**java\_springboot\_restful\_10\_Use user from Database using JPA for JWT Continues**

{noop} -> It make use of encoders

**application.properties -> Modify it**

# Server port

server.port=8080

# Database config

spring.datasource.url=jdbc:h2:file:./db/db

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=admin

spring.datasource.password=pass987

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

# setup local h2 database console

spring.h2.console.enabled=true

spring.h2.console.path=/db-console

spring.h2.console.settings.web-allow-others=false

# local h2 tables setup on startup; use the value "none" in production!

spring.jpa.hibernate.ddl-auto=create-drop

# Enable lazy loading

# spring.jpa.properties.hibernate.enable\_lazy\_load\_no\_trans=true

# application.properties

# Thymeleaf settings (No need for version settings here)

# spring.thymeleaf.version=3.0.15.RELEASE

# spring.thymeleaf-layout-dialect.version=2.5.3

# Static files setting

# spring.mvc.static-path-pattern=/resources/static/\*\*

# logging.level.org.springframework.security=DEBUG

**Create a folder -> models and repository -> in project -> In the model -> create a model -> Account.java**

package org.studyeasy.SpringRestdemo.model;

public class Account {

}

**Update UserLogin.java -> payload folder**

public record UserLogin(String username, String password) {

}

**Update it to**

public record UserLogin(String email, String password) {

}

**Account.java**

package org.studyeasy.SpringRestdemo.model;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

import lombok.Getter;

import lombok.Setter;

import lombok.ToString;

@Entity

@Getter

@Setter

@ToString

public class Account {

    @Id

    @GeneratedValue(strategy = GenerationType.SEQUENCE)

    private long id;

    private String email;

    private String password;

    private String role;

}

**Create a repository -> AccountRepository.java -> and make it as interface**

package org.studyeasy.SpringRestdemo.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import org.studyeasy.SpringRestdemo.model.Account;

public interface AccountRepository extends JpaRepository<Account, Long>{

}

**AuthController.java**

@PostMapping("/token")

    @ResponseBody

    public Token token(@RequestBody UserLogin userLogin) throws AuthenticationException {

        Authentication authentication = authenticationManager

                .authenticate(new UsernamePasswordAuthenticationToken(userLogin.username(), userLogin.password()));

        return new Token(tokenService.generateToken(authentication));

    }

**userLogin.username() -> change to -> userLogin.email()**

@PostMapping("/token")

    @ResponseBody

    public Token token(@RequestBody UserLogin userLogin) throws AuthenticationException {

        Authentication authentication = authenticationManager

                .authenticate(new UsernamePasswordAuthenticationToken(userLogin.email(), userLogin.password()));

        return new Token(tokenService.generateToken(authentication));

    }

**In config folder -> Create a file -> SeedData.java -> Add service in service folder -> AccountService.java**

**AccountService.java**

package org.studyeasy.SpringRestdemo.service;

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

import org.studyeasy.SpringRestdemo.repository.AccountRepository;

public class AccountService {

  @Autowired

  private AccountRepository accountRepository;

}

**SeedData.java**

package org.studyeasy.SpringRestdemo.config;

import org.springframework.boot.CommandLineRunner;

import org.springframework.stereotype.Component;

@Component

public class SeedData implements CommandLineRunner{

    @Override

    public void run(String... args) throws Exception {

        // TODO Auto-generated method stub

        throw new UnsupportedOperationException("Unimplemented method 'run'");

    }

}

**In SecurityConfig.java -> We will be creating our own user management system, so remove or uncomment below code.**

 @Bean

    public InMemoryUserDetailsManager users() {

        return new InMemoryUserDetailsManager(

                User.withUsername("chaand")

                        .password("{noop}password")

                        .authorities("read")

                        .build());

    }

**Add below code -> PasswordEncoder()**

@Bean

    public static PasswordEncoder passwordEncoder(){

        return new BCryptPasswordEncoder();

