**HTTPS**

This is an extended version of HTTP

Enabling HTTPs in Spring Boot:

HTTPs stands for **H**yper **T**ext **T**ransfer **P**rotocol Secure. When Spring Boot Application starts, by default it uses the HTTP 8080 port. To configure HTTPs protocol and port 443 in the spring boot application, we must follow the below 2 steps.

* Create the SSL/TLS Certificate – A self-signed certificate or we can get it from a certificate authority.
* Enable HTTPS protocol and port number 443

What is SSL Certificate?

An SSL (Secure socket layer) certificate is a Digital certificate that can be used for authentication of a website, and it creates a secure connection between client and web server. When a certificate is installed, it makes the website from HTTP to HTTPs.

* The SSL certificate enables the encryption of data which is then sent to the server-side. It has two keys one is public, and the other one is a private key. Data encrypted with the public key can be decrypted with a private key only. The web server with a private key can understand the data.  If data packets are stolen from in between those are useless because they are encrypted.
* Types –
* Single-domain SSL certificates
* Wildcard SSL certificates
* Multi-Domain SSL certificates
* Unified Communications SSL Certificate
* To create a self-signed certificate using Java Keytool, follow these steps:

Open your command console (terminal) on the operating system where you have Java installed.

Navigate to the directory where keytool.exe is located (usually where the JRE is installed, e.g., c:\Program Files\Java\jre6\bin on Windows machines).

Run the following command, adjusting the parameters as needed:

**keytool -genkey -keyalg RSA -alias selfsigned -keystore keystore.jks -storepass password -validity 360 -keysize 2048**

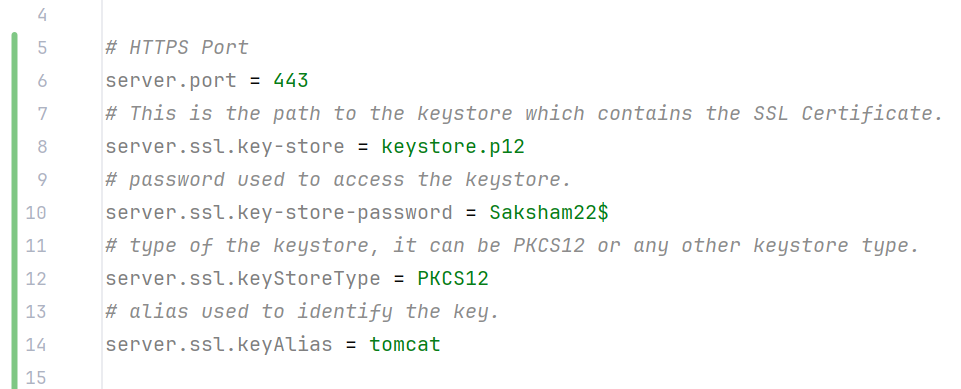
Replace keystore.jks with your desired keystore filename.

Modify the -validity value (number of days before the certificate expires) as necessary.

Fill in the prompts for your organization information. When asked for your first and last name, enter the domain name of the server users will connect to (e.g., www.example.com).

This command will create a keystore.jks file containing a private key and your fresh self-signed certificate.

* To enable HTTPs, we have to update the **application.properties** file in the project with explanation of important properties like keystore path and password etc.



* Build Maven.

