Authentication

To check identitiy of user

Validating user

- userid+pwd
- emailid+pwd
- otp
- emailid+opt
- dynamic link

Authorization

After authentication

Checks ROLE/Desig/Permission

Allows to access a particular resource

JPA Repo

Mongo Repo

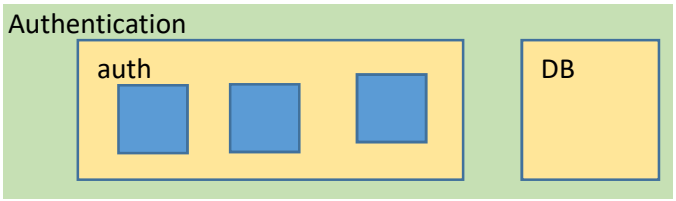
authentication

	op1	op2
	implement auth mechanisum in all applications individually	implement one centralized auth mechanism, let all apps use the same

client app

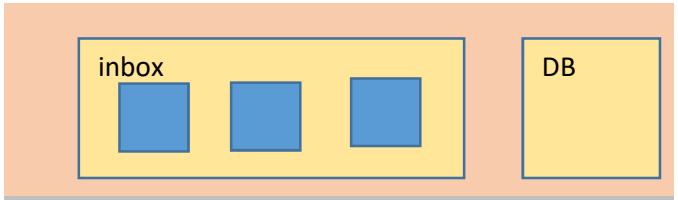
id+pwd

RESTAPI server



raj@gmail.com 12345
abc@gmail.com 12345
xyz

RESTAPI cleint

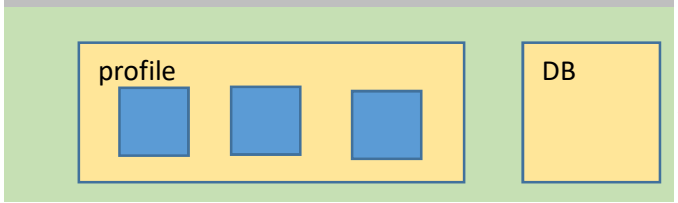


raj@gmail.com m1,m2,m3
abc@gmail.com m4,m5,m6
xyz

RESTAPI cleint

raj@gmail.com s1,s2,s3
abc@gmail.com s4,s5,s6
xyz

RESTAPI cleint

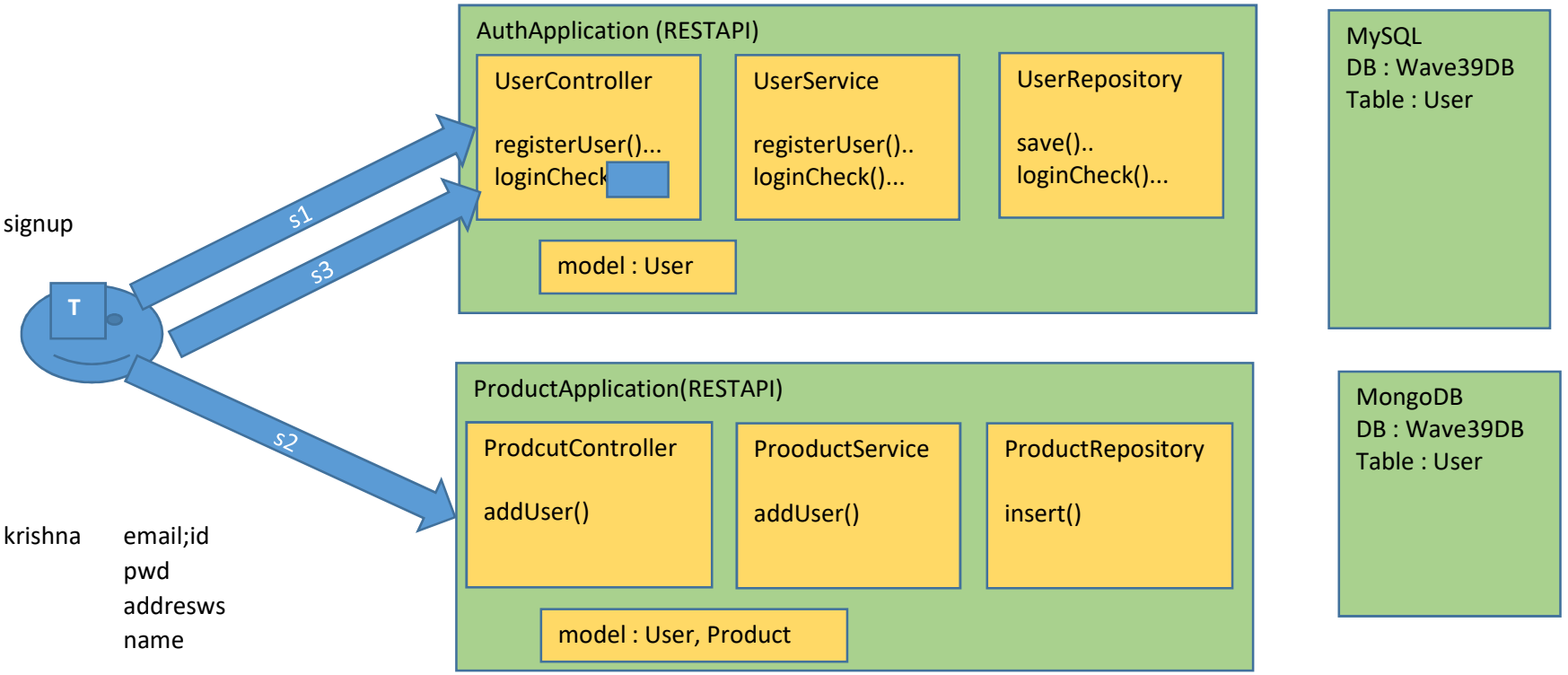


raj@gmail.com Rajeswar theam1