    }

**AccountService.java**

package org.studyeasy.SpringRestdemo.service;

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

import org.springframework.security.crypto.password.PasswordEncoder;

import org.studyeasy.SpringRestdemo.model.Account;

import org.studyeasy.SpringRestdemo.repository.AccountRepository;

@Service

public class AccountService {

    @Autowired

    private AccountRepository accountRepository;

    @Autowired

    private PasswordEncoder passwordEncoder;

    public Account save(Account account){

        account.setPassword(passwordEncoder.encode(account.getPassword()));

        return accountRepository.save(account);

    }

}

**In SeedData.java**

package org.studyeasy.SpringRestdemo.config;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.stereotype.Component;

import org.studyeasy.SpringRestdemo.model.Account;

import org.studyeasy.SpringRestdemo.service.AccountService;

@Component

public class SeedData implements CommandLineRunner{

    @Autowired

    private AccountService accountService;

    @Override

    public void run(String... args) throws Exception {

        Account account01 = new Account();

        Account account02 = new Account();

        account01.setEmail("user@user.com");

        account01.setPassword("pass987");

        account01.setRole("ROLE\_USER");

        accountService.save(account01);

        account02.setEmail("admin@admin.com");

        account02.setPassword("pass987");

        account02.setRole("ROLE\_ADMIN");

        accountService.save(account02);

    }

}

**In SecurityConfig.java -> update -> securityFilterChain method -> add below code**

.csrf(csrf -> csrf.ignoringRequestMatchers("/db-console/\*\*"))

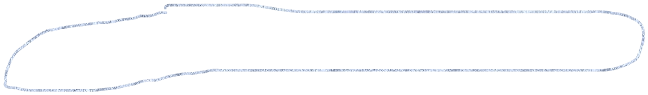
.headers(headers -> headers.frameOptions(frameOptions -> frameOptions.sameOrigin()))

 @Bean

    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

        http

                .csrf(csrf -> csrf.ignoringRequestMatchers("/db-console/\*\*"))

                .headers(headers -> headers.frameOptions(frameOptions -> frameOptions.sameOrigin()))

                .authorizeHttpRequests(authorize -> authorize

                        .requestMatchers("/token").permitAll()

                        .requestMatchers("/").permitAll()

                        .requestMatchers("/swagger-ui/\*\*").permitAll()

                        .requestMatchers("/v3/api-docs/\*\*").permitAll()

                        .requestMatchers("/test").authenticated())

                .oauth2ResourceServer(oauth2 -> oauth2

                        .jwt(Customizer.withDefaults()))

                .sessionManagement(session -> session

                        .sessionCreationPolicy(SessionCreationPolicy.STATELESS))

                .csrf(csrf -> csrf.disable())

                .headers(headers -> headers

                        .frameOptions(frameOptions -> frameOptions.disable()));

        return http.build();

    }

**Run The App -> You get error**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**APPLICATION FAILED TO START**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Description:**

**Parameter 0 of method authManager in org.studyeasy.SpringRestdemo.security.SecurityConfig required a bean of type 'org.springframework.security.core.userdetails.UserDetailsService' that could not be found.**

**Action:**

**Consider defining a bean of type 'org.springframework.security.core.userdetails.UserDetailsService' in your configuration.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**There should be on class which implements UserDetailsService and that class will be AccountService.java**

**Part 2**

**Update AccountService.java -> Implements UserDetailsService**

package org.studyeasy.SpringRestdemo.service;

import java.util.ArrayList;

import java.util.List;

import java.util.Optional;

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

import org.springframework.security.core.GrantedAuthority;

import org.springframework.security.core.authority.SimpleGrantedAuthority;

import org.springframework.security.core.userdetails.User;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.core.userdetails.UsernameNotFoundException;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.stereotype.Service;

import org.studyeasy.SpringRestdemo.model.Account;

import org.studyeasy.SpringRestdemo.repository.AccountRepository;

