📌 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 |

**📢 Step-by-Step Guide: Sending WhatsApp Notifications for Overdue Tasks in Spring Boot**

We will integrate **WhatsApp notifications** using **Meta's WhatsApp Cloud API**. Your application will now send **WhatsApp alerts** in addition to **Email & SMS** when a task is overdue.

## ****📌 Step 1: Set Up WhatsApp Cloud API****

Before writing code, you need to set up **WhatsApp Cloud API**:

### ****🔹 1.1: Create a Meta (Facebook) Developer Account****

1. **Go to** [Meta for Developers](https://developers.facebook.com/)
2. **Log in** with your Facebook account.
3. **Click on "Create App"** → Select **Business** type.
4. **Set App Name** and **Create the App**.

### ****🔹 1.2: Set Up WhatsApp Business API****

1. **In the app dashboard**, click **"Set up"** under **WhatsApp**.
2. **Create a WhatsApp Business Account** and verify your number.
3. **Get a "Test Phone Number"** from WhatsApp.
   * You can only send messages to the numbers **you verify**.
4. **Get the "Phone Number ID" and "Permanent Access Token".**
   * Copy them for later use.

### ****🔐 How to Get a Permanent Access Token for WhatsApp Cloud API****

By default, Meta provides a **temporary access token** that expires after a few hours. To use WhatsApp notifications permanently, you need a **long-lived access token**.

## ****📌 Step 1: Generate a Long-Lived Access Token****

Follow these steps to generate a permanent (long-lived) access token:

### ****🔹 1.1: Go to Meta Graph API Explorer****

1. **Open** [Meta Graph API Explorer](https://developers.facebook.com/tools/explorer/)
2. **Select Your App** (from the dropdown at the top)
3. **Click on "Get Token"** → Choose **"Get User Access Token"**

### ****🔹 1.2: Add Required Permissions****

When generating a token, you must include the required **permissions**:  
✔ whatsapp\_business\_messaging  
✔ whatsapp\_business\_management  
✔ pages\_show\_list

1. Click **"Edit Permissions"**
2. Enable the above **permissions**
3. Click **"Generate Access Token"**

🚨 **IMPORTANT:** The token generated here is still short-lived.

## ****📌 Step 2: Exchange for a Long-Lived Token****

Now, convert this short-lived token into a long-lived (60-day) token.

1. **Copy** the short-lived token from **Step 1.2**
2. Open a terminal or Postman
3. Execute the following **Graph API request**:

GET https://graph.facebook.com/v17.0/oauth/access\_token

?grant\_type=fb\_exchange\_token

&client\_id=YOUR\_APP\_ID

&client\_secret=YOUR\_APP\_SECRET

&fb\_exchange\_token=YOUR\_SHORT\_LIVED\_TOKEN

* Replace YOUR\_APP\_ID and YOUR\_APP\_SECRET from your **Meta App Dashboard**
* Replace YOUR\_SHORT\_LIVED\_TOKEN with the **token from Step 1.2**

✅ **Response:** You will receive a **long-lived (60-day) access token**.

## ****📌 Step 3: Get a Permanent (Never-Expiring) Token****

Meta allows **permanent access tokens** for business apps.

### ****🔹 3.1: Convert User Token to a System User Token****

To get a never-expiring token:

1. **Go to** [Meta Business Settings](https://business.facebook.com/settings/)
2. Navigate to **Business Settings > System Users**
3. **Create a System User**
4. Assign the **WhatsApp API permissions**
5. **Generate a System User Token** from there

This **system user token never expires** and can be used in your Spring Boot application.

