifications.

AWS SQS Integration:

There's mention of using AWS SQS for handling queued jobs, which is great for scalability. However, for simplicity, we can skip this initially and add it later if time permits.

Step 7: Simplify the Project Scope:

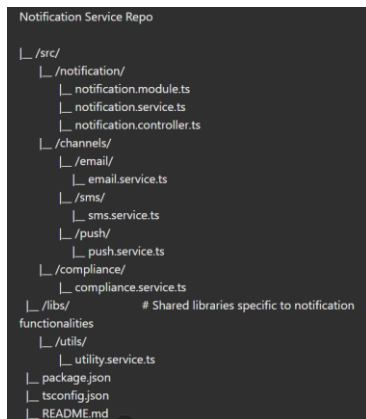
We'll focus on creating the notification service with basic email sending, retry logic, and a fallback mechanism.

Set Up Notifme SDK:

Implement the email sending service using Notifme SDK.

We'll focus solely on the email channel to keep things simple.

NOW:

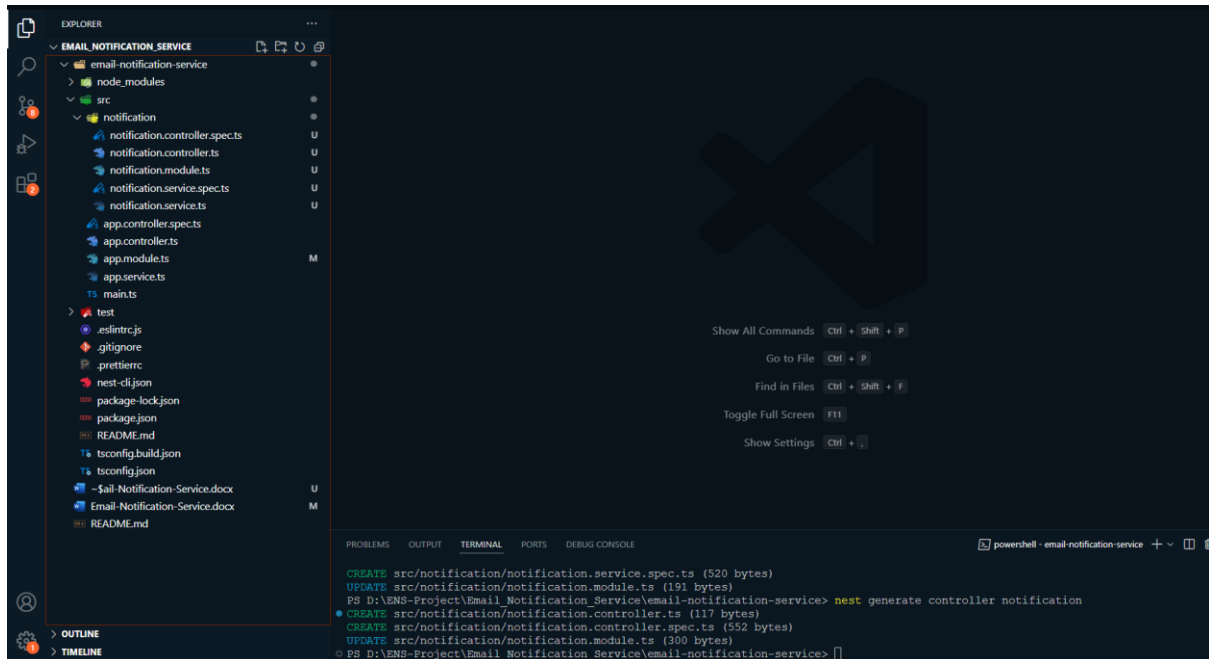


```
Notification Service Repo
├── /src/
│   ├── /notification/
│   │   ├── notification.module.ts
│   │   ├── notification.service.ts
│   │   └── notification.controller.ts
│   ├── /channels/
│   │   ├── /email/
│   │   │   └── email.service.ts
│   │   ├── /sms/
│   │   │   └── sms.service.ts
│   │   └── /push/
│   │       └── push.service.ts
│   ├── /compliance/
│   │   └── compliance.service.ts
│   └── /libs/ # Shared libraries specific to notification
├── functionalities
│   └── /utils/
│       └── utility.service.ts
├── package.json
├── tsconfig.json
└── README.md
```

