[Link Web Stranice](https://www.fym.it/blog/scrivere-email-di-lavoro/computer-2/)

Kemal Mešić – 236-ST

Zadaća 4

Testiranje Softvera

Sadržaj

[Zadatak 1 2](#_Toc220005296)

[1.1 Korištene metrike (Visual Studio – Code Metrics) 2](#_Toc220005297)

[1.2 Odabrani dio koda za refaktoring 2](#_Toc220005298)

[1.3 Metrike prije refaktoringa (primjer) 2](#_Toc220005299)

[1.4 Refaktoring – razdvajanje odgovornosti 3](#_Toc220005300)

[1.5 Metrike poslije refaktoringa 4](#_Toc220005301)

[1.6 Zaključak 4](#_Toc220005302)

[Zadatak 2 5](#_Toc220005303)

[2.1 Uvod 5](#_Toc220005304)

[2.2 Opis aplikacije Zovi.ba 5](#_Toc220005305)

[2.3 Metodologija testiranja 5](#_Toc220005306)

[2.4 Test osnovnih informacija o korisnicima 5](#_Toc220005307)

[2.4.1 Opis testa 5](#_Toc220005308)

[2.4.2. Postavljena pitanja 6](#_Toc220005309)

[2.4.3. Rezultati 6](#_Toc220005310)

[2.5 Test dizajna korisničkog interfejsa 6](#_Toc220005311)

[2.5.1. Opis testa 6](#_Toc220005312)

[2.5.2. Postavljena pitanja 6](#_Toc220005313)

[2.5.3. Rezultati 7](#_Toc220005314)

[2.6 Test prvog klika (First Click Test) 7](#_Toc220005315)

[2.6.1. Opis testa 7](#_Toc220005316)

[2.6.2. Zadatak korisnika 7](#_Toc220005317)

[2.6.3. Rezultati 8](#_Toc220005318)

[Zadatak 3 9](#_Toc220005319)

[3.1. HTTP BASICS 10](#_Toc220005320)

[3.2. HTTP Proxies 11](#_Toc220005321)

[3.3. Developer Tools 12](#_Toc220005322)

[3.4. CIA Triad 14](#_Toc220005323)

[3.5. SQL INJECTION(intro) 15](#_Toc220005324)

[3.6. SQL Injection (mitigation) 20](#_Toc220005325)

# Zadatak 1

## Korištene metrike (Visual Studio – Code Metrics)

U Visual Studio alatu **Analyze → Calculate Code Metrics** mjerene su sljedeće metrike:

* **Cyclomatic Complexity (CC)** – složenost toka programa
* **Maintainability Index (MI)** – koliko je kod lak za održavanje
* **Class Coupling (CCoupling)** – zavisnost klase od drugih klasa
* **Lines of Code (LOC)** – broj linija koda

## 1.2 Odabrani dio koda za refaktoring

Za refaktoring je odabrana metoda:

static void BookMenu(LibraryInventory inventory)

Razlozi:

* Metoda je **predugačka**
* Ima **više switch/case grana**
* Ima **visoku ciklomatsku složenost**
* Krši **Single Responsibility Principle**
* Direktno miješa:
  + UI logiku (Console)
  + poslovnu logiku
  + validaciju unosa

## 1.3 Metrike prije refaktoringa (primjer)

|  |  |  |  |
| --- | --- | --- | --- |
| Metoda | CC | MI | LOC |
| BookMenu | 12 | 58 | ~95 |

**Problemi:**

* CC > 10 → metoda je previše složena
* MI ispod 65 → teže održavanje
* Velik broj linija → loša čitljivost

## Refaktoring – razdvajanje odgovornosti

Cilj refaktoringa:

* Smanjiti Cyclomatic Complexity
* Povećati Maintainability Index
* Povećati čitljivost i testabilnost

Refaktoring pristup:

Svaka opcija iz menija se izdvaja u **posebnu metodu**.

