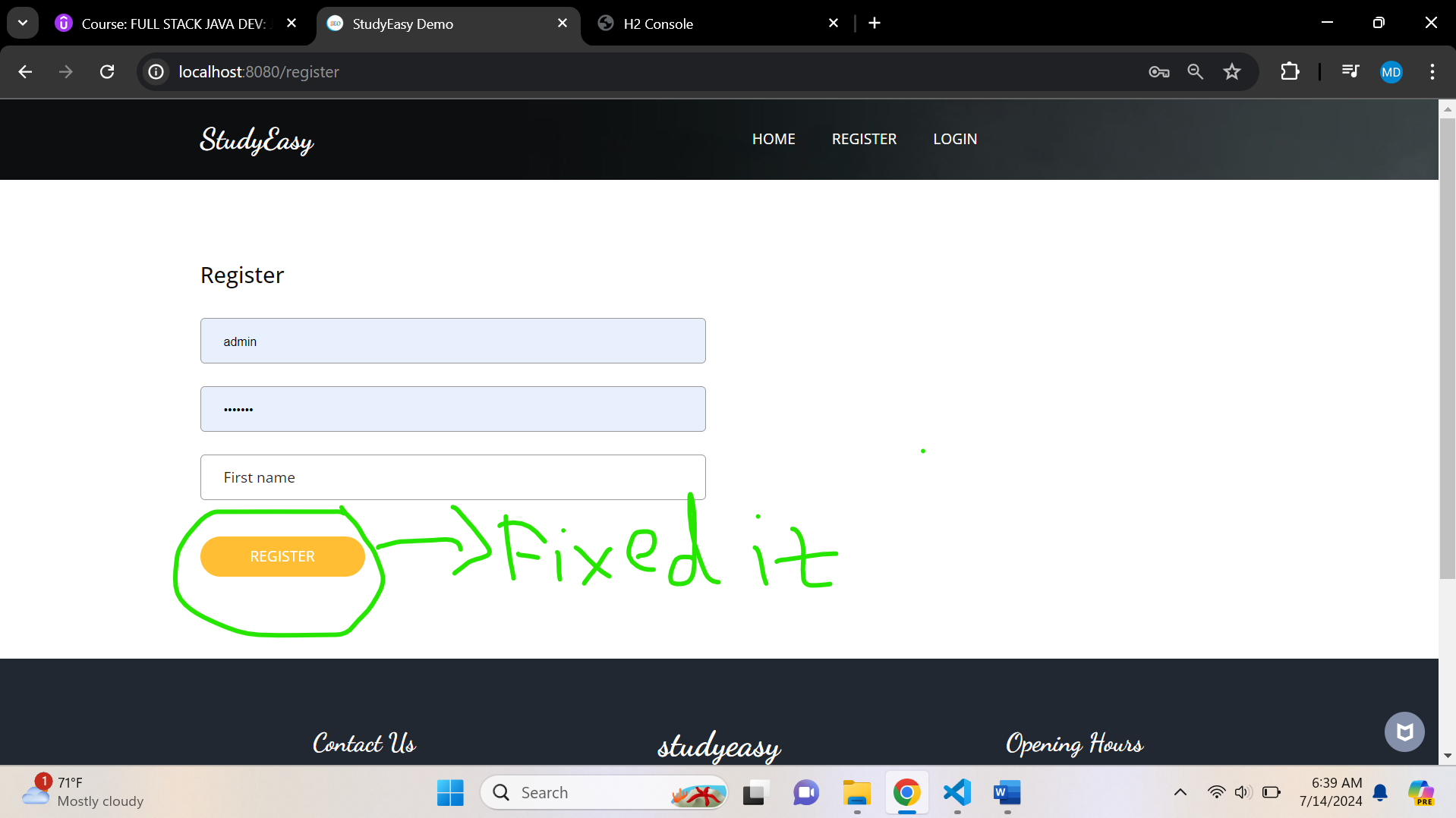
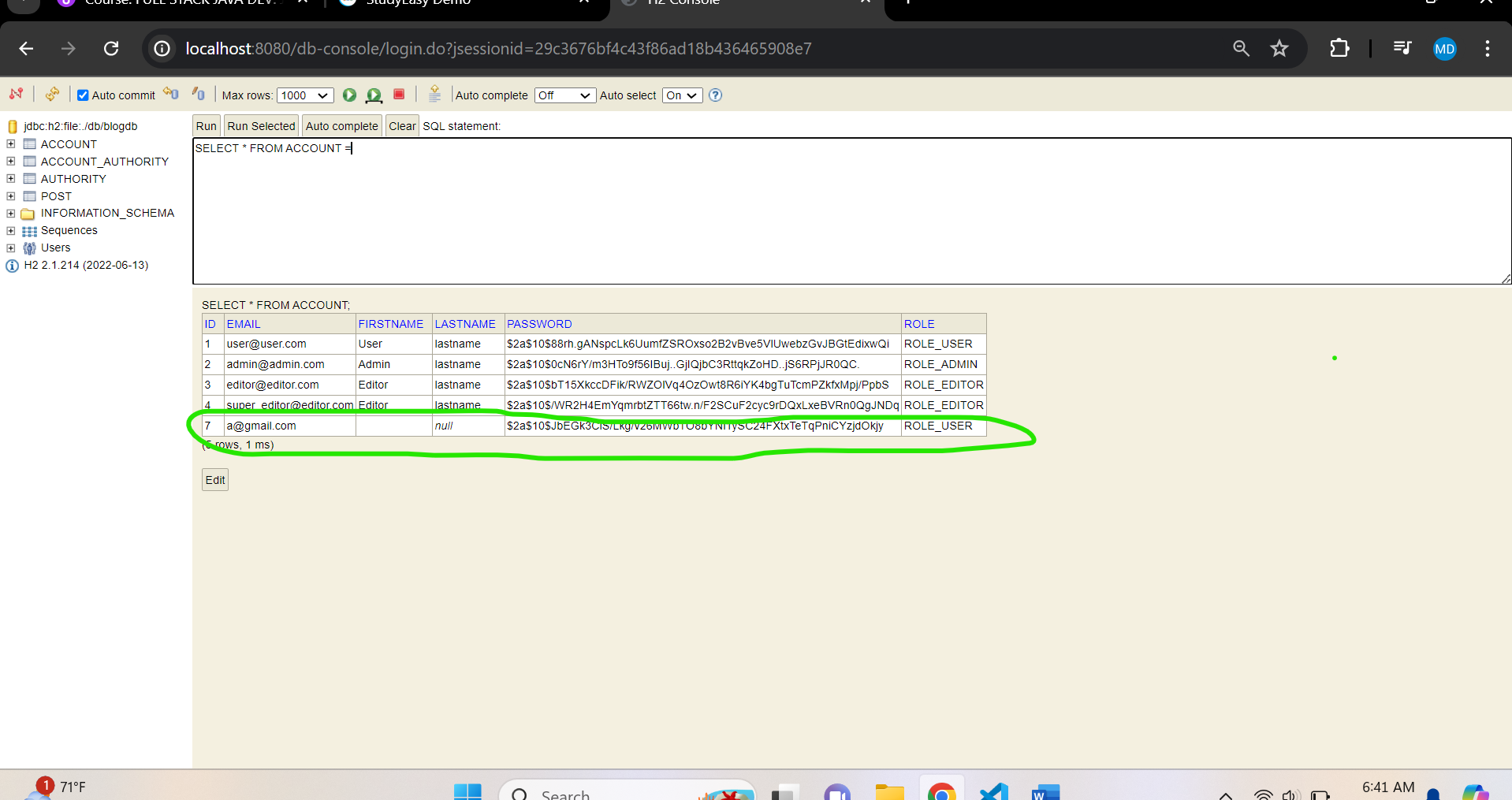
**SpringBoot\_Add Validation Model In Register Form In Spring Boot**