abc@gmail.com Anand theam2
xyz

How to implement manual logincheck mechanism

capture userid+pwd
send the same to BE
BE checks whether the data matching with DB record or not
if matching : login is success
else : login failed

emailid *	password	role
rajeswar@gmail.com	12345	ROLE_USER
anand@gmail.com	12345	ROLE_USER
krsishna@gmail.com	12345	ROLE_USER



krishna email;id
pwd
addresws
name

signup data
emailid pwd name address

product
name desc price

authapp
emailid pwd

productapp
emailid name address

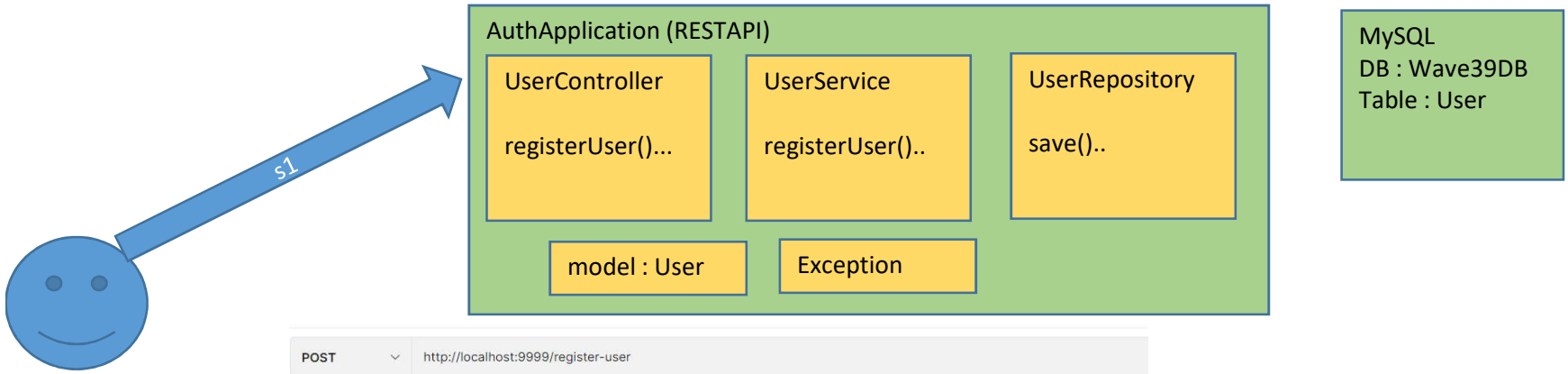
emailid	name	products			
rajeswar@gmail.com	rajeswar	[ ]	s1	creating user a/c in auth app	auth app
anand@gmail.com	anand	[ ]	s2	creating user a/c in product app	product app
krishna@gmail.com	krishna	[ ]	s3	login check	auth app
			s4	access product data with token	