## ****📌 Step 4: Use the Permanent Token in Spring Boot****

Once you get the **long-lived or permanent token**, update it in your application.properties:

whatsapp.token=YOUR\_PERMANENT\_ACCESS\_TOKEN

## ****📌 Step 2: Add WhatsApp Credentials in**** application.properties

Replace with your actual **WhatsApp API details**.

whatsapp.api.url=https://graph.facebook.com/v17.0/

whatsapp.phone.number.id=YOUR\_PHONE\_NUMBER\_ID

whatsapp.token=YOUR\_ACCESS\_TOKEN

## ****📌 Step 3: Create**** WhatsAppService ****to Send Messages****

This service will send a WhatsApp message to employees when a task is overdue.

📌 **File:** WhatsAppService.java

package com.example.whatsapp.service;

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

import org.springframework.http.\*;

import org.springframework.stereotype.Service;

import org.springframework.web.client.RestTemplate;

import java.util.Map;

@Service

public class WhatsAppService {

@Value("${whatsapp.api.url}")

private String apiUrl;

@Value("${whatsapp.phone.number.id}")

private String phoneNumberId;

@Value("${whatsapp.token}")

private String accessToken;

private final RestTemplate restTemplate = new RestTemplate();

public void sendWhatsAppMessage(String to, String taskId, String priority, String dueDate) {

String url = apiUrl + phoneNumberId + "/messages";

String message = "⚠️ Task Overdue: " + taskId +

"\nPriority: " + priority +

"\nDue Date: " + dueDate +

"\nPlease complete it ASAP.";

Map<String, Object> requestBody = Map.of(

"messaging\_product", "whatsapp",

"recipient\_type", "individual",

"to", to,

"type", "text",

"text", Map.of("body", message)

);

HttpHeaders headers = new HttpHeaders();

headers.setContentType(MediaType.APPLICATION\_JSON);

headers.setBearerAuth(accessToken);

HttpEntity<Map<String, Object>> requestEntity = new HttpEntity<>(requestBody, headers);

ResponseEntity<String> response = restTemplate.exchange(url, HttpMethod.POST, requestEntity, String.class);

System.out.println("📢 WhatsApp Response: " + response.getBody());

}

}

## ****📌 Step 4: Modify the**** checkOverdueTasks() ****Method****

Now, integrate **WhatsAppService** with the existing email and SMS notifications.

📌 **Modify in TaskSchedulerService.java**

@Scheduled(fixedRate = 5000) // Runs every 5 seconds

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

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

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

// Send Email

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

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

// Send SMS

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

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

// Send WhatsApp Notification

whatsappService.sendWhatsAppMessage(employee.getPhoneNumber(),

task.getTaskId(),

task.getPriority(),

task.getDueDate());

System.out.println("📢 WhatsApp Notification sent to " + employee.getPhoneNumber());

}

}

}

## ****📌 Step 5: Run & Test****

### ✅ ****Run Your Spring Boot Application****

1. Start the Spring Boot application.
2. Ensure your task data is in place.
3. Wait for the **scheduler (@Scheduled) to execute**.
4. Check your WhatsApp for the **overdue task alert**.

## ****📌 Step 6: Common Errors & Fixes****

| **❌ Error** | **✅ Solution** |
| --- | --- |
| Invalid 'To' Phone Number | Ensure the phone number is registered on WhatsApp and is in **E.164 format** (+91xxxxxxxxxx). |
| Account exceeded daily messages limit | Free tier allows **1,000 messages per month**. Upgrade plan if needed. |
| 403 Forbidden | Check if your **access token is valid** and you have **business verification**. |

## ****📌 FAQ****

### ****🔹 Q: Can I send messages to anyone?****

**No**. You can only send messages to **verified phone numbers** unless you have a business-verified account.

### ****🔹 Q: Is WhatsApp notification free?****

* **Yes**: Up to **1,000 messages per month** (Free Tier).
* **After that**: You need to pay **per message**.

### ****🔹 Q: Can we send messages to groups?****

Not directly. You can send messages only to **individual phone numbers**.

**WhatsApp Notifications with Twilio in Spring Boot** 🚀

If you want to send **WhatsApp notifications** instead of SMS, you can use **Twilio's WhatsApp API**.