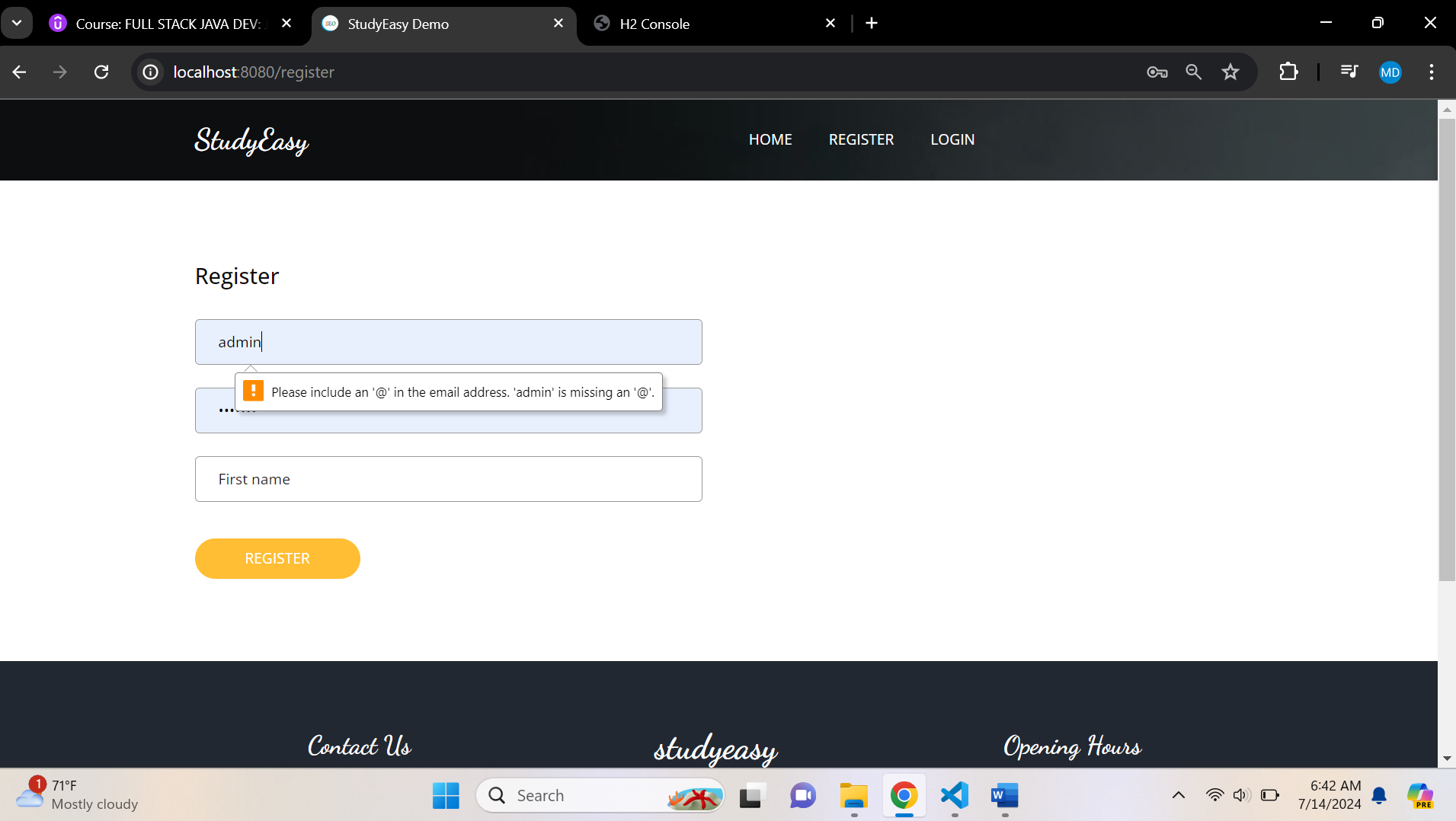
**Screen**

****

**There was an error coming on the button, so fixed it. And When you click on Register button, it will be redirected home page. In Db, it will be added like below.**

****

**After changing, validation is not a problem like below.**

****

**These are known as Client-Side Validations i.e. handled by the UI side.**

**This is a Monolithic Architecture where front and back end are combined. But in microservices it is different.**

**Server-Side Validations**

**Add a dependency: -**

<!--

        https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-validation -->

<dependency>

   <groupId>org.springframework.boot</groupId>

  <artifactId>spring-boot-starter-validation</artifactId>

   <version>3.3.1</version>

</dependency>

**Validations – Email and password**

@Email(message = "Invalid email")

    @NotEmpty(message = "Email missing")

    private String email;