Step 8: Set Up the Notification Module

Step 9: Set up the Notification Service

Step 10: Set up the Notification Controller

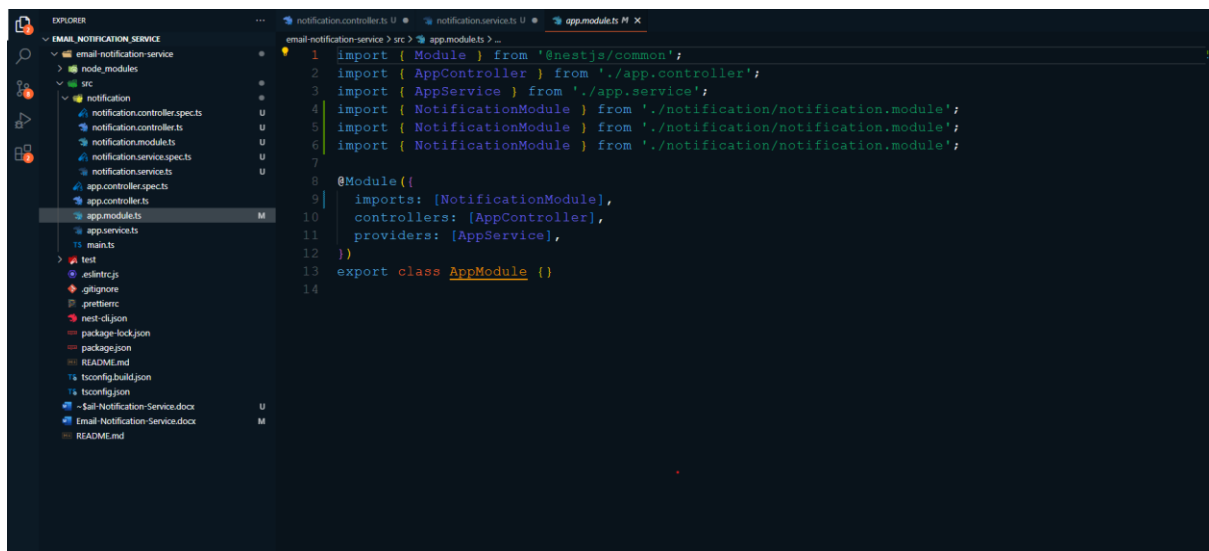


Now, let's add some basic implementation in the `notification.controller.ts` to set up a route that handles the incoming notification requests.

Step 12: Implement the `sendNotification` Method in the 'NotificationService'

Step 13: Update the AppModule

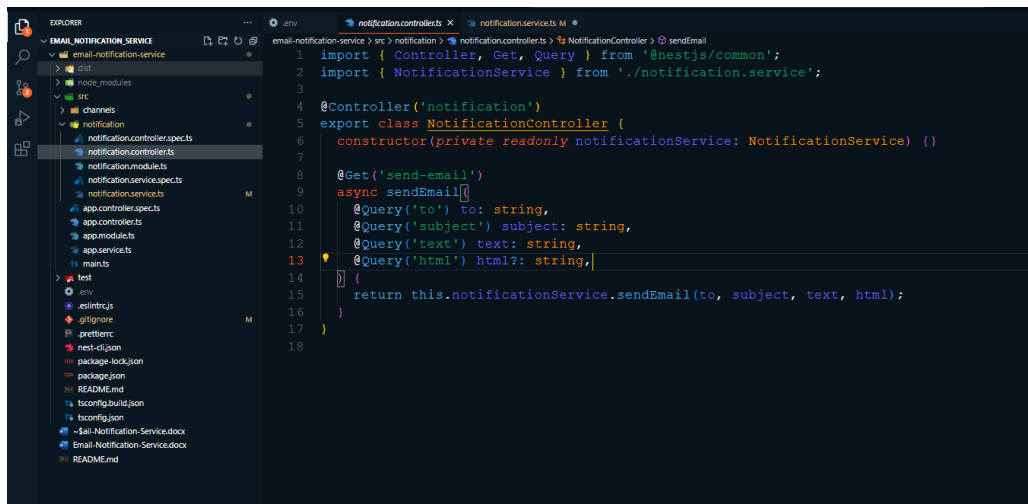
Finally, make sure the `NotificationModule` is imported and added to the `AppModule`:



Step 14 : Step 4: Implement the NotificationController

Now, let's add some basic implementation in the `notification.controller.ts` to set up a route that handles the incoming notification requests.