Refaktorisani primjer

static void BookMenu(LibraryInventory inventory)

{

while (true)

{

PrintBookMenu();

string opcija = Console.ReadLine();

if (opcija == "0") return;

switch (opcija)

{

case "1": AddBook(inventory); break;

case "2": UpdateBook(inventory); break;

case "3": DeleteBook(inventory); break;

case "4": SearchBooks(inventory); break;

case "5": ShowAllBooks(inventory); break;

case "6": ChangeAvailability(inventory); break;

}

}

}

Primjer izdvojene metode:

static void AddBook(LibraryInventory inventory)

{

Console.Write("Naslov: ");

string title = Console.ReadLine();

Console.Write("Autor: ");

string author = Console.ReadLine();

Console.Write("Žanr: ");

string genre = Console.ReadLine();

inventory.AddBook(title, author, genre);

Console.WriteLine("Knjiga dodana!");

}

## Metrike poslije refaktoringa

|  |  |  |  |
| --- | --- | --- | --- |
| Metoda | CC | MI | LOC |
| BookMenu | 4 | 78 | ~30 |
| AddBook | 2 | 90 | ~15 |
| UpdateBook | 3 | 88 | ~20 |

Poboljšanja:

* Cyclomatic Complexity značajno smanjen
* Maintainability Index znatno veći
* Kraće i jasnije metode
* Lakše testiranje i buduće izmjene

## Zaključak

Refaktoring je doveo do **značajnog poboljšanja kvaliteta koda**.  
Funkcionalnost aplikacije se nije promijenila, ali kod je postao čitljiviji, lakši za održavanje, manje sklon greškama i bolje organizovan.

Najveće poboljšanje je ostvareno u metrikama **Cyclomatic Complexity** i **Maintainability Index**, što potvrđuje opravdanost refaktoringa.

# Zadatak 2

## 2.1 Uvod

Cilj ovog zadatka je izvršiti testiranje upotrebljivosti odabrane web aplikacije korištenjem alata za testiranje upotrebljivosti. Kao testirana aplikacija odabrao sam svoju web aplikaciju **Zovi.ba**, koja predstavlja novi online oglasnik namijenjen pregledu i objavi oglasa, prvenstveno vozila.

Testiranje je izvršeno korištenjem platforme **Lyssna (bivši UsabilityHub)**, koja omogućava prikupljanje povratnih informacija korisnika o dizajnu i funkcionalnosti korisničkog interfejsa.

## 2.2 Opis aplikacije Zovi.ba

Zovi.ba je web aplikacija koja korisnicima omogućava:

* pregled i pretragu oglasa,
* filtriranje oglasa prema različitim kriterijima,
* pregled detalja oglasa,
* kontakt s oglašivačem,
* registraciju i prijavu korisnika radi objave oglasa.

Aplikacija je namijenjena širokom spektru korisnika i dizajnirana je tako da bude jednostavna za korištenje.

## 2.3 Metodologija testiranja

Testiranje upotrebljivosti provedeno je korištenjem platforme **Lyssna**, pri čemu su korištena tri različita tipa testova:

* test osnovnih informacija o korisnicima,
* test dizajna korisničkog interfejsa,
* test prvog klika (First Click Test).

Testovi su podijeljeni korisnicima putem generisanih linkova, a rezultati su analizirani kroz statističke prikaze koje platforma automatski generiše.