@Service

public class AccountService implements UserDetailsService {

    @Autowired

    private AccountRepository accountRepository;

    @Autowired

    private PasswordEncoder passwordEncoder;

    public Account save(Account account) {

        account.setPassword(passwordEncoder.encode(account.getPassword()));

        return accountRepository.save(account);

    }

    @Override

    public UserDetails loadUserByUsername(String email) throws UsernameNotFoundException {

        Optional<Account> optionalAccount = accountRepository.findByEmail(email);

        if (!optionalAccount.isPresent()) {

             throw new UsernameNotFoundException("Account not found");

        }

        Account account = optionalAccount.get();

        List<GrantedAuthority> grantedAuthority = new ArrayList<>();

        grantedAuthority.add(new SimpleGrantedAuthority(account.getRole()));

        // Set<Authority> authorities = account.getAuthorities();

        // for (Authority \_auth : authorities) {

        //     grantedAuthority.add(new SimpleGrantedAuthority(\_auth.getName()));

        // }

         // return null;

        return new User(account.getEmail(), account.getPassword(), grantedAuthority);

    }

}

**Update AccountRepository.java**

package org.studyeasy.SpringRestdemo.repository;

import java.util.Optional;

import org.springframework.data.jpa.repository.JpaRepository;

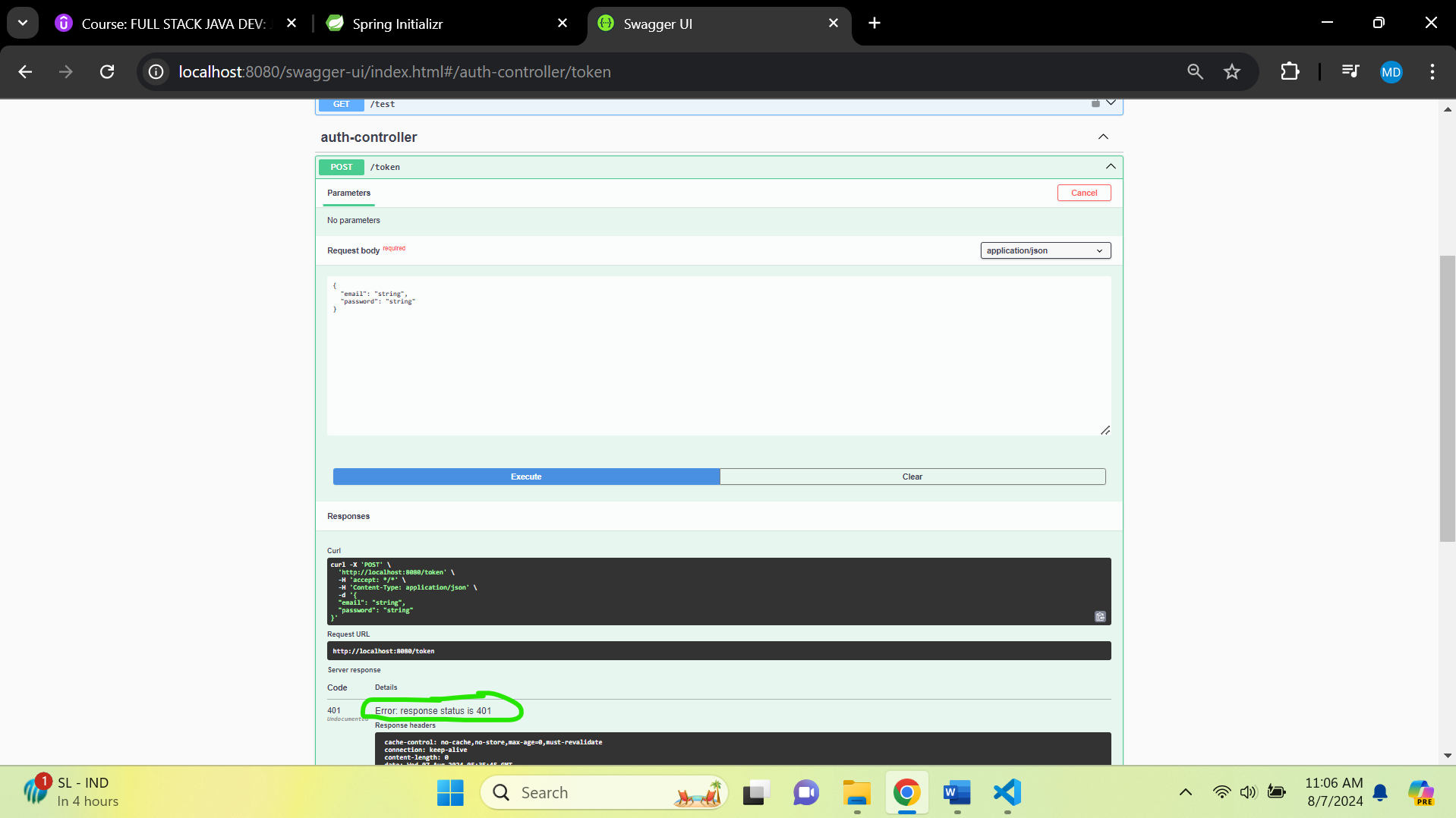
import org.studyeasy.SpringRestdemo.model.Account;

