📌 Feature: Email Notifications for Overdue Tasks

We will modify the system to:  
✅ Automatically **send an email** when a task becomes overdue  
✅ Use **JavaMailSender** for sending emails  
✅ Schedule email alerts **every 30 seconds**

**📌 Step 1: Add dependency in pom.xml**

<dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-mail</artifactId>

            <version>3.4.3</version>

        </dependency>

**📌 Step 2: Add Email Configuration in application.properties**

spring.mail.host=smtp.gmail.com

spring.mail.port=587

spring.mail.username=abc@gmail.com

spring.mail.password=your app password

spring.mail.properties.mail.smtp.auth=true

spring.mail.properties.mail.smtp.starttls.enable=true

spring.mail.properties.mail.smtp.starttls.required=true

spring.mail.properties.mail.smtp.ssl.trust=smtp.gmail.com

🔹 **Use App Password** instead of your email password for security.

**📌 Step 3: Create EmailService for Sending Emails**

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.mail.SimpleMailMessage;

import org.springframework.mail.javamail.JavaMailSender;

import org.springframework.stereotype.Service;

@Service

public class EmailService {

@Autowired

private JavaMailSender mailSender;

public void sendEmail(String to, String subject, String body) {

SimpleMailMessage message = new SimpleMailMessage();

message.setTo(to);

message.setSubject(subject);

message.setText(body);

mailSender.send(message);

}

}

**📌 Step 4: Modify Employee Class to Store Email**

public class Employee {

private final String employeeId;

private final String name;

private final String email; // New field for email

public Employee(String name, String email) {

this.employeeId = UUID.randomUUID().toString();

this.name = name;

this.email = email;

}

public String getEmail() {

return email;

}

}

**📌 Step 5: Modify EmployeeTaskManagerService for Overdue Email Alerts**

import org.springframework.scheduling.annotation.Scheduled;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import org.springframework.context.annotation.Configuration;

import org.springframework.scheduling.annotation.EnableScheduling;

@Service

@Configuration

@EnableScheduling // Enables @Scheduled execution

public class EmployeeTaskManagerService {

@Autowired

private EmailService emailService;

private final Map<String, Employee> employees = new HashMap<>();

private final Map<String, Task> assignedTasks = new HashMap<>();

// 🕒 Check overdue tasks every 30 seconds & send email

@Scheduled(fixedRate = 30000)

public void checkOverdueTasks() {

for (Task task : assignedTasks.values()) {

if (task.isOverdue()) {

Employee employee = employees.get(task.getAssignedTo());

String emailBody = "⚠️ Task Overdue: " + task.getTaskId() +

"\nPriority: " + task.getPriority() +

"\nDue Date: " + task.getDueDate();

emailService.sendEmail(employee.getEmail(), "Overdue Task Alert", emailBody);

System.out.println("📧 Email sent to " + employee.getEmail());

}

}

}

}

**📌 Step 6: Modify API to Register Employees with Emails**

@PostMapping("/register")

public Employee registerEmployee(@RequestParam String name, @RequestParam String email) {

return taskManagerService.registerEmployee(name, email);

}

**📌 Step 7: Testing the Feature**

**1️⃣ Register an Employee with Email**

POST http://localhost:8080/employees/register?name=Alice&email=alice@example.com

**2️⃣ Assign a Task with a Past Due Date**

POST http://localhost:8080/employees/12345/assign-task?priority=HIGH&dueDate=2024-02-01T12:00:00

**3️⃣ Wait 30 seconds & Check Email Inbox**

📧 **Email Received**:

Subject: Overdue Task Alert

Body:

⚠️ Task Overdue: task-67890

Priority: HIGH

Due Date: 2024-02-01T12:00:00

**📌 Feature Summary**

| **Feature** | **Implementation** |
| --- | --- |
| **Email Alerts** | JavaMailSender |
| **Check Overdue Tasks** | @Scheduled(fixedRate = 30000) |
| **Store Employee Emails** | Added email field in Employee |
| **Send Overdue Emails** | EmailService.sendEmail() |

**🚀 Next Steps:**

### ­­­✅ Steps to Generate an App Password in Gmail

Since Google blocks direct email-password authentication, you need to generate an **App Password** to use with Spring Boot’s JavaMailSender.

**🔹 Step 1: Enable 2-Step Verification**

1️⃣ Open Google Account Security.  
2️⃣ Scroll down to **"Signing in to Google"**.  
3️⃣ Click on **"2-Step Verification"** and follow the setup process.

**🔹 Step 2: Generate App Password**

1️⃣ Open **Google Account Security** → App Passwords Page.  
2️⃣ Select **Mail** from the drop-down list.  
3️⃣ Choose a device (e.g., **Windows PC**).  
4️⃣ Click **Generate**.  
5️⃣ Google will show a **16-character password** (e.g., abcd efgh ijkl mnop).  
6️⃣ **Copy and save it** (you won’t see it again).