## ****✅ Steps to Send WhatsApp Messages Using Twilio****

### ****1️⃣ Prerequisites****

✔ **Twilio Account** – [Sign up here](https://www.twilio.com/)  
✔ **Twilio WhatsApp Sandbox** – Activate it in [Twilio Console](https://www.twilio.com/console/sms/whatsapp/learn)  
✔ **Spring Boot project with Twilio dependency**

### ****2️⃣ Add Twilio Dependency in**** pom.xml

<dependency>

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

<artifactId>twilio</artifactId>

<version>9.0.0</version>

</dependency>

### ****3️⃣ Configure Twilio Credentials in**** application.properties

twilio.account.sid=YOUR\_TWILIO\_ACCOUNT\_SID

twilio.auth.token=YOUR\_TWILIO\_AUTH\_TOKEN

twilio.whatsapp.number=whatsapp:+14155238886 # Twilio Sandbox Number

📌 **Note**: Twilio provides a sandbox number (+14155238886) for testing.  
👉 You need to **send "join {your-code}"** to this number on WhatsApp **once** to activate the sandbox.

### ****4️⃣ Create a Service to Send WhatsApp Messages****

package com.example.demo.service;

import com.twilio.Twilio;

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

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

import org.springframework.stereotype.Service;

@Service

public class WhatsAppService {

@Value("${twilio.account.sid}")

private String accountSid;

@Value("${twilio.auth.token}")

private String authToken;

@Value("${twilio.whatsapp.number}")

private String twilioWhatsAppNumber;

public void sendWhatsAppMessage(String to, String messageBody) {

Twilio.init(accountSid, authToken);

Message message = Message.creator(

new com.twilio.type.PhoneNumber("whatsapp:" + to), // Receiver's WhatsApp Number

new com.twilio.type.PhoneNumber(twilioWhatsAppNumber), // Twilio WhatsApp Number

messageBody

).create();

System.out.println("✅ WhatsApp Message Sent! SID: " + message.getSid());

}

}

### ****What Does "Send 'join {your-code}'" Mean?****

When using **Twilio’s WhatsApp Sandbox**, you must first **opt-in** by sending a specific message to Twilio’s sandbox number.

📌 **Twilio Sandbox WhatsApp Number** → +14155238886

📌 **To activate the sandbox, send:**

join <your-code>

to +14155238886 on WhatsApp.

#### **Where Do I Find ""?**

* **Log in to Twilio Console** → [Twilio WhatsApp Sandbox](https://www.twilio.com/console/sms/whatsapp/learn)
* Your **unique "join" code** will be displayed there.
* Example:
* join bright-sky

📌 **Without this step, your phone number won't receive WhatsApp messages from Twilio.**

## ****🚀 Enhance Overdue Task Notification: Add WhatsApp Alerts****

Currently, you're sending **Email & SMS** for overdue tasks.  
Let’s also send a **WhatsApp alert** using Twilio.

### ****5 Update**** checkOverdueTasks() ****to Include WhatsApp****

Modify your existing checkOverdueTasks() method:

@Scheduled(fixedRate = 5000)

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

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

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

// Send Email

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

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

// Send SMS

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

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

// Send WhatsApp

whatsappService.sendWhatsAppMessage(employee.getPhoneNumber(), messageBody);

System.out.println("💬 WhatsApp message sent to " + employee.getPhoneNumber());

}

}

}

### ****6 Test the WhatsApp Integration****

✅ Ensure your phone is opted in (**send 'join ' to +14155238886**)  
✅ Run your **Spring Boot app**  
✅ Trigger an **overdue task scenario**

📌 **Expected Output in Console:**

📧 Email sent to alice@example.com

📲 SMS sent to +919876543210

💬 WhatsApp message sent to +919876543210

📌 **Expected WhatsApp Message on Employee's Phone:**

⚠️ Task Overdue: 12345

Priority: HIGH

Due Date: 2025-03-01