## 2.4 Test osnovnih informacija o korisnicima

### ****2.4.1 Opis testa****

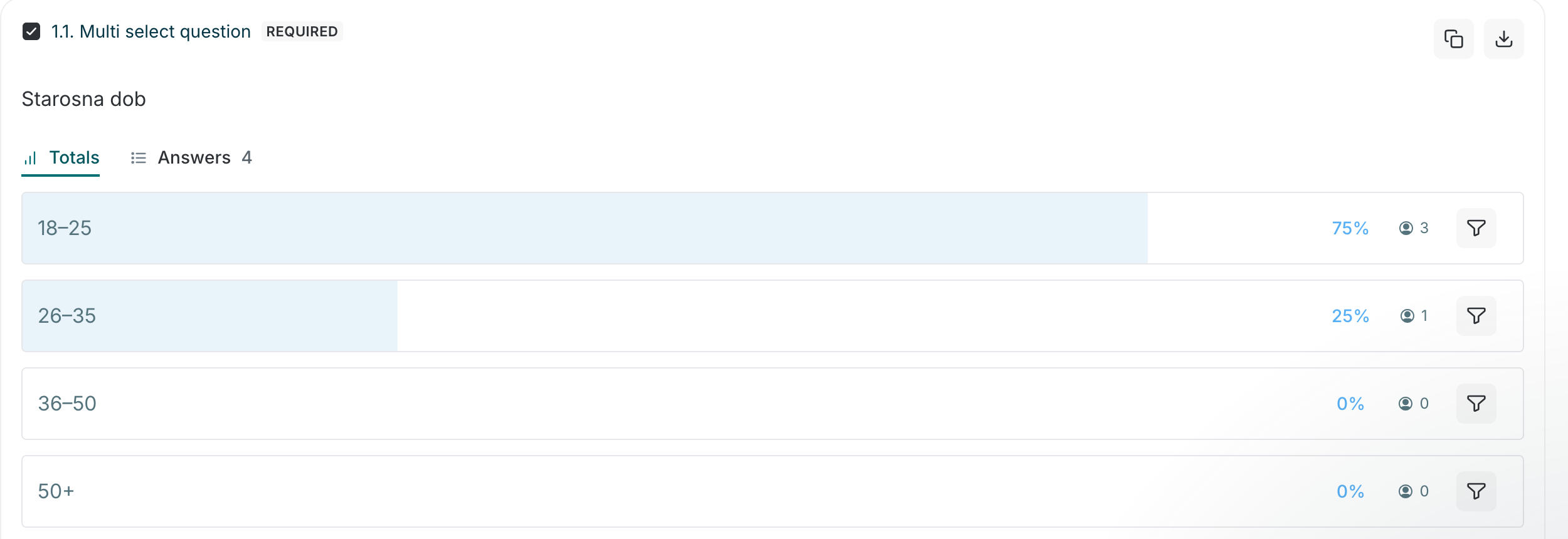
Test osnovnih informacija ima za cilj prikupljanje osnovnih podataka o korisnicima koji su učestvovali u testiranju.

### 2.4.2. Postavljena pitanja

1. Kojoj starosnoj grupi pripadate?
2. Nivo obrazovanja.
3. Zanimanje.
4. Nivo tehničke pismenosti (1–5).
5. Poznavanje engleskog jezika (1–5).

### 2.4.3. Rezultati

Na osnovu prikupljenih odgovora može se zaključiti da većina testnih korisnika pripada starosnoj grupi od 18 do 35 godina te posjeduje srednji ili visok nivo tehničke pismenosti.

  
Slika 1. Prikaz odgovora Starosne dobi

## 2.5 Test dizajna korisničkog interfejsa

### 2.5.1. Opis testa

Cilj ovog testa bio je procijeniti jasnoću dizajna i razumljivost korisničkog interfejsa aplikacije Zovi.ba.

### 2.5.2. Postavljena pitanja

 Da li je jasna namjena aplikacije?

 Koliko vam je dizajn aplikacije privlačan (1–5)?