**📌 Feature: SMS Reminders for Overdue Tasks**

Just like the email notifications, we'll send **SMS reminders** when a task is overdue.  
We will use **Twilio**, a popular SMS service provider. 🚀

### ****📌 Step 1: Create a Twilio Account & Get Credentials****

1️⃣ Sign up at [**Twilio**](https://www.twilio.com/try-twilio) (Free trial available).  
2️⃣ Verify your phone number.  
3️⃣ Go to **Console Dashboard** → Copy:

* **Account SID**
* **Auth Token**
* **Twilio Phone Number**

### ****📌 Step 2: Add Twilio Dependency****

In your **pom.xml** (for Maven users):

xml

<dependency>

<groupId>com.twilio.sdk</groupId>

<artifactId>twilio</artifactId>

<version>9.0.0</version>

</dependency>

### ****📌 Step 3: Add Twilio Configurations in**** application.properties

twilio.accountSid=your\_account\_sid

twilio.authToken=your\_auth\_token

twilio.phoneNumber=your\_twilio\_number

### ****📌 Step 4: Create**** SmsService ****to Send SMS****

import org.springframework.beans.factory.annotation.Value;

import org.springframework.stereotype.Service;

import com.twilio.Twilio;

import com.twilio.rest.api.v2010.account.Message;

@Service

public class SmsService {

@Value("${twilio.accountSid}")

private String accountSid;

@Value("${twilio.authToken}")

private String authToken;

@Value("${twilio.phoneNumber}")

private String twilioPhoneNumber;

public void sendSms(String to, String body) {

Twilio.init(accountSid, authToken);

Message message = Message.creator(

new com.twilio.type.PhoneNumber(to), // Recipient's phone number

new com.twilio.type.PhoneNumber(twilioPhoneNumber), // Twilio number

body

).create();

System.out.println("📲 SMS sent to: " + to);

}

}

### ****📌 Step 5: Modify**** Employee ****Class to Store Phone Number****

public class Employee {

private final String employeeId;

private final String name;

private final String email;

private final String phoneNumber; // New field for SMS

public Employee(String name, String email, String phoneNumber) {

this.employeeId = UUID.randomUUID().toString();

this.name = name;

this.email = email;

this.phoneNumber = phoneNumber;

}

public String getPhoneNumber() {

return phoneNumber;

}

}

### ****📌 Step 6: Modify**** EmployeeTaskManagerService ****to Send SMS for Overdue Tasks****

import org.springframework.scheduling.annotation.Scheduled;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

@Service

public class EmployeeTaskManagerService {

@Autowired

private EmailService emailService;

@Autowired

private SmsService smsService;

private final Map<String, Employee> employees = new HashMap<>();

private final Map<String, Task> assignedTasks = new HashMap<>();

// 🕒 Check overdue tasks every 30 seconds & send email/SMS

@Scheduled(fixedRate = 30000)

public void checkOverdueTasks() {

System.out.println("⏰ Checking for overdue tasks...");

for (Task task : assignedTasks.values()) {

if (task.isOverdue()) {

Employee employee = employees.get(task.getAssignedTo());

String alertMessage = "⚠️ Task Overdue: " + task.getTaskId() +

"\nPriority: " + task.getPriority() +

"\nDue Date: " + task.getDueDate();

// Send Email

emailService.sendEmail(employee.getEmail(), "Overdue Task Alert", alertMessage);

System.out.println("📧 Email sent to " + employee.getEmail());

// Send SMS

smsService.sendSms(employee.getPhoneNumber(), alertMessage);

System.out.println("📲 SMS sent to " + employee.getPhoneNumber());

}

}

}

}

### ****📌 Step 7: Modify API to Register Employees with Phone Numbers****

@PostMapping("/register")

public Employee registerEmployee(@RequestParam String name,

@RequestParam String email,

@RequestParam String phoneNumber) {

return taskManagerService.registerEmployee(name, email, phoneNumber);

}

### ****📌 Step 8: Testing the Feature****

✅ **Register an Employee with Phone Number**

POST http://localhost:8080/employees/register?name=Alice&email=alice@example.com&phoneNumber=%2B1234567890

✅ **Assign a Task with a Past Due Date**

POST http://localhost:8080/employees/12345/assign-task?priority=HIGH&dueDate=2024-02-01T12:00:00

✅ **Wait 30 seconds & Check SMS Inbox**  
📲 **SMS Received**:

⚠️ Task Overdue: task-67890

Priority: HIGH

Due Date: 2024-02-01T12:00:00

### ****📌 Feature Summary****

| **Feature** | **Implementation** |
| --- | --- |
| **Email Alerts** | JavaMailSender |
| **SMS Alerts** | Twilio API |
| **Check Overdue Tasks** | @Scheduled(fixedRate = 30000) |
| **Store Employee Contacts** | Added phoneNumber field in Employee |