    @NotEmpty(message = "Missing password")

    private String password;

**In AccountController**

 @PostMapping("/register")

    public String register\_user(@ModelAttribute Account account){

        accountService.save(account);

        return "redirect:/";

    }

**Update to**

 @PostMapping("/register")

    public String register\_user(@Valid @ModelAttribute Account account, BindingResult result){

        if(result.hasErrors()){

            return "account\_views/register";

        }

        accountService.save(account);

        return "redirect:/";

    }

**BindingResult – It binds all the error possible in this object**

**Error Validation Code**

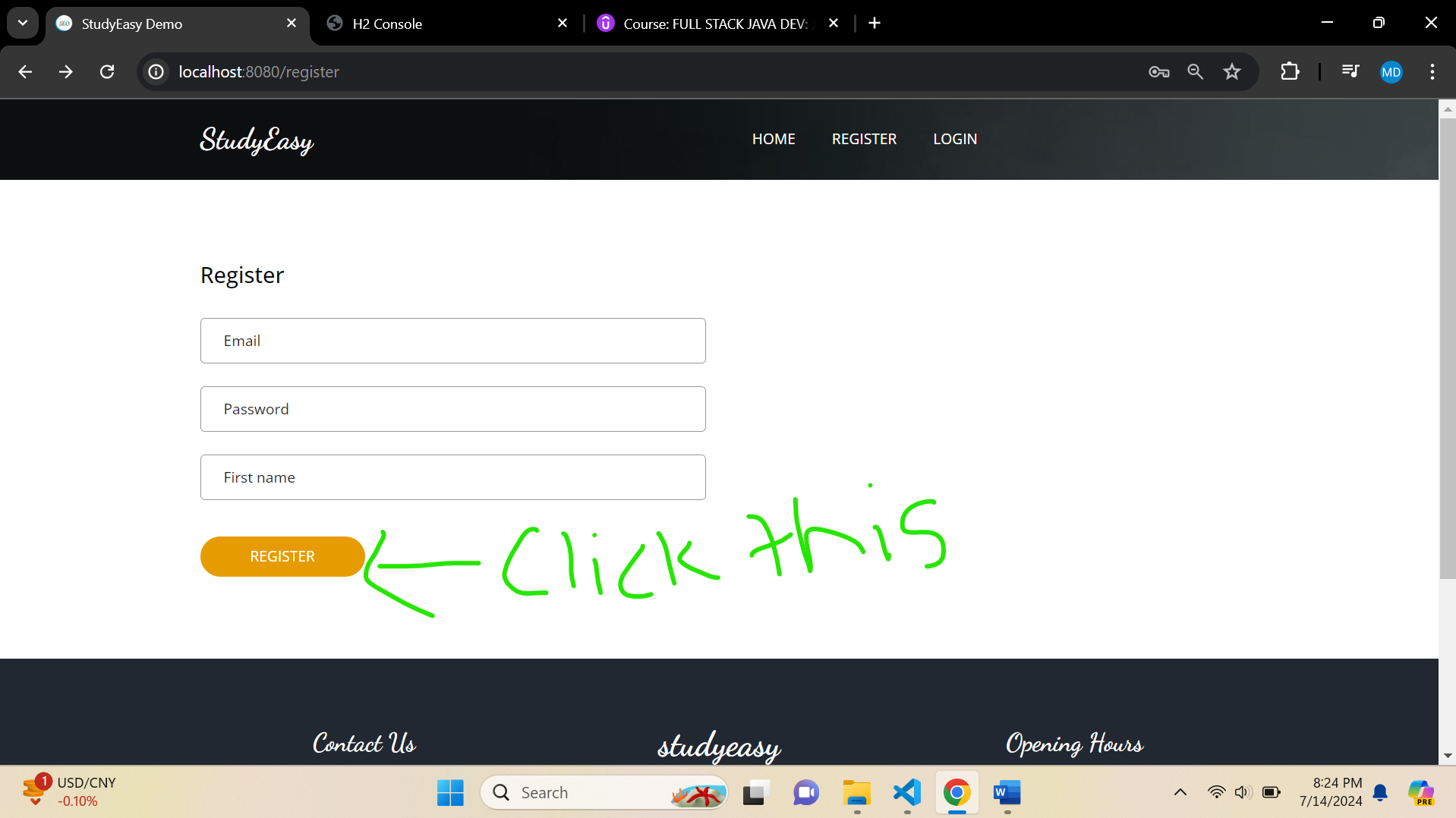
<div>

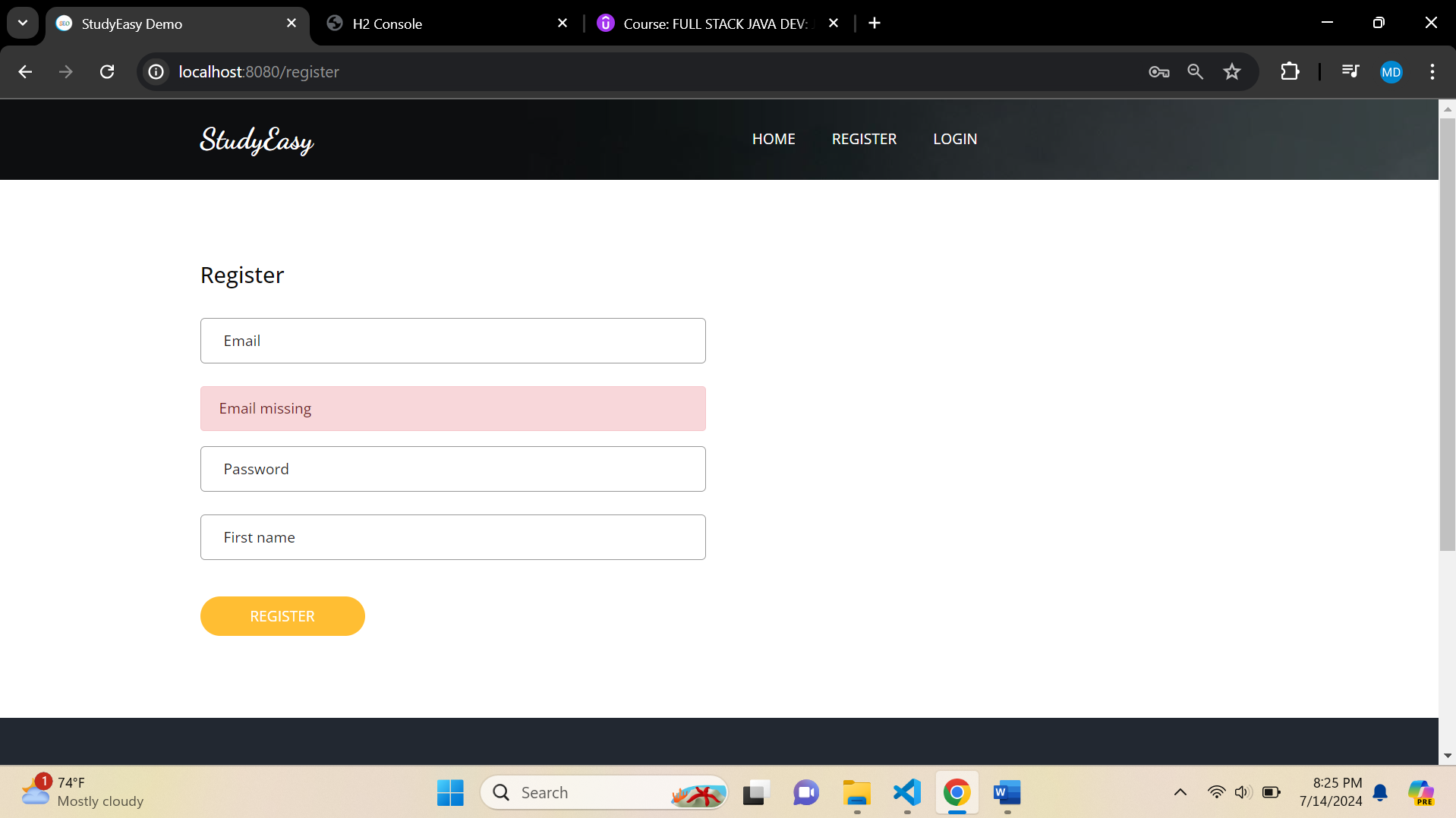
   <input id="email" type="text" th:field="\*{email}" class="form-control" placeholder="Email" />