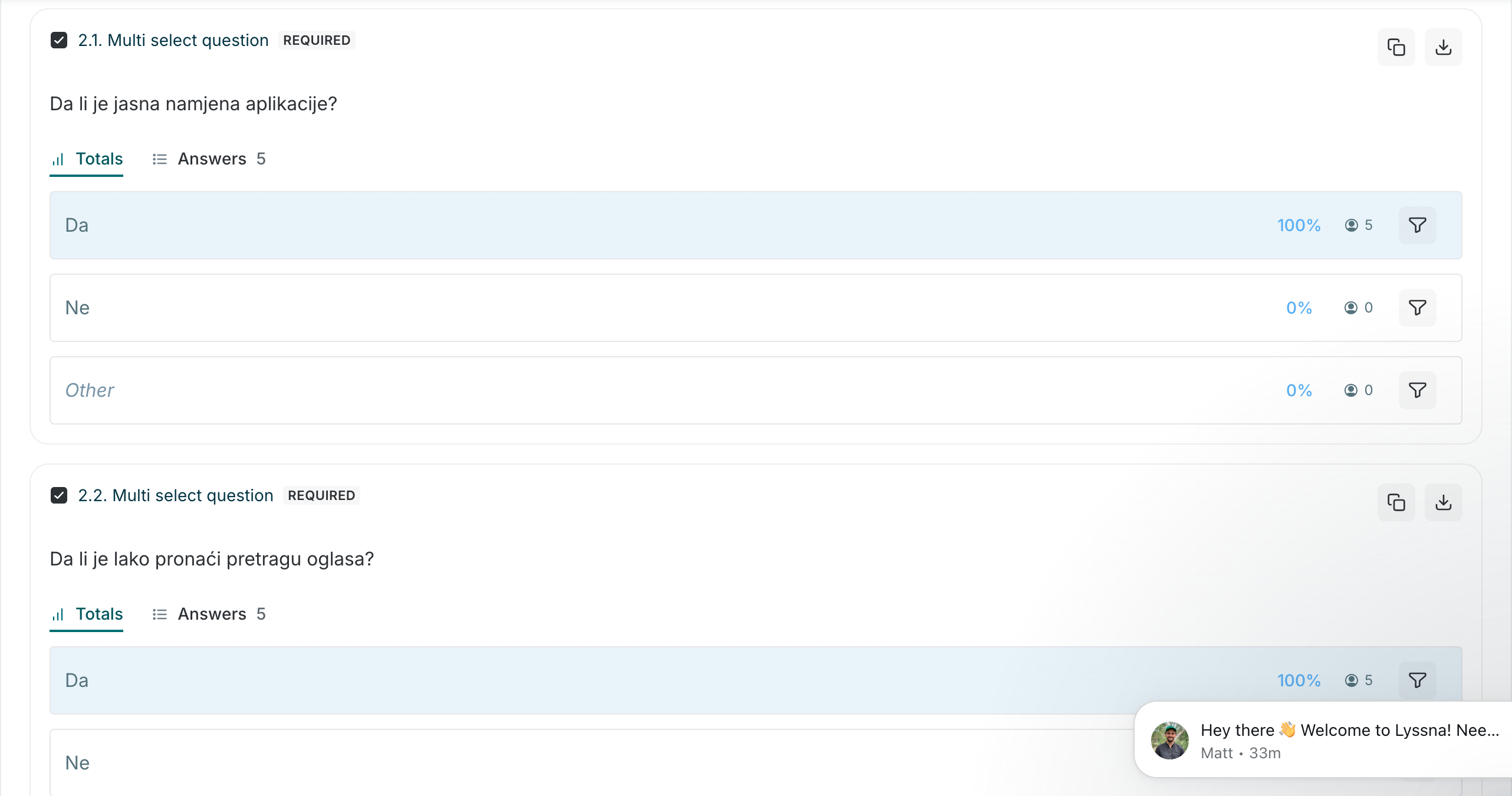
 Da li je lako pronaći opciju za pretragu oglasa?

 Da li su filteri razumljivi?

 Da li je tekst čitljiv i razumljiv?

### 2.5.3. Rezultati

Većina korisnika je navela da je namjena aplikacije jasna te da je dizajn aplikacije vizuelno prihvatljiv.