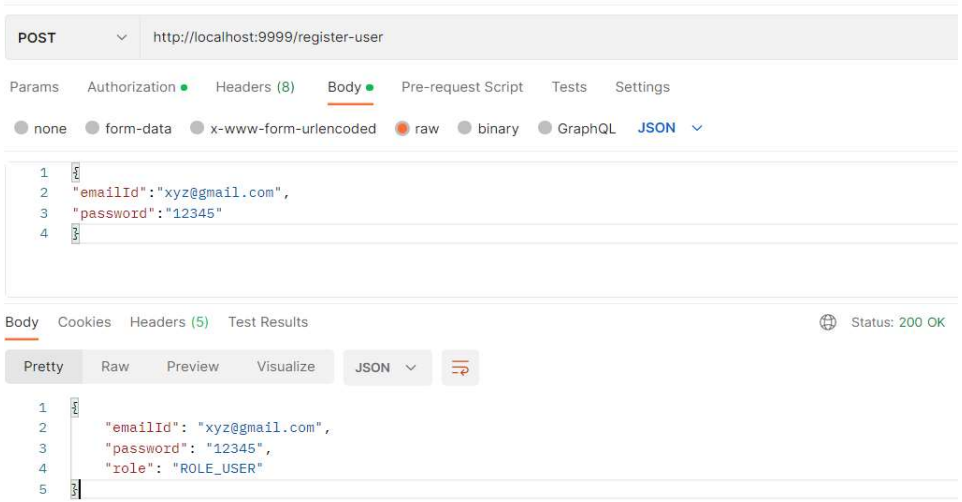
S1 creating user account in auth app

create new springboot application (web,mysql,jpa,lombok)