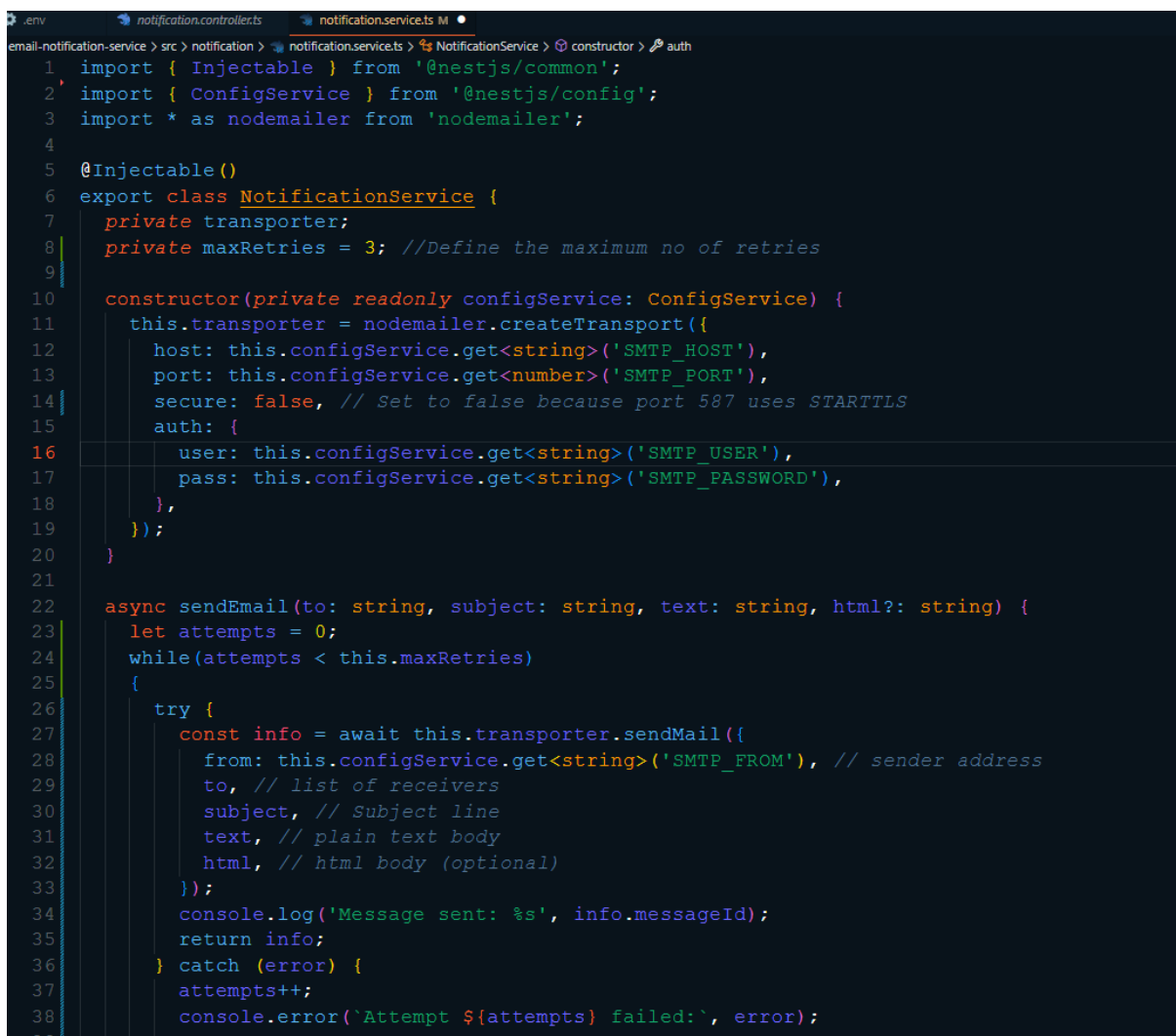
Open the `notification.controller.ts` file.



Step 15 : Step 5: Implement the sendNotification Method in the NotificationService

Open the notification.service.ts file.

Add the sendNotification method:



Step 16: Update the AppModule

Finally, make sure the NotificationModule is imported and added to the AppModule:

Open the app.module.ts file.

Update the imports and module configuration:

```
.env notification.service.ts M app.module.ts X
email-notification-service > src > app.module.ts > ...
1 import { Module } from '@nestjs/common';
2 import { AppController } from './app.controller';
3 import { AppService } from './app.service';
4 import { NotificationModule } from './notification/notification.module';
5 import { ConfigModule } from '@nestjs/config';
6
7 @Module({
8   imports: [
9     ConfigModule.forRoot({
10       isGlobal: true, // Makes the ConfigModule available globally
11     }),
12     NotificationModule,
13   ],
14   controllers: [AppController],
15   providers: [AppService],
16 })
17
18 export class AppModule {
19
20 }
21
```

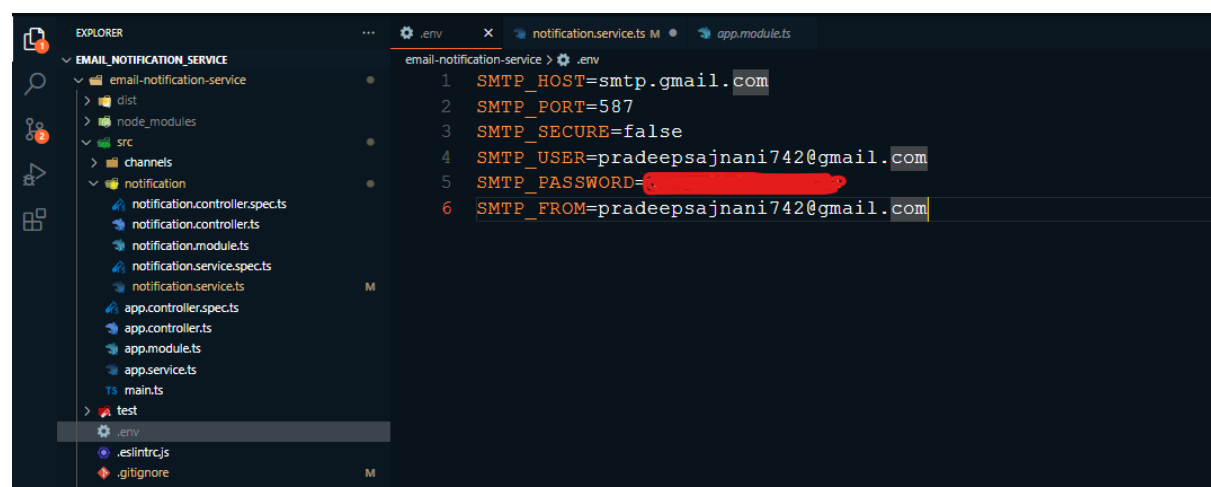
Now Imp Step 17: Step

Create the .env file:

In VS Code, right-click on the root directory of your project (EMAIL_NOTIFICATION_SERVICE) in the Explorer pane.