  
Slika 2. Prikaz rezultata dizajn testa

## 2.6 Test prvog klika (First Click Test)

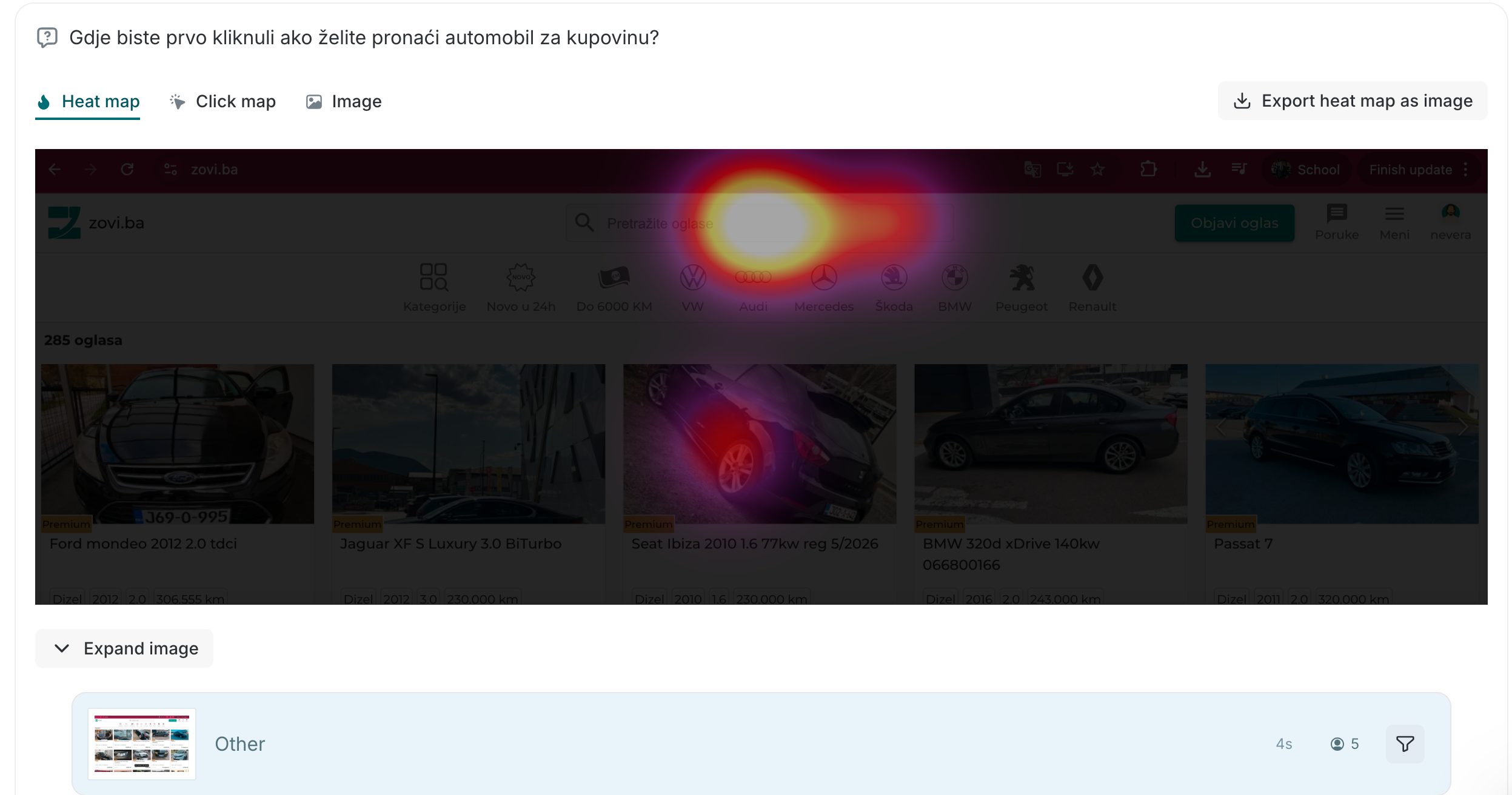
### 2.6.1. Opis testa

Test prvog klika korišten je kako bi se analiziralo na koji dio početnog korisničkog interfejsa korisnici prvo obraćaju pažnju.