model : emailid, pwd, role

create flow for creating user a/c in db



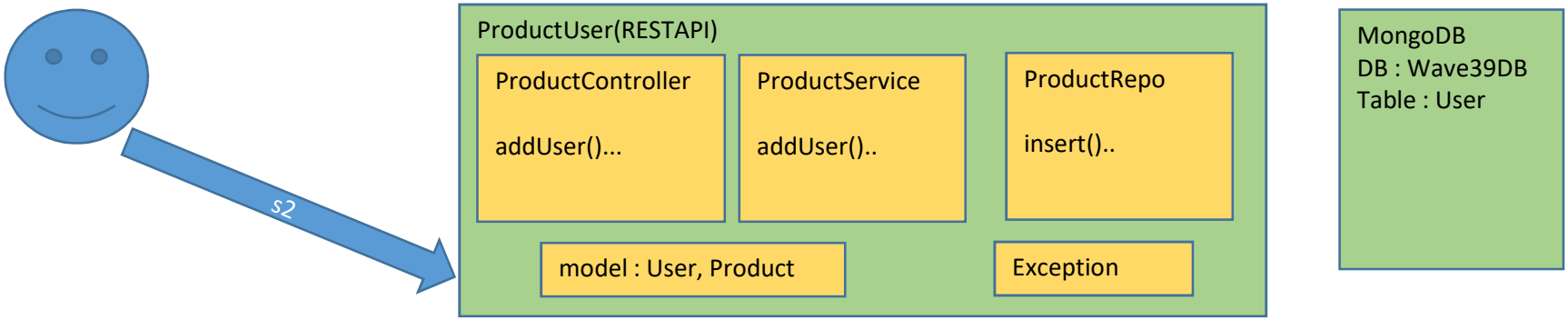


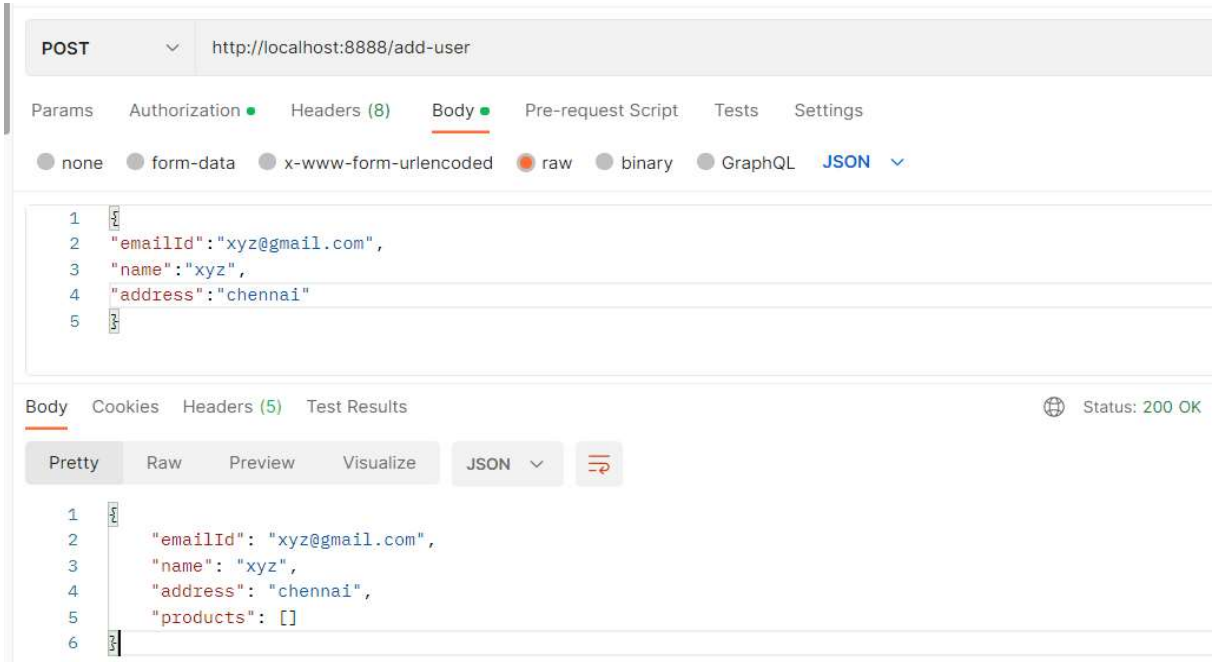
S2 creating user account in prduct app

create new springboot application (web,mongo data,lombok)

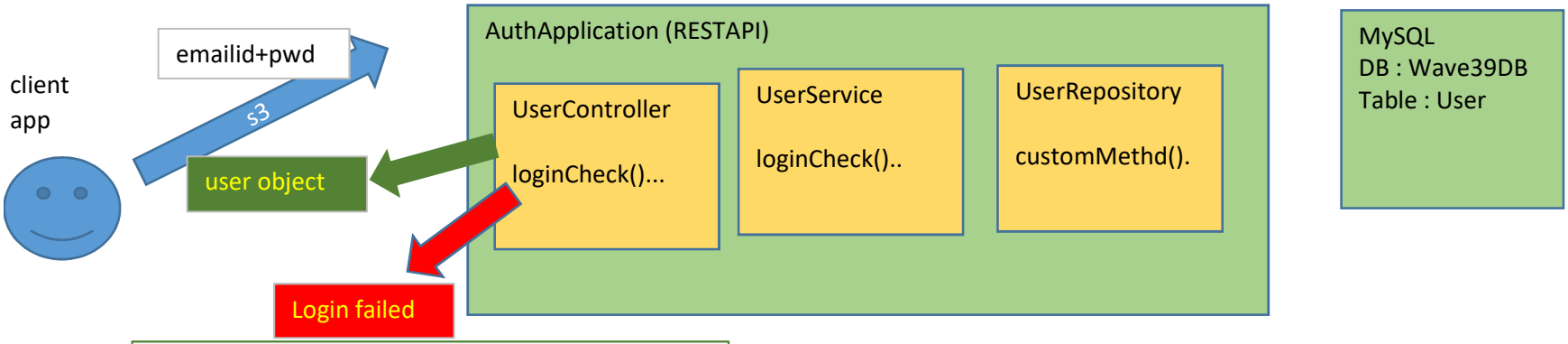
model : emailid, name, address,product[ ]