Select New File.

Name the file .env.



```
EXPLORER
EMAIL_NOTIFICATION_SERVICE
├── email-notification-service
│   ├── dist
│   ├── node_modules
│   ├── src
│   ├── channels
│   └── notification
│       ├── notification.controllers.spect
│       ├── notification.controllers
│       ├── notification.module.ts
│       ├── notification.service.spect
│       └── notification.service.ts
├── app.controllers.spect
├── app.controllers
├── app.module.ts
├── app.service.ts
└── TS main.ts
> test
.env
.eslintrc.js
.gitignore
prettierrc

.env
email-notification-service > .env
1 SMTP_HOST=smtp.gmail.com
2 SMTP_PORT=587
3 SMTP_SECURE=false
4 SMTP_USER=pradeepsajnani742@gmail.com
5 SMTP_PASSWORD=
6 SMTP_FROM=pradeepsajnani742@gmail.com
```

Step 18: Save and Test

Save the file after making these changes.

Restart your application using the command:

- `npm run start:dev`

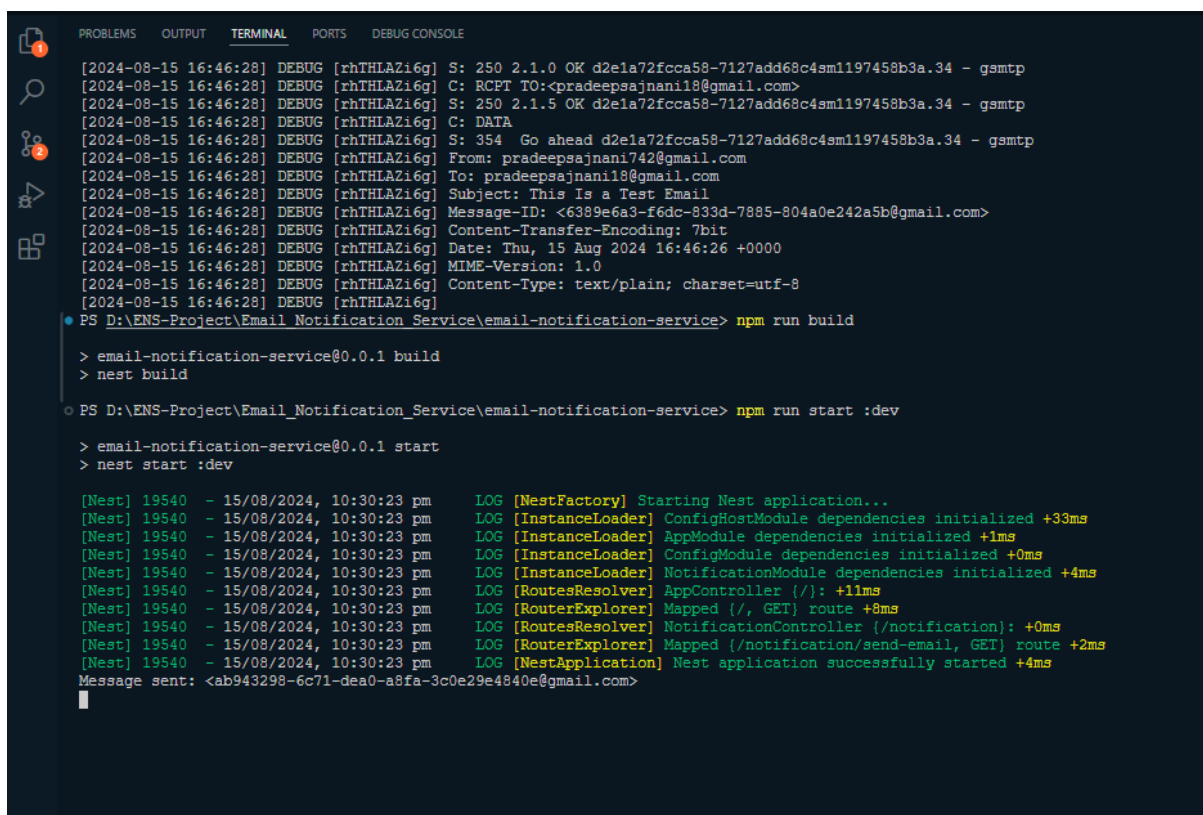
Try sending an email again using Postman.

What to Expect:

The system will retry sending the email up to three times if it fails.

If all attempts fail, it will switch to the backup EmailService to try sending the email.

