Certificate Web App

Technologies:

- **Programming Languages**: Core Java 17
- **Frameworks**: Spring Boot, Spring Security, Spring Batch, Spring Framework, Spring Retry, Spring AOP
- **Design Patterns**: Strategy, Factory, Observer, Adapter, Null Handler
- **Databases**: Firebase Database, PostgreSQL
- **Messaging Queues**: Kafka, RabbitMQ
- **Frontend Technologies**: React.js, Redux
- **Scripting**: Google Apps Script ,Google Calendar
- **Code Quality**: SonarQube
- **Rules Engine**: Drools
- **Compression**: Gzip
- **Logging and Monitoring**: ELK (Elasticsearch, Logstash, Kibana), Log4J
- **Containerization**: Docker
- **API Documentation**: Swagger
- **Libraries**: iTextPDF, Apache POI
- **Video Communication**: OpenVidu
- **Reporting**: Jasper Reports
- **Caching**: Redis
- **Web Servers**: Jetty

Kafka Setup

1. **Unzip Kafka and Place It in Your Desired Directory**

- First, download the Apache Kafka zip file from the official website.
- After downloading, unzip the Kafka folder into your desired directory (for example, `C:\kafka`).

2. **Open Command Prompt as Administrator**
   - Press `Windows + R`, type `cmd`, and press Enter to open Command Prompt.
   - To run it as Administrator, press `Ctrl + Shift + Enter` after typing `cmd`.

---

1. **Start Zookeeper**

   Kafka requires **Zookeeper** to manage distributed processes. Start Zookeeper before starting Kafka.

   Steps to Start Zookeeper:

   - Open the **Administrator Command Prompt** window.
   - Navigate to the Kafka directory:

```
C:\Windows\System32>cd C:\kafka\bin\windows
```

Start Zookeeper with the following command:

```
C:\kafka\bin\windows>.\zookeeper-server-start.bat C:\kafka\conf
```

This will start the Zookeeper server. Leave this command window open.

Start Kafka Server

After Zookeeper is up and running, you can start the Kafka server.

Steps to Start Kafka:

- Open a new **Administrator Command Prompt** window.
- Navigate to the Kafka directory again:

```
C:\Windows\System32>cd C:\kafka\bin\windows
```

Run Kafka server:

```
C:\kafka\bin\windows>kafka-server-start.bat C:\kafka\config\serv
```

This will start the Kafka server. Kafka will use the default settings from the `server.properties` file.

Create a Kafka Topic for Your App

Now that Kafka is running, create a topic for your application (e.g., `logs`).

Steps to Create a Topic:

- In the same **Administrator Command Prompt** window:

```
C:\Windows\System32>cd C:\kafka\bin\windows
C:\kafka\bin\windows>kafka-topics.bat --create --bootstrap
-server localhost:9092 --replication-factor 1 --partitions
1 --topic logs
```

This command creates a Kafka topic named `logs` with 1 partition and a replication factor of 1 (for a single-node Kafka setup).

Redis Setup

1. **Download Redis**
   - Go to the Redis download page and download the latest version for Windows.
   - Alternatively, you can download a Windows-compatible version from Microsoft's Redis GitHub repository.
2. **Unzip Redis**
   - After downloading, unzip the Redis folder into your desired directory (for example, `C:\Redis`).
3. **Run Redis**

- Open **Command Prompt** and navigate to the Redis directory:

```
C:\Windows\System32>cd C:\Redis
```

Start the Redis server by running:

```
C:\Redis>redis-server.exe
```

Redis will start and listen for connections on the default port `6379`.

RabbitMQ Setup

1. **Download RabbitMQ**
   - Visit the RabbitMQ download page and download the latest version of RabbitMQ for Windows.
2. **Install RabbitMQ**
   - Run the downloaded installer and follow the installation steps to install RabbitMQ on your system.
3. **Install Erlang (Dependency for RabbitMQ)**
   - RabbitMQ requires Erlang to run, so you need to install Erlang first.
   - You can download Erlang from the official Erlang website.
   - After installing Erlang, you can proceed with the RabbitMQ installation.
4. **Start RabbitMQ Server**
   - After installation, open **Command Prompt** and navigate to the RabbitMQ `sbin` directory:

```
C:\Windows\System32>cd C:\Program Files\RabbitMQ Server
\rabbitmq_server-<version>\sbin
```

Start the RabbitMQ service:

```
rabbitmq-server.bat
```

-
    - This will start the RabbitMQ service. By default, RabbitMQ will be available at `http://localhost:15672` (the management web interface).
- **Enable RabbitMQ Management Plugin (Optional)**
    - To enable the RabbitMQ management plugin (for accessing the web UI):

```
rabbitmq-plugins enable rabbitmq_management
```