public interface AccountRepository extends JpaRepository<Account, Long> {

    Optional<Account> findByEmail(String email);

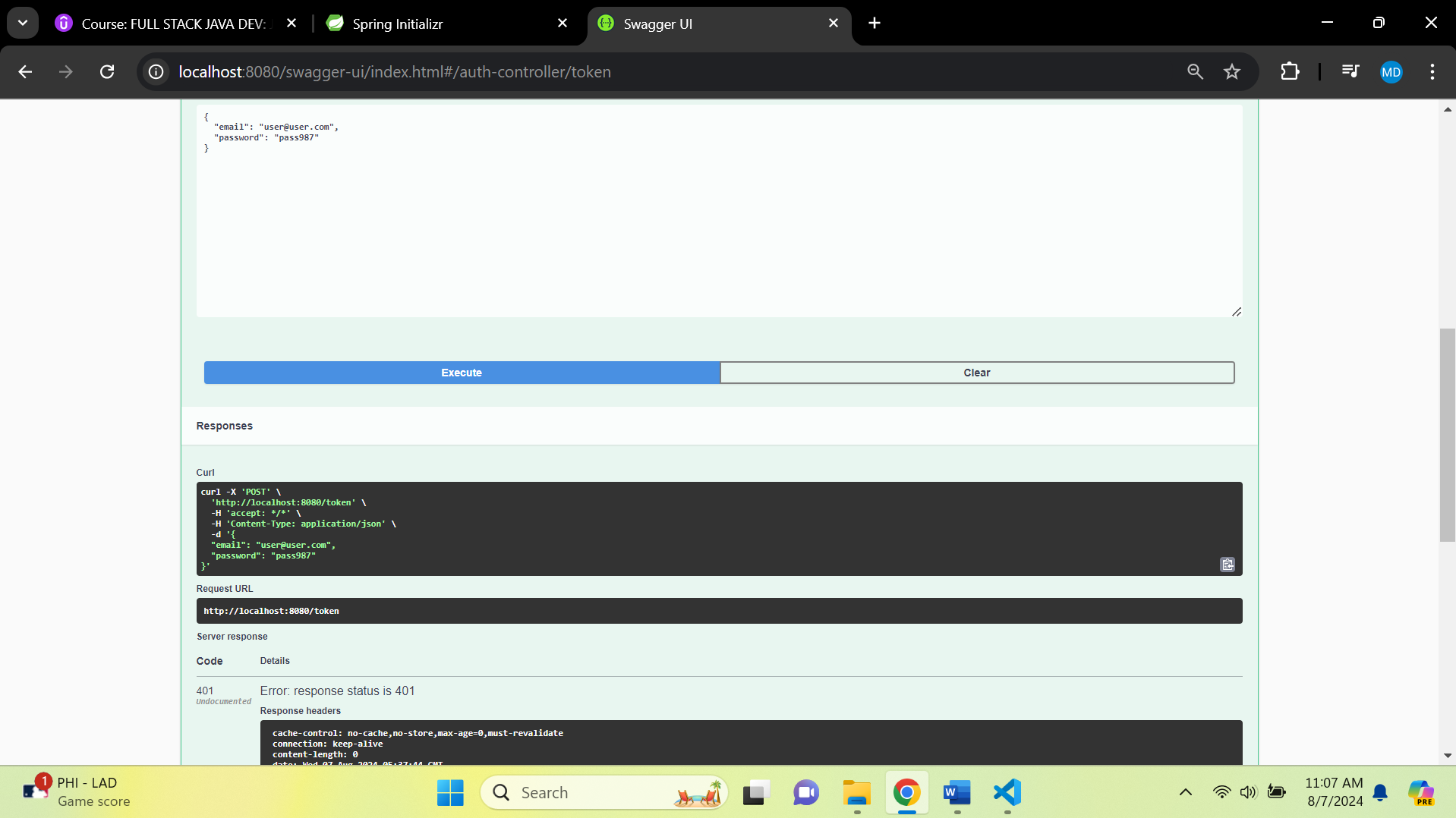
}

**Run the app**

****

**When you enter the details and run the api, you still