Let me know once you've done this, and we'll proceed to the next step!



```
PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE
[2024-08-15 16:46:28] DEBUG [rhTHLAZi6g] S: 250 2.1.0 OK d2e1a72fccca58-7127add68c4sm1197458b3a.34 - gsmt
[2024-08-15 16:46:28] DEBUG [rhTHLAZi6g] C: RCPT TO:<pradeepsajnnani18@gmail.com>
[2024-08-15 16:46:28] DEBUG [rhTHLAZi6g] S: 250 2.1.5 OK d2e1a72fccca58-7127add68c4sm1197458b3a.34 - gsmt
[2024-08-15 16:46:28] DEBUG [rhTHLAZi6g] C: DATA
[2024-08-15 16:46:28] DEBUG [rhTHLAZi6g] S: 354 Go ahead d2e1a72fccca58-7127add68c4sm1197458b3a.34 - gsmt
[2024-08-15 16:46:28] DEBUG [rhTHLAZi6g] From: pradeepsajnnani742@gmail.com
[2024-08-15 16:46:28] DEBUG [rhTHLAZi6g] To: pradeepsajnnani18@gmail.com
[2024-08-15 16:46:28] DEBUG [rhTHLAZi6g] Subject: This Is a Test Email
[2024-08-15 16:46:28] DEBUG [rhTHLAZi6g] Message-ID: <6389e6a3-f6dc-833d-7885-804a0e242a5b@gmail.com>
[2024-08-15 16:46:28] DEBUG [rhTHLAZi6g] Content-Transfer-Encoding: 7bit
[2024-08-15 16:46:28] DEBUG [rhTHLAZi6g] Date: Thu, 15 Aug 2024 16:46:26 +0000
[2024-08-15 16:46:28] DEBUG [rhTHLAZi6g] MIME-Version: 1.0
[2024-08-15 16:46:28] DEBUG [rhTHLAZi6g] Content-Type: text/plain; charset=utf-8
[2024-08-15 16:46:28] DEBUG [rhTHLAZi6g]
PS D:\ENS-Project\Email Notification Service\email-notification-service> npm run build

> email-notification-service@0.0.1 build
> nest build

PS D:\ENS-Project\Email Notification Service\email-notification-service> npm run start :dev

> email-notification-service@0.0.1 start
> nest start :dev

[Nest] 19540 - 15/08/2024, 10:30:23 pm LOG [NestFactory] Starting Nest application...
[Nest] 19540 - 15/08/2024, 10:30:23 pm LOG [InstanceLoader] ConfigHostModule dependencies initialized +33ms
[Nest] 19540 - 15/08/2024, 10:30:23 pm LOG [InstanceLoader] AppModule dependencies initialized +1ms
[Nest] 19540 - 15/08/2024, 10:30:23 pm LOG [InstanceLoader] ConfigModule dependencies initialized +0ms
[Nest] 19540 - 15/08/2024, 10:30:23 pm LOG [InstanceLoader] NotificationModule dependencies initialized +4ms
[Nest] 19540 - 15/08/2024, 10:30:23 pm LOG [RoutesResolver] AppController (/): +11ms
[Nest] 19540 - 15/08/2024, 10:30:23 pm LOG [RouterExplorer] Mapped {/, GET} route +8ms
[Nest] 19540 - 15/08/2024, 10:30:23 pm LOG [RoutesResolver] NotificationController (/notification): +0ms
[Nest] 19540 - 15/08/2024, 10:30:23 pm LOG [RouterExplorer] Mapped {/notification/send-email, GET} route +2ms
[Nest] 19540 - 15/08/2024, 10:30:23 pm LOG [NestApplication] Nest application successfully started +4ms
Message sent: <ab943298-6c71-dea0-a8fa-3c0e29e4840e@gmail.com>
```

it's a Success so Now We Will Implementing the Retry Logic

We'll modify the sendEmail method to retry sending the email up to three times. If all attempts fail, we'll switch to the fallback EmailService.

Here's the updated code for your NotificationService:

```
7 export class NotificationService {
25
26   async sendEmail(to: string, subject: string, text: string, html?: string) {
27     let attempts = 0;
28     const maxRetries = 3; //Define the maximum no of retries
29     while(attempts < maxRetries)
30     {
31       try {
32         const info = await this.transporter.sendMail({
33           from: this.configService.get<string>('SMTP_FROM'), // sender address
34           to, // list of receivers
35           subject, // Subject line
36           text, // plain text body
37           html, // html body (optional)
38         });
39         console.log('Message sent: %s', info.messageId);
40         return info;
41       } catch (error) {
42         attempts++;
43         console.error('Attempt ${attempts} failed:', error);
44       }
45       if (attempts >= maxRetries) {
46         console.log('Switching to backup service...');
47         return this.emailService.sendMail(to, subject, html || text); // Use the backup service
48       }
49     }
50   }
51
52   private async sendViaFallbackService(to: string, subject: string, text: string, html?: string) {
53     try {
54       await this.emailService.sendMail(to, subject, html || text);
55       console.log('Fallback service sent the email successfully');
56     } catch (error) {
57       console.error('Fallback service failed to send the email:', error);
58       throw error;
59     }
60   }
61 }
```

Now Our Logic is

- Retry Loop: We use a while loop to retry sending the email up to three times.
- Fallback Mechanism: If all three attempts fail, the sendViaFallbackService method is called, which uses the EmailService to send the email.
- Error Handling: If both the main service and fallback fail, the error is logged, and the exception is thrown.