You can then access the RabbitMQ UI in your browser at

```
http://localhost:15672/
```

. The default login credentials are:

- **Username:** guest
- **Password:** guest

Firebase Push Notification Setup

Follow this tutorial to get more info

https://docs.ngrow.ai/en/articles/8886580-how-to-grant-access-to-firebase-cloud-messaging-v1-api

Step 1: Create a Firebase Account

1. Visit Firebase Console.
2. Sign in with your Google account. If you don’t have one, create a Google account first.

Step 2: Create a Firebase Project

1. Click on **"Add Project"** or **"Create a Project"**.
2. Enter the project name (e.g., `PushNotificationsApp`).
3. (Optional) Enable Google Analytics for the project.
    - If you enable Analytics, configure the Analytics account.
4. Click **"Create Project"** and wait for the process to complete.

Step 3: Add Your App to the Firebase Project

1. After creating the project, navigate to the Firebase dashboard.
2. Click **"Add App"** and choose the platform:
    - **Android**: Select the Android icon.
    - **iOS**: Select the iOS icon.
    - **Web**: Select the Web icon.

For Android:

1. Enter the **package name** of your Android app.
2. (Optional) Add an **App nickname**.
3. (Optional) Enter your **SHA-1 key** (needed for certain integrations like Google Sign-In).
4. Download the **google-services.json** file and place it in your app's root directory (`app/`).

For iOS:

1. Enter the **iOS bundle ID** (found in Xcode).
2. (Optional) Add an **App nickname**.
3. (Optional) Enter the **App Store ID**.
4. Download the **GoogleService-Info.plist** file and add it to your iOS project in Xcode.

For Web:

1. Register the app with a nickname.
2. Firebase provides a **Firebase SDK configuration object**. Copy it.
3. Integrate this object into your web app's JavaScript code.

Step 4: Enable Push Notifications

1. Go to **Project Settings** (gear icon in the top-left menu).
2. Navigate to the **Cloud Messaging** tab.
3. For Android/iOS:
   - Retrieve the **Server Key** and **Sender ID**. These are used to send push notifications via Firebase Cloud Messaging (FCM).
4. For Web:
   - Generate a **Web Push Certificate**:
     - Click **Generate Key Pair** under the "Web Push Certificates" section.
     - Copy the public key and add it to your web application.

Step 5: Integrate Firebase into Your App

Android Integration:

1. Add Firebase dependencies to your `build.gradle` files:
   - **Project-Level (`build.gradle`)**:

```
dependencies {
    classpath 'com.google.gms:google-services:4.3.15'
}
```

App-Level (`build.gradle`):

```
dependencies {
implementation 'com.google.firebase:firebase-messaging:24.4.1'
```

```
}
```

Apply the Google Services plugin:

```
apply plugin: 'com.google.gms.google-services'
```

iOS Integration:

1. Use **CocoaPods** to install Firebase Messaging:

```
pod 'Firebase/Messaging'
```

Configure Firebase in the `AppDelegate`.

Note : - Changes in your react app we are using web version, You can skip two steps android or IOS setup

Web Integration:

1. Use the Firebase SDK for JavaScript to integrate FCM.
2. Add the Firebase configuration and initialize Firebase in your app:

```
import { initializeApp } from "firebase/app";
import { getMessaging, getToken, onMessage } from "firebase/mess

const firebaseConfig = {
    apiKey: "your-api-key",
    authDomain: "your-project-id.firebaseapp.com",
    projectId: "your-project-id",
    storageBucket: "your-project-id.appspot.com",
    messagingSenderId: "your-sender-id",
    appId: "your-app-id"
};
```

```
const app = initializeApp(firebaseConfig);
const messaging = getMessaging(app);
```

Enable "Allow Less Secure Apps" (For Non-2FA Accounts)

If your account does **not** have 2-step verification enabled:

1. Log in to your Gmail account.
2. Go to Google Account Security Settings.
3. Scroll down to **"Less secure app access"**.
4. Enable the option to allow less secure apps to access your account.

Step 2: Enable 2-Step Verification (For Secure Accounts)

If your account uses 2-step verification (recommended for better security):

1. Go to Google Account Security Settings.
2. Under **"Signing in to Google"**, click **"2-Step Verification"** and follow the steps to enable it.

Step 3: Create an App Password

1. Log in to your Gmail account.
2. Go to Google Account Security Settings.
3. Under **"Signing in to Google"**, select **"App Passwords"**.
4. Enter your Google account password if prompted.
5. From the **"Select app"** dropdown, choose **Mail** or **Other (Custom)**, then specify a name (e.g., "Email Integration").
6. From the **"Select device"** dropdown, choose your device or leave as default.
7. Click **"Generate"**.

8. Copy the generated 16-character app password and save it securely. This password will be used in place of your regular Gmail password in email configurations.