get 401 error. Why? You also get error like this.**

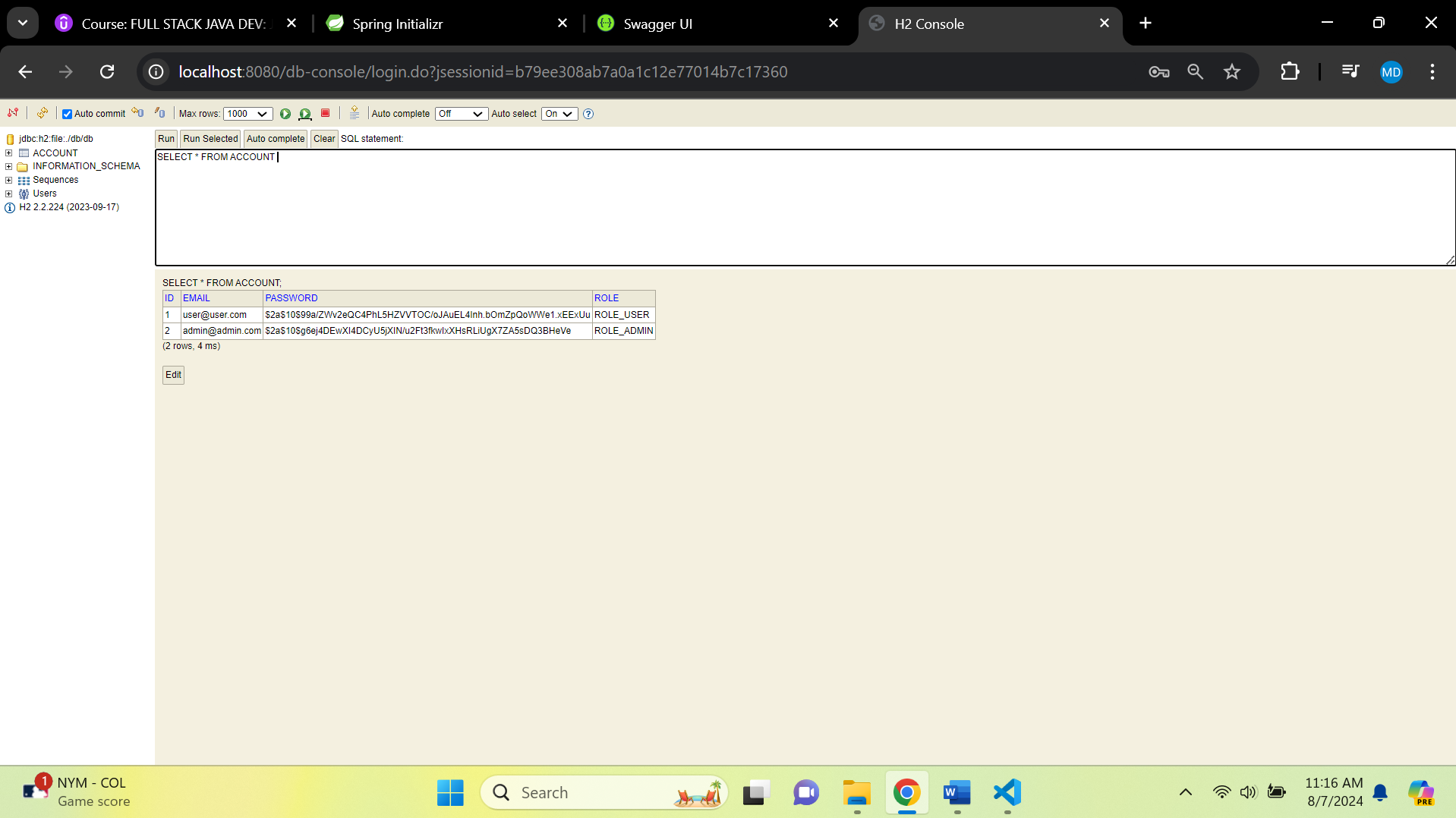
**java. lang. IllegalArgumentException: You have entered a password with no Password Encoder. If that is your intent, it should be prefixed with `{noop}`.**