Next Steps:

- Save this updated NotificationService code.
- Run your application and test it by intentionally causing the primary email service to fail (e.g., by using incorrect SMTP credentials).
- Verify that the fallback service is used after three failed attempts.

```
PROBLEMS  OUTPUT  TERMINAL  PORTS  DEBUG CONSOLE

PS D:\ENS-Project\Email_Notification_Service(email-notification-service) npm run start :dev

> email-notification-service@0.0.1 start
> nest start :dev

[Nest] 9076 - 15/08/2024, 10:57:38 pm LOG [NestFactory] Starting Nest application...
[Nest] 9076 - 15/08/2024, 10:57:38 pm LOG [InstanceLoader] ConfigModule dependencies initialized +16ms
[Nest] 9076 - 15/08/2024, 10:57:38 pm LOG [InstanceLoader] AppModule dependencies initialized +0ms
[Nest] 9076 - 15/08/2024, 10:57:38 pm LOG [InstanceLoader] ConfigModule dependencies initialized +0ms
[Nest] 9076 - 15/08/2024, 10:57:38 pm LOG [InstanceLoader] NotificationModule dependencies initialized +2ms
[Nest] 9076 - 15/08/2024, 10:57:38 pm LOG [RoutesResolver] AppController (/): +5ms
[Nest] 9076 - 15/08/2024, 10:57:38 pm LOG [RouterExplorer] Mapped {/, GET} route +3ms
[Nest] 9076 - 15/08/2024, 10:57:38 pm LOG [RoutesResolver] NotificationController (/notification): +0ms
[Nest] 9076 - 15/08/2024, 10:57:38 pm LOG [RouterExplorer] Mapped {/notification/send-email, GET} route +1ms
[Nest] 9076 - 15/08/2024, 10:57:38 pm LOG [NestApplication] Nest application successfully started +2ms
Attempt 1 failed: Error: Invalid login: 535-5.7.8 Username and Password not accepted. For more information, go to
535 5.7.8 https://support.google.com/mail/?p=BadCredentials d2e1a72fcca58-7127add7cbcsml273824b3a.14 - gsmt
at SMTPConnection.formatError (D:\ENS-Project\Email_Notification_Service(email-notification-service\node_modules\nodemailer\lib\smtp-connection\index.js:798:19)
at SMTPConnection.actionAUTHComplete (D:\ENS-Project\Email_Notification_Service(email-notification-service\node_modules\nodemailer\lib\smtp-connection\index.js:1577:34)
at SMTPConnection.actionAUTHComplete (D:\ENS-Project\Email_Notification_Service(email-notification-service\node_modules\nodemailer\lib\smtp-connection\index.js:554:26)
at SMTPConnection.processResponse (D:\ENS-Project\Email_Notification_Service(email-notification-service\node_modules\nodemailer\lib\smtp-connection\index.js:982:20)
at SMTPConnection.onData (D:\ENS-Project\Email_Notification_Service(email-notification-service\node_modules\nodemailer\lib\smtp-connection\index.js:763:14)
at SMTPConnection.onSocketData (D:\ENS-Project\Email_Notification_Service(email-notification-service\node_modules\nodemailer\lib\smtp-connection\index.js:992:20)
at TLSocket.emit (node:events:519:28)
at addChunk (node:internal/streams/readable:559:12)
at readableAddChunkPushByteMode (node:internal/streams/readable:510:3)
at Readable.push (node:internal/streams/readable:390:5) {
  code: 'EAUTH',
  response: '535-5.7.8 Username and Password not accepted. For more information, go to\n' +
    '535 5.7.8 https://support.google.com/mail/?p=BadCredentials d2e1a72fcca58-7127add7cbcsml273824b3a.14 - gsmt',
  responseCode: 535,
  command: 'AUTH PLAIN'
}
Attempt 2 failed: Error: Invalid login: 535-5.7.8 Username and Password not accepted. For more information, go to
535 5.7.8 https://support.google.com/mail/?p=BadCredentials d2e1a72fcca58-7127add7cbcsml230037b3a.57 - gsmt
at SMTPConnection.formatError (D:\ENS-Project\Email_Notification_Service(email-notification-service\node_modules\nodemailer\lib\smtp-connection\index.js:798:19)
at SMTPConnection.actionAUTHComplete (D:\ENS-Project\Email_Notification_Service(email-notification-service\node_modules\nodemailer\lib\smtp-connection\index.js:1577:34)
at SMTPConnection.actionAUTHComplete (D:\ENS-Project\Email_Notification_Service(email-notification-service\node_modules\nodemailer\lib\smtp-connection\index.js:554:26)
at SMTPConnection.processResponse (D:\ENS-Project\Email_Notification_Service(email-notification-service\node_modules\nodemailer\lib\smtp-connection\index.js:982:20)
at SMTPConnection.onData (D:\ENS-Project\Email_Notification_Service(email-notification-service\node_modules\nodemailer\lib\smtp-connection\index.js:763:14)
at SMTPConnection.onSocketData (D:\ENS-Project\Email_Notification_Service(email-notification-service\node_modules\nodemailer\lib\smtp-connection\index.js:992:20)
at TLSocket.emit (node:events:519:28)
at addChunk (node:internal/streams/readable:559:12)
at readableAddChunkPushByteMode (node:internal/streams/readable:510:3)
at Readable.push (node:internal/streams/readable:390:5) {
  code: 'EAUTH',
  response: '535-5.7.8 Username and Password not accepted. For more information, go to\n' +
    '535 5.7.8 https://support.google.com/mail/?p=BadCredentials d2e1a72fcca58-7127add7cbcsml230037b3a.57 - gsmt',
  responseCode: 535,
  command: 'AUTH PLAIN'
}
Attempt 3 failed: Error: Invalid login: 535-5.7.8 Username and Password not accepted. For more information, go to