Note:

Now go to your backend app and put your app password in application. properties file

SonarQube

Step 1: Install and Run SonarQube Locally

1. **Download SonarQube:**
   - Visit SonarQube Downloads and download the Community Edition.
2. **Install Java:**
   - Ensure that Java 11 or higher is installed on your system (required for SonarQube).(You need open jdk)

   https://adoptium.net/en-GB/temurin/archive/?version=17
3. **Extract and Start SonarQube:**
   - Extract the downloaded file to a desired location.
   - Navigate to the `bin` folder inside the extracted directory.
   - Run the appropriate script based on your operating system:
     - Windows: `StartSonar.bat`
     - Linux/Mac: `./sonar.sh start`
4. **Access SonarQube:**
   - Open a browser and go to `http://localhost:9000`.
   - Log in with the default credentials:
     - Username: `admin`

- Password: `admin`

5. **Change the Default Password:**
   - After logging in, change the admin password for security.

Step 2: Generate a Sonar Token

1. Navigate to **My Account** → **Security** in SonarQube.
2. Click **Generate Token**.
3. Enter a name for the token (e.g., `maven-token`) and click **Generate**.
4. Copy the generated token. **Save it securely**, as it won't be shown again.

Step 3: Configure Maven (`pom.xml`)

Add the SonarQube token and properties to your `pom.xml`:

1. **Add the Sonar Token:**

```
<properties>
   <!-- Sonar User Token -->
    <sonar.login>YOUR_SONAR_QUBE_TOKEN_HERE</sonar.login>
    <java.version>17</java.version>
 </properties>
```

2. **Add the sonar plugin:**

```
<plugins>
      <plugin>
           <groupId>org.sonarsource.scanner.maven</groupId>
           <artifactId>sonar-maven-plugin</artifactId>
           <version>3.7.0.1746</version>
      </plugin>
</plugins>
```

Step 4: Run Sonar Analysis

1. Open a terminal in your project directory.
2. Run the following Maven command:

```
mvn clean install sonar:sonar
```

Step 5: View Results in SonarQube

1. Go to `http://localhost:9000`.
2. Log in and navigate to your project dashboard.
3. Review the analysis, including bugs, vulnerabilities, and code smells.

LogConsumer Service Setup

Follow all kafka setup steps

Step 1: Install Prerequisites

1. **Java Installation**:
   - Elasticsearch and Logstash require Java. Install Java 8 or higher.
   - Verify Java installation:

```
java -version
```

Step 2: Download ELK Components

1. **Elasticsearch**:
   - Download from the official Elasticsearch website.
   - Extract the archive:

```
tar -xzf elasticsearch-<version>.tar.gz
```

2. **Logstash**:

- Download from the official Logstash website.
- Extract the archive:

```
tar -xzf logstash-<version>.tar.gz
```

3. **Kibana**:

- Download from the official Kibana website.
- Extract the archive:

```
tar -xzf kibana-<version>.tar.gz
```

Step 3: Start Elasticsearch

Navigate to the Elasticsearch directory:

```
cd elasticsearch-<version>/bin
```

Start Elasticsearch:

```
./elasticsearch
```

Verify Elasticsearch is running by accessing:

- http://localhost:9200
- You should see a JSON response with cluster information.

Step 4: Start Kibana

1. Navigate to the Kibana directory:

```
cd kibana-<version>/bin
```

Start Kibana:

```
./kibana
```

Access Kibana in your browser at:

- http://localhost:5601

Step 5: Configure Logstash

1. Navigate to the Logstash directory:

```
cd logstash-<version>
```

Create a Logstash configuration file (e.g., `logstash.conf`):

```
nano logstash.conf
```

Start Logstash with the configuration file:

```
./bin/logstash -f logstash.conf
```

Step 6: Test the ELK Setup

1. Generate log data in the specified log file (e.g., `/path/to/logfile.log`).
2. Logstash should process the data and send it to Elasticsearch.
3. In Kibana:

- Go to **"Discover"**.
- Set up the default index pattern (e.g., `logstash-*`).
- View and analyze the logs.

Frontend App Setup

Got to app directory and run command

```
npm install // For Download node modules package
```

```
npm start // For lauch app
```

Google App Script Setup

Now your all setup done but for generate documents you need to setup google app script for perform different different task like For generate Salary, Document, Access google drive data and Schedule your interview

Step 1: Access Google Apps Script

1. Open your browser and go to Google Apps Script.
2. If prompted, sign in with your Google account.

Step 2: Create a New Project

1. On the Apps Script dashboard, click **New Project**.

2. You'll be directed to the Apps Script editor with a blank script file named `Code.gs`.
3. After that you need to deploy the script and and get the script url and copy this url and store in database to perform own task and to excute script

Step 3: Pull all your script one by one

1. Mark share anyone