****

**You missed this line in SecurityConfig.java in securityFilterChain method**

.requestMatchers("/db-console/\*\*").permitAll()

**DB Console**

****

**In SecurityConfig.java -> modify authManager() method**

@Bean

    public AuthenticationManager authManager(UserDetailsService userDetailsService) {

        var authProvider = new DaoAuthenticationProvider();

        authProvider.setUserDetailsService(userDetailsService);

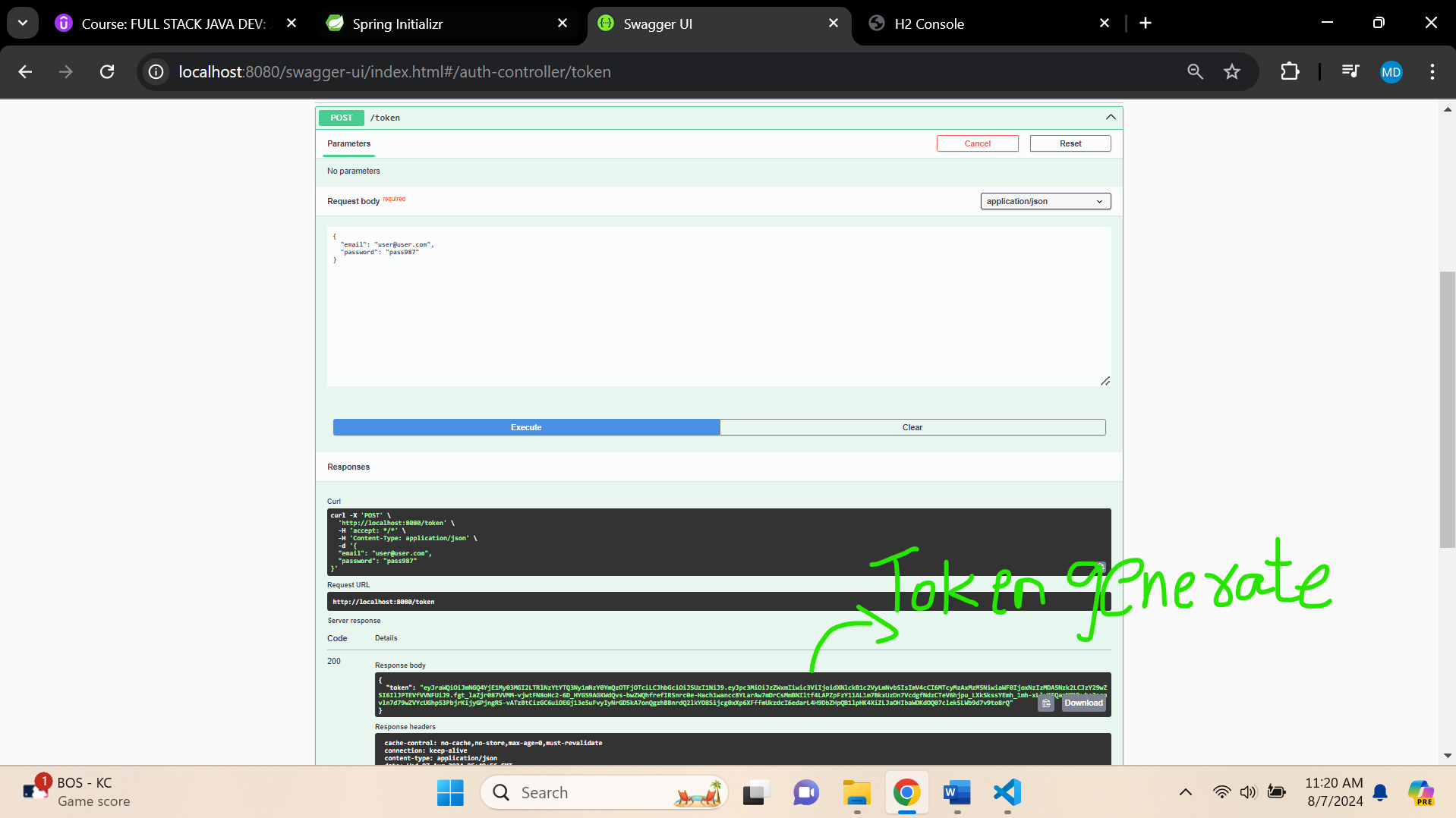
        return new ProviderManager(authProvider);

    }

**Add this line**

 authProvider.setPasswordEncoder(passwordEncoder());

