pipeline {

    agent any

    stages {

        stage('Build') {

            steps {

                echo 'Build the code using a build automation tool.'

                echo 'Tool: Maven'

            }

        }

        stage('Unit and Integration Tests') {

            steps {

                echo 'Run unit tests and integration tests.'

                echo 'Tool: JUnit for unit tests.'

                echo 'Tool: Selenium for integration tests.'

            }

            post {

                success {

                    script {

                        def logFile = "${env.BUILD\_ID}\_unit\_integration.log"

                        writeFile file: logFile, text: currentBuild.rawBuild.getLog().join("\n")

                        emailext(

                            to: 'tamlac2012@yahoo.com.vn',

                            subject: "Unit and Integration Tests Successful: ${env.JOB\_NAME} Build #${env.BUILD\_NUMBER}",

                            body: """The Unit and Integration Tests stage was successful. Check console output at ${env.BUILD\_URL} to view the results.""",

                            attachLog: true,

                            attachmentsPattern: logFile

                        )

                    }

                }

                failure {

                    script {

                        def logFile = "${env.BUILD\_ID}\_unit\_integration.log"

                        writeFile file: logFile, text: currentBuild.rawBuild.getLog().join("\n")

                        emailext(

                            to: 'tamlac2012@yahoo.com.vn',

                            subject: "Unit and Integration Tests Failed: ${env.JOB\_NAME} Build #${env.BUILD\_NUMBER}",

                            body: """The Unit and Integration Tests stage failed. Check console output at ${env.BUILD\_URL} to view the results.""",

                            attachLog: true,

                            attachmentsPattern: logFile

                        )

                    }

                }

            }

        }

        stage('Code Analysis') {

            steps {

                echo 'Analyze the code to ensure it meets industry standards.'

                echo 'Tool: SonarQube'

            }

        }

        stage('Security Scan') {

            steps {

                echo 'Perform a security scan to identify vulnerabilities.'

                echo 'Tool: OWASP ZAP'

            }

            post {

                success {

                    script {

                        def logFile = "${env.BUILD\_ID}\_security\_scan.log"

                        writeFile file: logFile, text: currentBuild.rawBuild.getLog().join("\n")

                        emailext(

                            to: 'tamlac2012@yahoo.com.vn',

                            subject: "Security Scan Successful: ${env.JOB\_NAME} Build #${env.BUILD\_NUMBER}",

                            body: """The Security Scan stage was successful. Check console output at ${env.BUILD\_URL} to view the results.""",

                            attachLog: true,

                            attachmentsPattern: logFile

                        )

                    }

                }

                failure {

                    script {

                        def logFile = "${env.BUILD\_ID}\_security\_scan.log"

                        writeFile file: logFile, text: currentBuild.rawBuild.getLog().join("\n")

                        emailext(

                            to: 'tamlac2012@yahoo.com.vn',

                            subject: "Security Scan Failed: ${env.JOB\_NAME} Build #${env.BUILD\_NUMBER}",

                            body: """The Security Scan stage failed. Check console output at ${env.BUILD\_URL} to view the results.""",

                            attachLog: true,

                            attachmentsPattern: logFile

                        )

                    }

                }

            }

        }

        stage('Deploy to Staging') {

            steps {

                echo 'Deploy the application to a staging server.'

                echo 'Staging Server: AWS EC2 instance'

            }

        }

        stage('Integration Tests on Staging') {

            steps {

                echo 'Run integration tests on the staging environment.'

                echo 'Tool: Selenium'

            }

            post {

                success {

                    script {

                        def logFile = "${env.BUILD\_ID}\_staging\_integration.log"

                        writeFile file: logFile, text: currentBuild.rawBuild.getLog().join("\n")

                        emailext(

                            to: 'tamlac2012@yahoo.com.vn',

                            subject: "Integration Tests on Staging Successful: ${env.JOB\_NAME} Build #${env.BUILD\_NUMBER}",

                            body: """The Integration Tests on Staging stage was successful. Check console output at ${env.BUILD\_URL} to view the results.""",

                            attachLog: true,

                            attachmentsPattern: logFile

                        )

                    }

                }

                failure {

                    script {

                        def logFile = "${env.BUILD\_ID}\_staging\_integration.log"

                        writeFile file: logFile, text: currentBuild.rawBuild.getLog().join("\n")

                        emailext(

                            to: 'tamlac2012@yahoo.com.vn',

                            subject: "Integration Tests on Staging Failed: ${env.JOB\_NAME} Build #${env.BUILD\_NUMBER}",

                            body: """The Integration Tests on Staging stage failed. Check console output at ${env.BUILD\_URL} to view the results.""",

                            attachLog: true,

                            attachmentsPattern: logFile

                        )

                    }

                }

            }

        }

        stage('Deploy to Production') {

            steps {

                echo 'Deploy the application to a production server.'

                echo 'Production Server: AWS EC2 instance'

            }

        }

    }

    post {

        always {

            echo 'Pipeline has finished.'

        }

    }

}