535 5.7.8 https://support.google.com/mail/?p=BadCredentials 41be03b00d2f7-7c6b61a6ce4sm1378586a12.15 - gsmt
at SMTPConnection.formatError (D:\ENS-Project\Email_Notification_Service(email-notification-service\node_modules\nodemailer\lib\smtp-connection\index.js:798:19)
at SMTPConnection.actionAUTHComplete (D:\ENS-Project\Email_Notification_Service(email-notification-service\node_modules\nodemailer\lib\smtp-connection\index.js:1577:34)
at SMTPConnection.actionAUTHComplete (D:\ENS-Project\Email_Notification_Service(email-notification-service\node_modules\nodemailer\lib\smtp-connection\index.js:554:26)
at SMTPConnection.processResponse (D:\ENS-Project\Email_Notification_Service(email-notification-service\node_modules\nodemailer\lib\smtp-connection\index.js:982:20)
at SMTPConnection.onData (D:\ENS-Project\Email_Notification_Service(email-notification-service\node_modules\nodemailer\lib\smtp-connection\index.js:763:14)
at SMTPConnection.onSocketData (D:\ENS-Project\Email_Notification_Service(email-notification-service\node_modules\nodemailer\lib\smtp-connection\index.js:992:20)
at TLSocket.emit (node:events:519:28)
at addChunk (node:internal/streams/readable:559:12)
at readableAddChunkPushByteMode (node:internal/streams/readable:510:3)
at Readable.push (node:internal/streams/readable:390:5) {
  code: 'EAUTH',
  response: '535-5.7.8 Username and Password not accepted. For more information, go to\n' +
    '535 5.7.8 https://support.google.com/mail/?p=BadCredentials 41be03b00d2f7-7c6b61a6ce4sm1378586a12.15 - gsmt',
  responseCode: 535,
  command: 'AUTH PLAIN'
}
Switching to backup service...
Failed to send email to pradeepsajnanil8@gmail.com Error: Invalid login: 535-5.7.8 Username and Password not accepted. For more information, go to
535 5.7.8 https://support.google.com/mail/?p=BadCredentials 41be03b00d2f7-7c6b61a6ce4sm1378586a12.15 - gsmt
at SMTPConnection.formatError (D:\ENS-Project\Email_Notification_Service(email-notification-service\node_modules\nodemailer\lib\smtp-connection\index.js:798:19)
at SMTPConnection.actionAUTHComplete (D:\ENS-Project\Email_Notification_Service(email-notification-service\node_modules\nodemailer\lib\smtp-connection\index.js:1577:34)
at SMTPConnection.actionAUTHComplete (D:\ENS-Project\Email_Notification_Service(email-notification-service\node_modules\nodemailer\lib\smtp-connection\index.js:554:26)
at SMTPConnection.processResponse (D:\ENS-Project\Email_Notification_Service(email-notification-service\node_modules\nodemailer\lib\smtp-connection\index.js:982:20)
at SMTPConnection.onData (D:\ENS-Project\Email_Notification_Service(email-notification-service\node_modules\nodemailer\lib\smtp-connection\index.js:763:14)
at SMTPConnection.onSocketData (D:\ENS-Project\Email_Notification_Service(email-notification-service\node_modules\nodemailer\lib\smtp-connection\index.js:992:20)
at TLSocket.emit (node:events:519:28)
at addChunk (node:internal/streams/readable:559:12)
at readableAddChunkPushByteMode (node:internal/streams/readable:510:3)
at Readable.push (node:internal/streams/readable:390:5) {
  code: 'EAUTH',
  response: '535-5.7.8 Username and Password not accepted. For more information, go to\n' +
    '535 5.7.8 https://support.google.com/mail/?p=BadCredentials 41be03b00d2f7-7c6b61a6ce4sm1378586a12.15 - gsmt',
  responseCode: 535,
  command: 'AUTH PLAIN'
}
[Nest] 9076 - 15/08/2024, 10:58:06 pm ERROR [ExceptionsHandler] Invalid login: 535-5.7.8 Username and Password not accepted. For more information, go to
535 5.7.8 https://support.google.com/mail/?p=BadCredentials 41be03b00d2f7-7c6b61a6ce4sm1378586a12.15 - gsmt
Error: Invalid login: 535-5.7.8 Username and Password not accepted. For more information, go to
535 5.7.8 https://support.google.com/mail/?p=BadCredentials 41be03b00d2f7-7c6b61a6ce4sm1378586a12.15 - gsmt
at SMTPConnection.formatError (D:\ENS-Project\Email_Notification_Service(email-notification-service\node_modules\nodemailer\lib\smtp-connection\index.js:798:19)
at SMTPConnection.actionAUTHComplete (D:\ENS-Project\Email_Notification_Service(email-notification-service\node_modules\nodemailer\lib\smtp-connection\index.js:1577:34)
at SMTPConnection.actionAUTHComplete (D:\ENS-Project\Email_Notification_Service(email-notification-service\node_modules\nodemailer\lib\smtp-connection\index.js:554:26)
at SMTPConnection.processResponse (D:\ENS-Project\Email_Notification_Service(email-notification-service\node_modules\nodemailer\lib\smtp-connection\index.js:982:20)
at SMTPConnection.onData (D:\ENS-Project\Email_Notification_Service(email-notification-service\node_modules\nodemailer\lib\smtp-connection\index.js:763:14)
at SMTPConnection.onSocketData (D:\ENS-Project\Email_Notification_Service(email-notification-service\node_modules\nodemailer\lib\smtp-connection\index.js:992:20)
at TLSocket.emit (node:events:519:28)
at addChunk (node:internal/streams/readable:559:12)
at readableAddChunkPushByteMode (node:internal/streams/readable:510:3)
at Readable.push (node:internal/streams/readable:390:5)
}
```