**Token Generation from frontend**

****

**Sample Token Generated displayed in the frontend**

**eyJraWQiOiJmNGQ4YjE1My03MGI2LTRlNzYtYTQ3Ny1mNzY0YmQzOTFjOTciLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJzZWxmIiwic3ViIjoidXNlckB1c2VyLmNvbSIsImV4cCI6MTcyMzAxMzM5NiwiaWF0IjoxNzIzMDA5Nzk2LCJzY29wZSI6IlJPTEVfVVNFUiJ9.fgt\_laZjr087VVMM-vjwtFN8oHc2-6D\_HYGS9AGKWdQvs-bwZWQhfrefIRSnrc0e-Hach1wancc8YLarAw7mDrCsMmBNIltf4LAPZpFzY11AL1m7BkxUzDn7VcdgfNdzCTeV6hjpu\_LXkSkssYEmh\_1mh-xLlyKEQazNQYbwho2oqavln7d79wZVYcUGhp53PbjrKijyGPjngR5-vATzBtCizGC6uiOEGj13e5uFvyIyNrGD5kA7onQgzhBBnrdQ2lkYO8Sijcg0xXp6XFffmUkzdcI6edarL4H9DbZHpQB1lpHK4XiZLJaOHIbaWOKdOQ07clekSLWb9d7v9to8rQ**