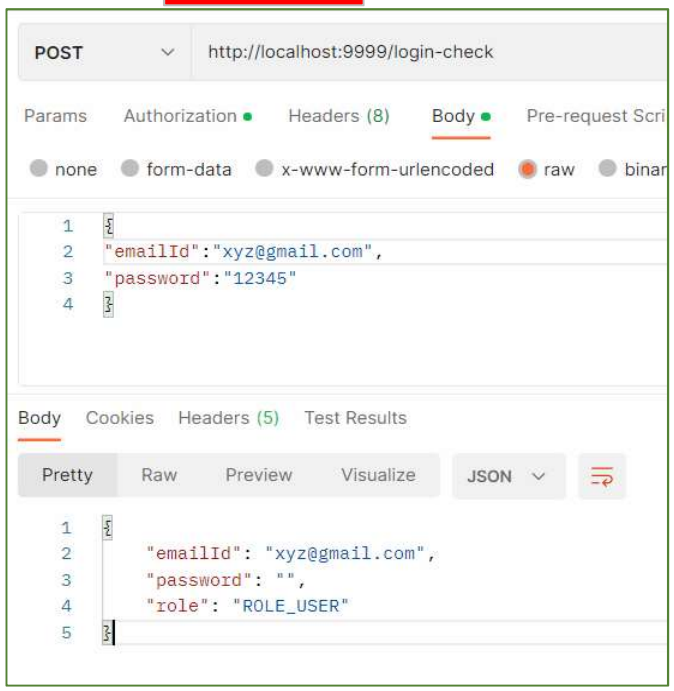
create flow for creating user a/c in db

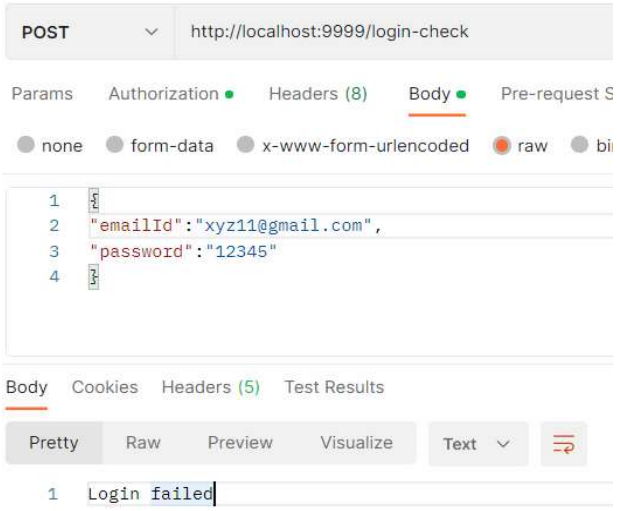




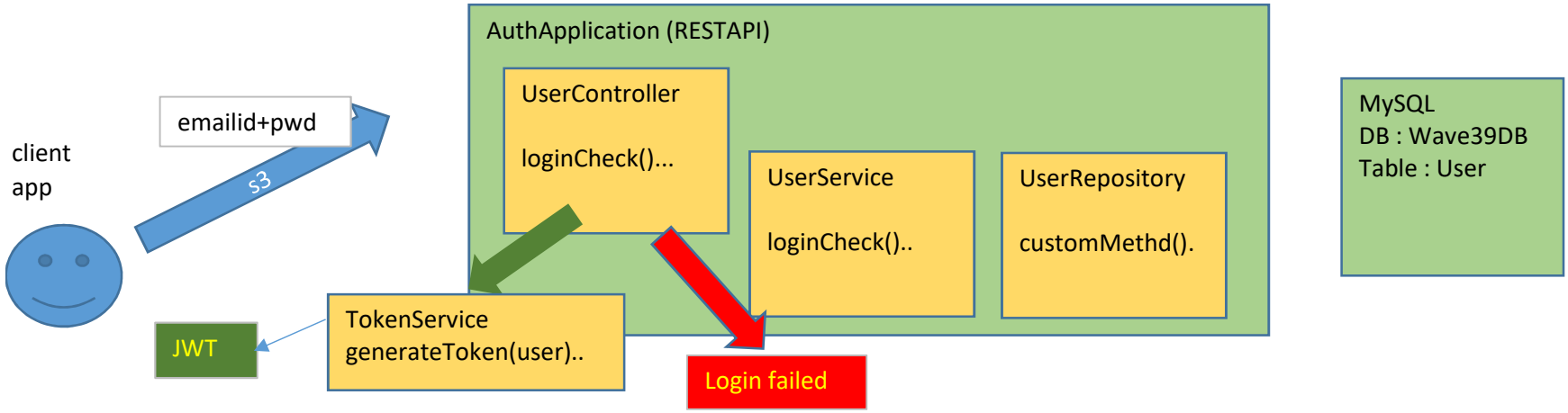
S3 logincheck in auth app





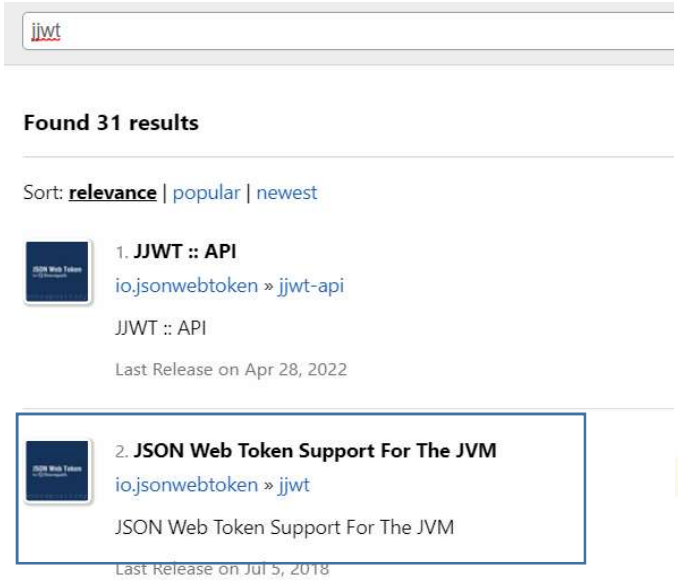


User object is returned as response if login is success



JWT to be given as response if login is success

step 1 add required dependency




```
<!-- https://mvnrepository.com/artifact/io.jsonwebtoken/jjwt -->
<dependency>
   <groupId>io.jsonwebtoken</groupId>
   <artifactId>jjwt</artifactId>
   <version>0.9.1</version>
</dependency>
```

```
<!-- https://mvnrepository.com/artifact/io.jsonwebtoken/jjwt -->
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt</artifactId>
    <version>0.9.1</version>
</dependency>
```

Step 2 Define service layer

```java
public interface JwtTokenGenerator {
    public abstract Map<String,String> generateJwt(User user);
}
```

```java
@Service
public class JwtTokenGeneratorImpl implements JwtTokenGenerator {

    @Override
    public Map<String, String> generateJwt(User user) {
        Map<String, String> result = new HashMap<String, String>();
        // build token ( header,payload, by compacting)
        // user has emailid, password, role
        Map<String,Object> userdata = new HashMap<>();
        userdata.put("emailid",user.getEmailId());
        userdata.put("role",user.getRole());

        String jwt = Jwts.builder()
                .setClaims(userdata) // user data to be filled as claims
                .setIssuedAt(new Date())
                .setIssuer("MyCompany")
                .signWith(SignatureAlgorithm.HS512, s: "idontsay")
                .compact();
        result.put("token",jwt);
        result.put("message","Login success, Token generated");
        return result;
        // return map :  token(key,val), message(key,val)
    }
}
```

Step 3 in controller

add tokengenerator service dependency

```java
    @Autowired
    private JwtTokenGenerator jwtTokenGenerator;
```

logincheck()

call token service

get token

give response as token

```java
/* POST
http://localhost:9999/login-check
*/
@PostMapping("/login-check")
public ResponseEntity<?> loginCheck(@RequestBody User user){
    User result = userService.loginCheck(user.getEmailId(), user.getPassword());
    System.out.println(result);
    if(result!=null){
        result.setPassword("");
        //return new ResponseEntity<>(result,HttpStatus.OK); // old one
        return new ResponseEntity<>(jwtTokenGenerator.generateJwt(result),HttpStatus.OK);
    }
    else{
        return new ResponseEntity<>( body: "Login failed",HttpStatus.OK);
    }
}
```