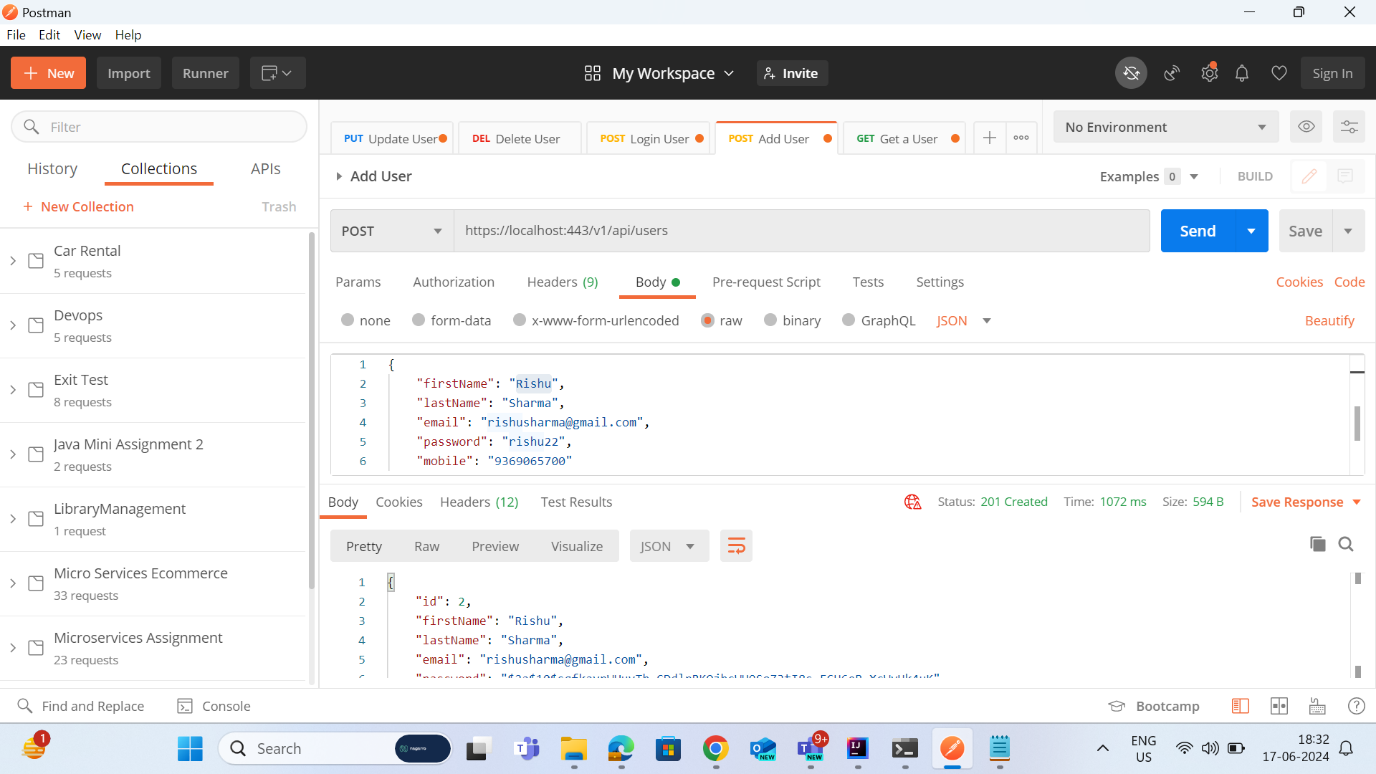
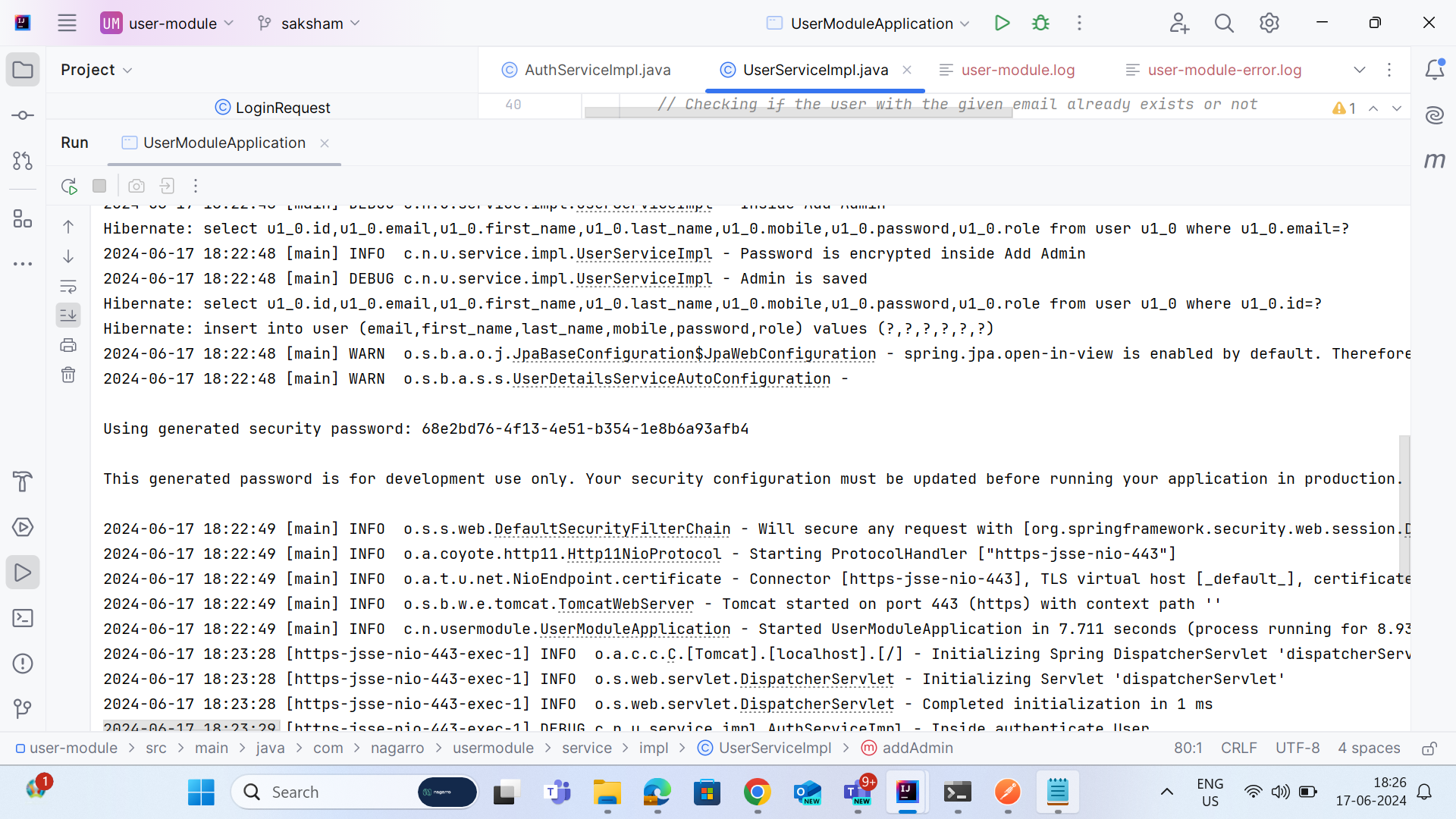
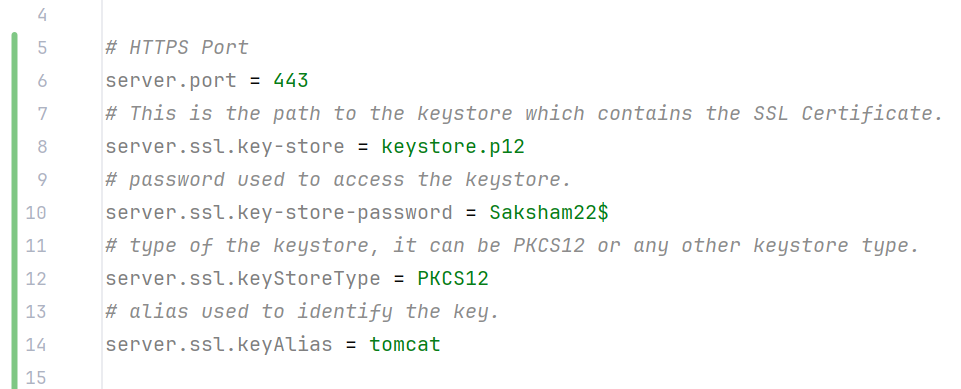
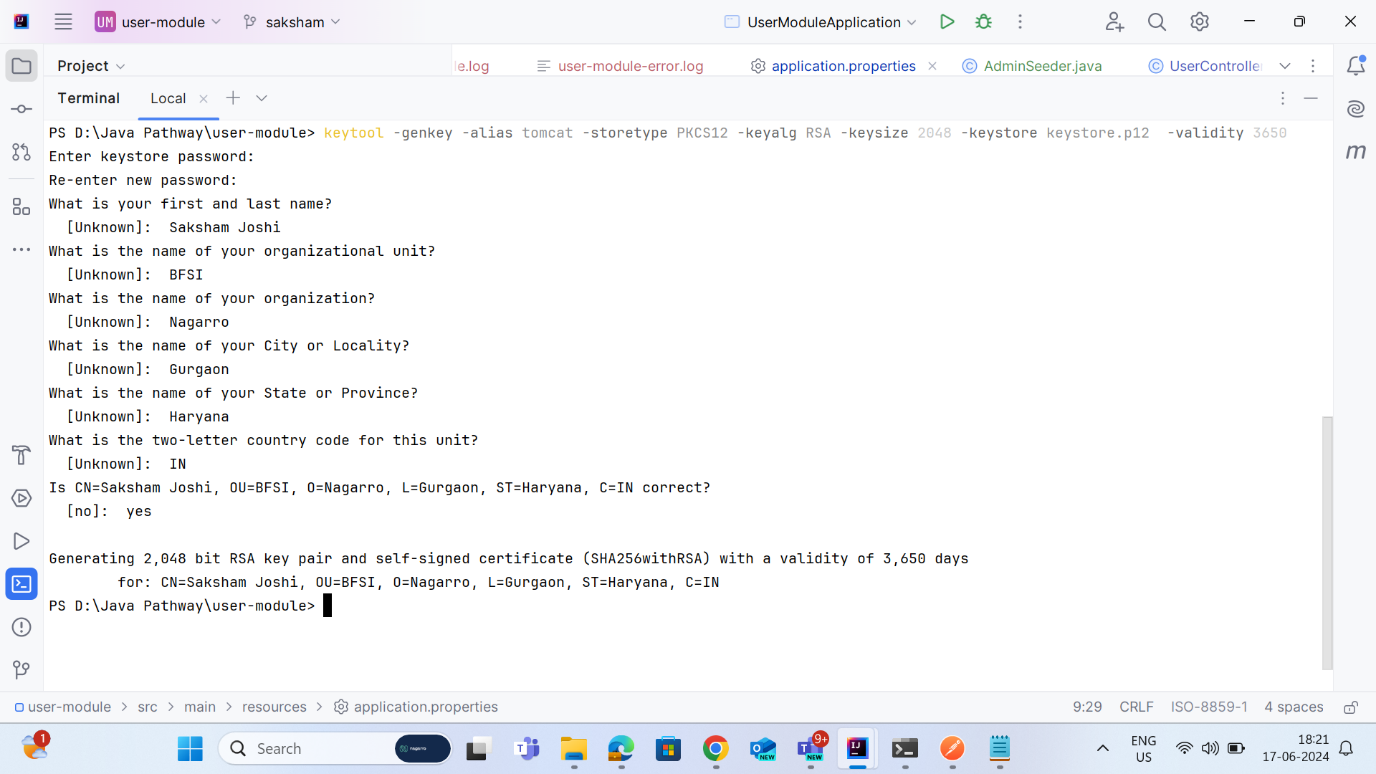
**Advantages of HTTPs**

* Provides in-transit data security.
* Shields your website from data breaches, phishing, and MITM attacks.
* Increases the visitors’ trust to your website.
* Eliminates the “NOT Secure” alerts.
* Assist you in raising your website’s ranking.

**Disadvantages of HTTP**

* When switching to HTTPS, an SSL certificate needs to be bought. Even though website hosts often give SSL certificates, these should be renewed annually by paying a charge.
* Encrypting and decrypting data across HTTPS connections requires a lot of computation.
* There will be issues with caching some information over HTTPS. Public caching of those that previously took place will not happen again.
* Certain proxy servers and firewalls prevent users from accessing HTTPS websites. Both deliberate and inadvertent actions might result from this.
* If there are configuration issues, HTTP will be used by your website to obtain files rather than HTTPS (HTTP Secure).

Enabling HTTPS in User-Module



Ref - [How to Enable HTTPs in Spring Boot Application? - GeeksforGeeks](https://www.geeksforgeeks.org/how-to-enable-https-in-spring-boot-application/)

Ref - [Difference Between HTTP and HTTPS - GeeksforGeeks](https://www.geeksforgeeks.org/difference-between-http-and-https-2/)