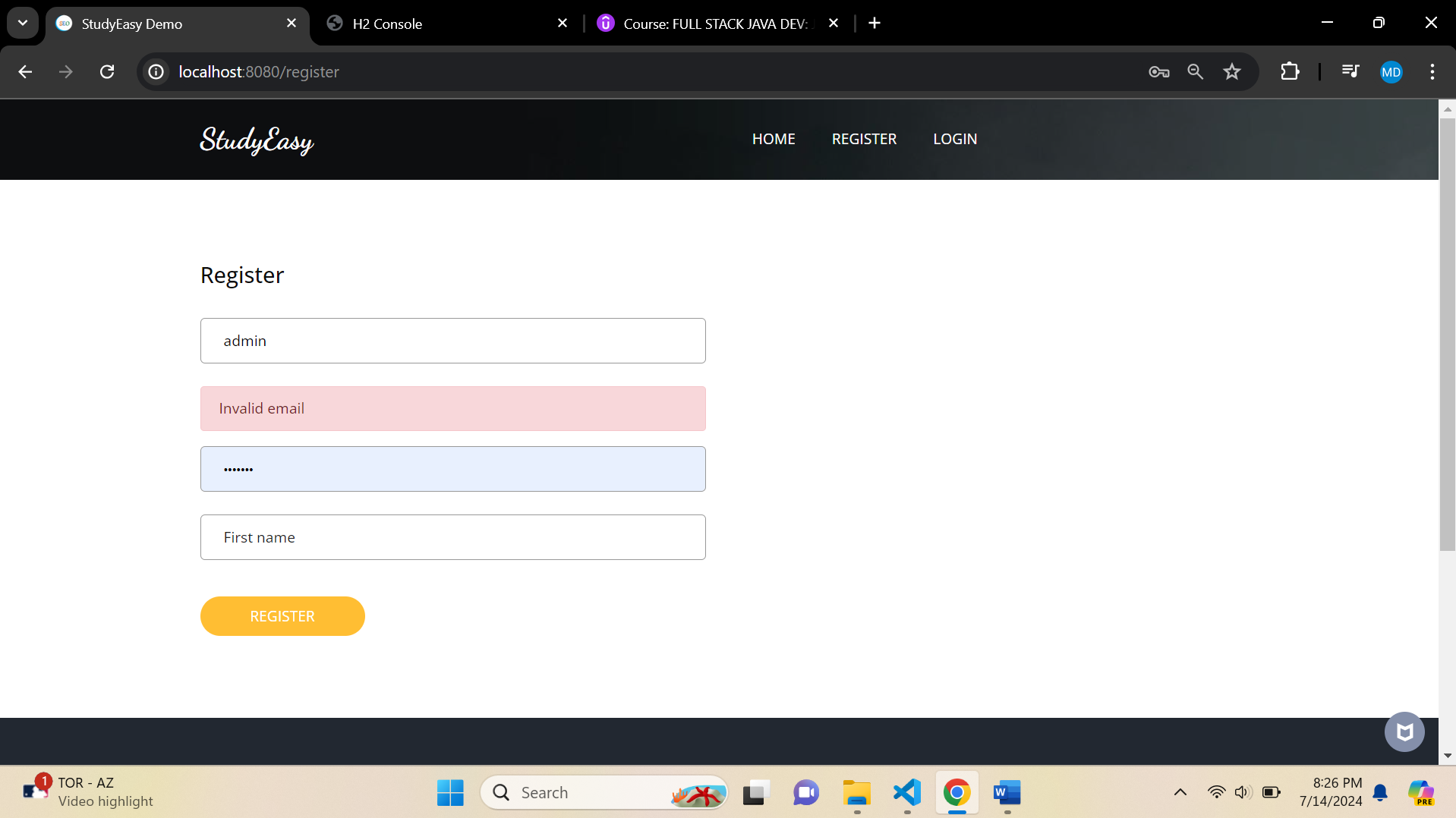
  <p th:if="${#fields.hasErrors('email')}" th:errors="\*{email}" class =”alert-danger”>Email Error</p>

</div>

**Output**

****

****

****