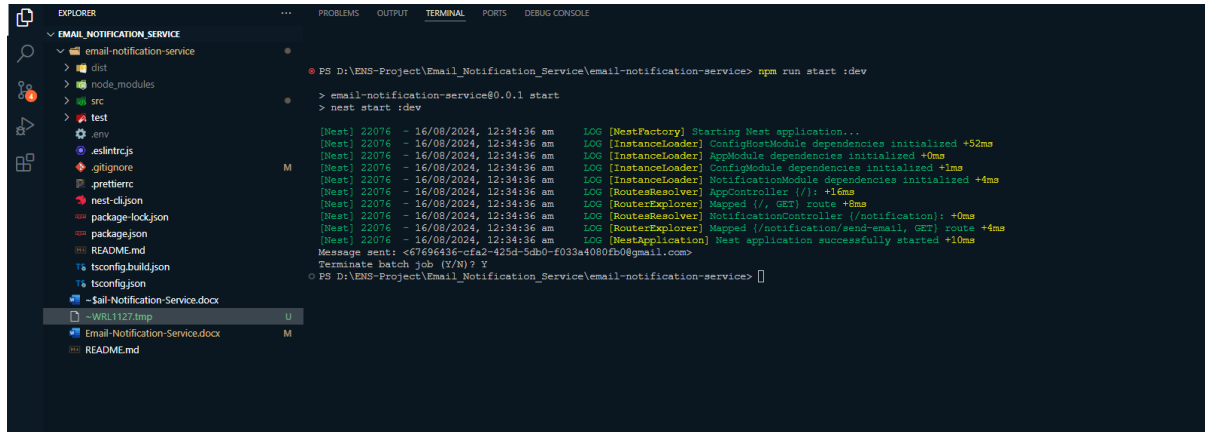
That's All it looks like we've covered everything needed for your email notification service:

Retry Mechanism: Implemented and tested with a fallback to a secondary service.

Next Step

Restore Correct SMTP Credentials: Update the .env file with the correct SMTP_USER and SMTP_PASSWORD values to ensure your primary email service works properly again.

Final Testing:



The screenshot shows a VS Code interface with a file explorer on the left and a terminal window on the right. The file explorer shows a project structure for 'EMAIL_NOTIFICATION_SERVICE' with files like .env, .eslintrc.js, .gitignore, .prettierrc, nest-cli.json, package-lock.json, package.json, README.md, tsconfig.build.json, and tsconfig.json. The terminal window shows the command 'npm run start:dev' being executed, followed by the command 'email-notification-service@0.0.1 start' and 'nest start:dev'. The terminal output shows the NestJS application starting successfully, with logs indicating that the application is running on port 3000 and that the email service is working correctly.

After restoring the correct credentials, test the email sending again to confirm that everything is functioning as expected with the primary service.

Security: Ensured secure handling of environment variables and added .env to .gitignore.

Basic Functionality: Successfully sent emails and handled errors appropriately

Thank You

If Any Doubt feel free to reach out..

Pradeep Sajjani

pradeepsajjani742@gmail.com