POST http://localhost:9999/login-check

Params Authorization Headers (8) Body Pre-request Script Tests Settings

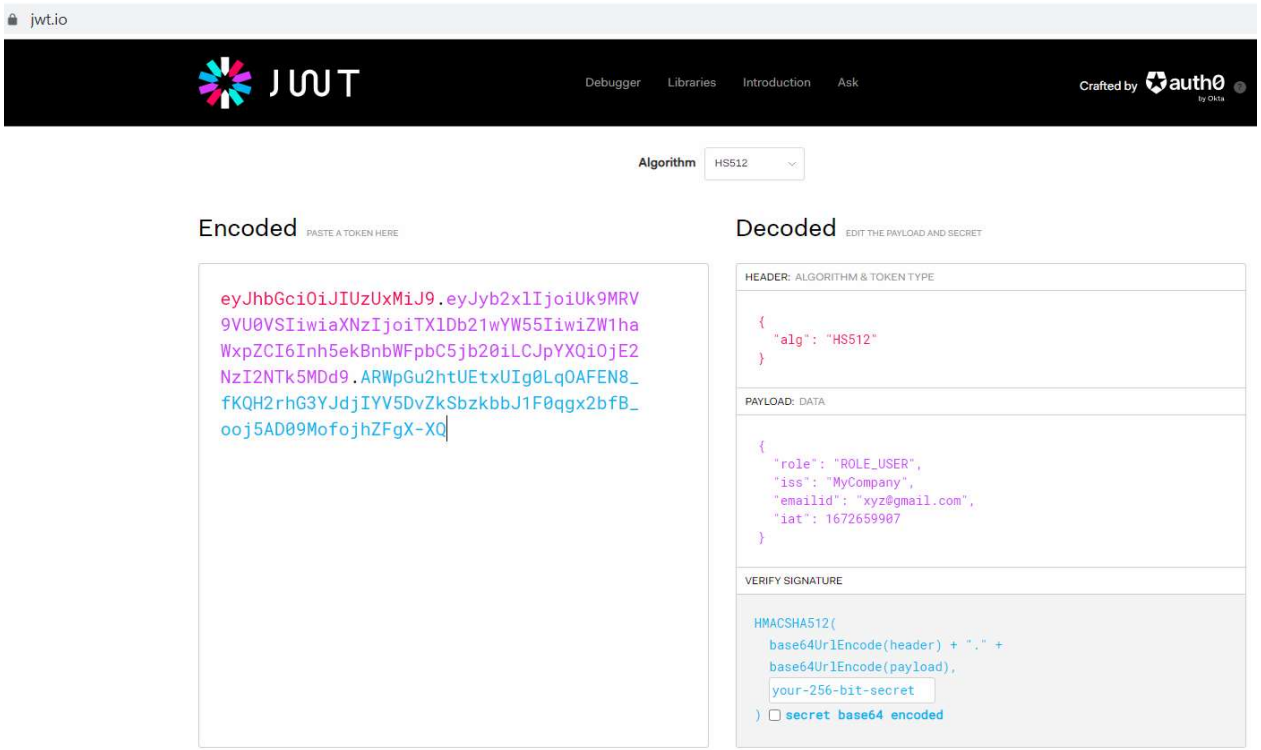
none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
"emailId":"xyz@gmail.com",
"password":"12345"
}
```

Body Cookies Headers (5) Test Results Status: 200 OK Time: 165 ms

Pretty Raw Preview Visualize JSON

```
{
    "message": "Login success, Token generated",
    "token": "eyJhbGciOiJIUzUxMiJ9.
        eyJyb2xlIjoiUk9MRV9VU0VSIiwiaXNzIjoiTXlDb21wYW55IiwiZW1haWxpZCI6Inh5ekBnbWFpbC5jb20iLCJpYXQiOjE2NzI2NTk5MDd9.
        ARWpGu2htUEtxUIg0LqOAFEN8_fKQH2rhG3YJdjIYV5DvZkSbzkbbJ1F0qgx2bfB_ooj5AD09MofojhZFgX-XQ"
}
```


jwt.io
JWT
Debugger
Libraries
Introduction
Ask
Crafted by auth0 by Okta
Algorithm HS512
Encoded PASTE A TOKEN HERE
eyJhbGciOiJIUzUxMiJ9.eyJyb2xlIjoiUk9MRV9VU0VSIiwiaXNzIjoiTXlDb21wYW55IiwiZW1haWxpZCI6Inh5ekBnbWFpbC5jb20iLCJpYXQiOjE2NzI2NTk5MDd9.ARWpGu2htUEtxUIg0LqOAFEN8_fKQH2rhG3YJdjIYV5DvZkSbzkbbJ1F0qgx2bfB_ooj5AD09MofojhZFgX-XQ
Decoded EDIT THE PAYLOAD AND SECRET
HEADER: ALGORITHM & TOKEN TYPE
{
"alg": "HS512"
}
PAYLOAD: DATA
{
"role": "ROLE_USER",
"iss": "MyCompany",
"emailid": "xyz@gmail.com",
"iat": 1672659907
}
VERIFY SIGNATURE
HMACSHA512(
base64UrlEncode(header) + "." +
base64UrlEncode(payload),
your-256-bit-secret
) secret base64 encoded