**Tip testa:** First Click Test

### 2.6.2. Zadatak korisnika

„Gdje biste prvo kliknuli ako želite pronaći automobil za kupovinu?“

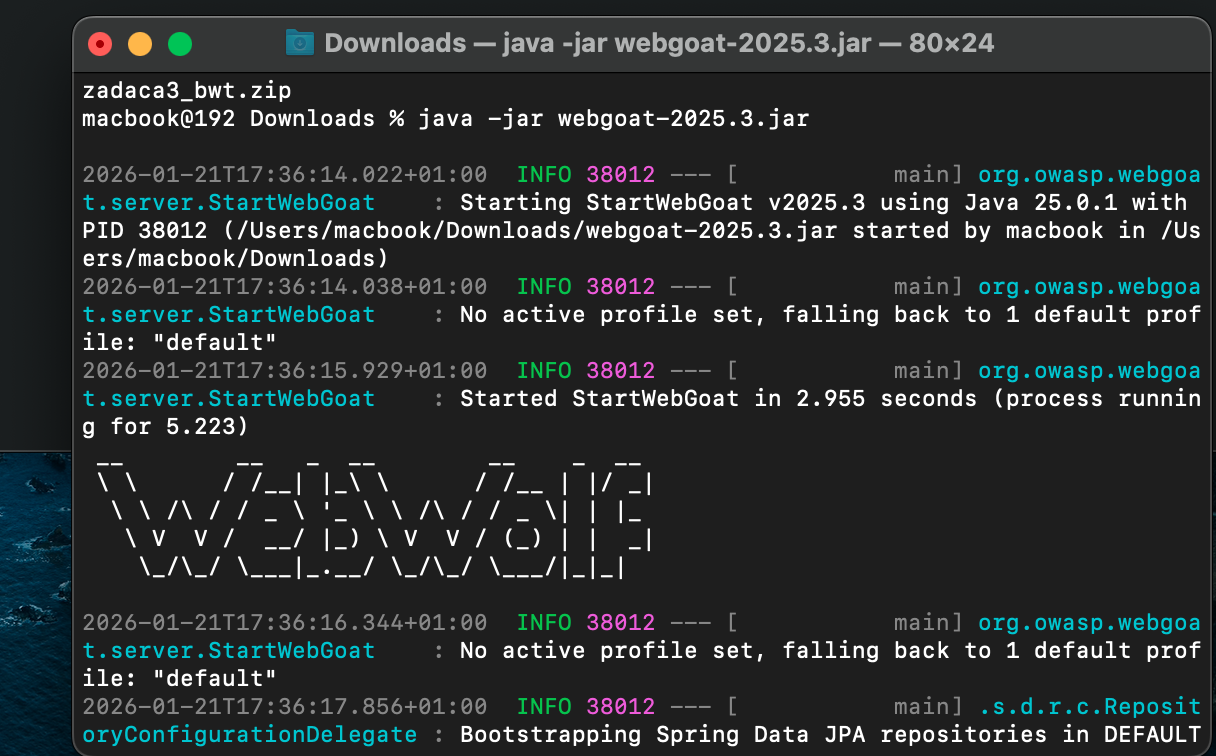
  
Slika 3. Heatmap screenshot iz Lyssna

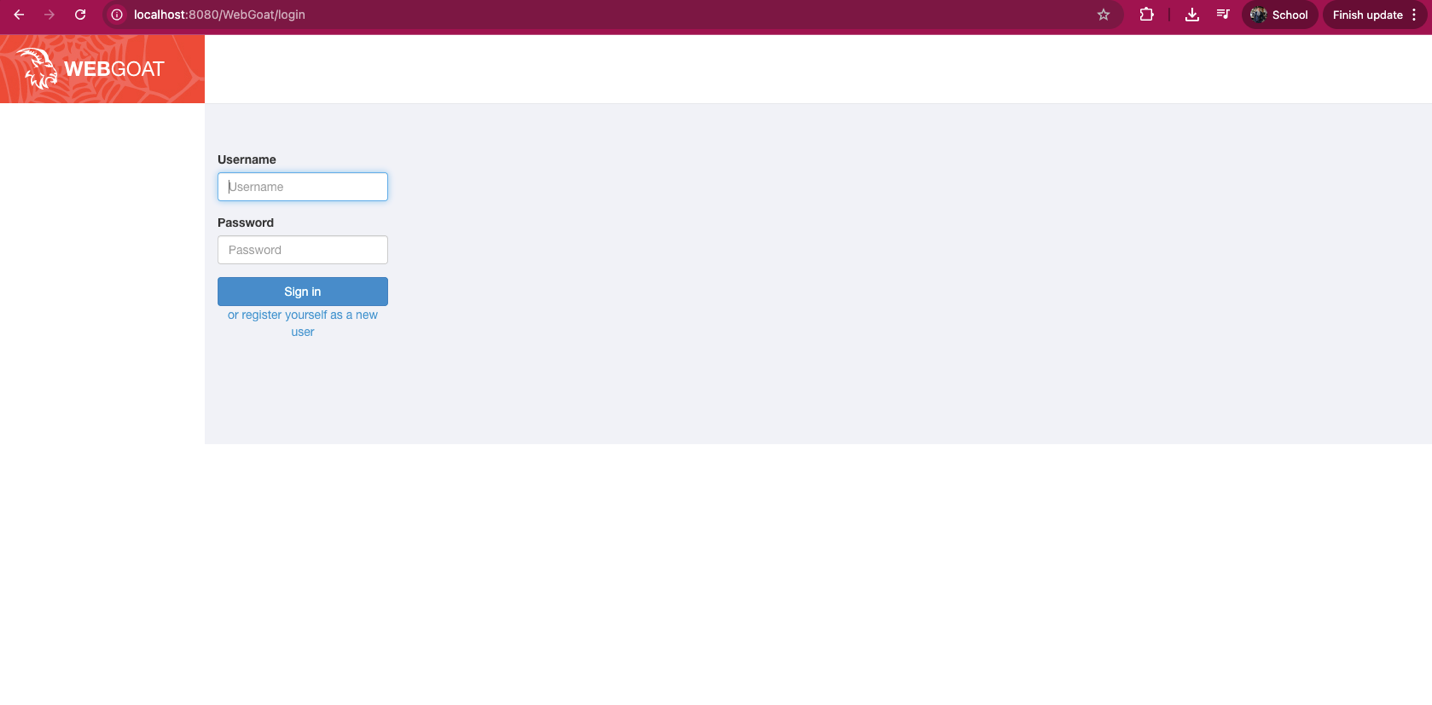
### 2.6.3. Rezultati

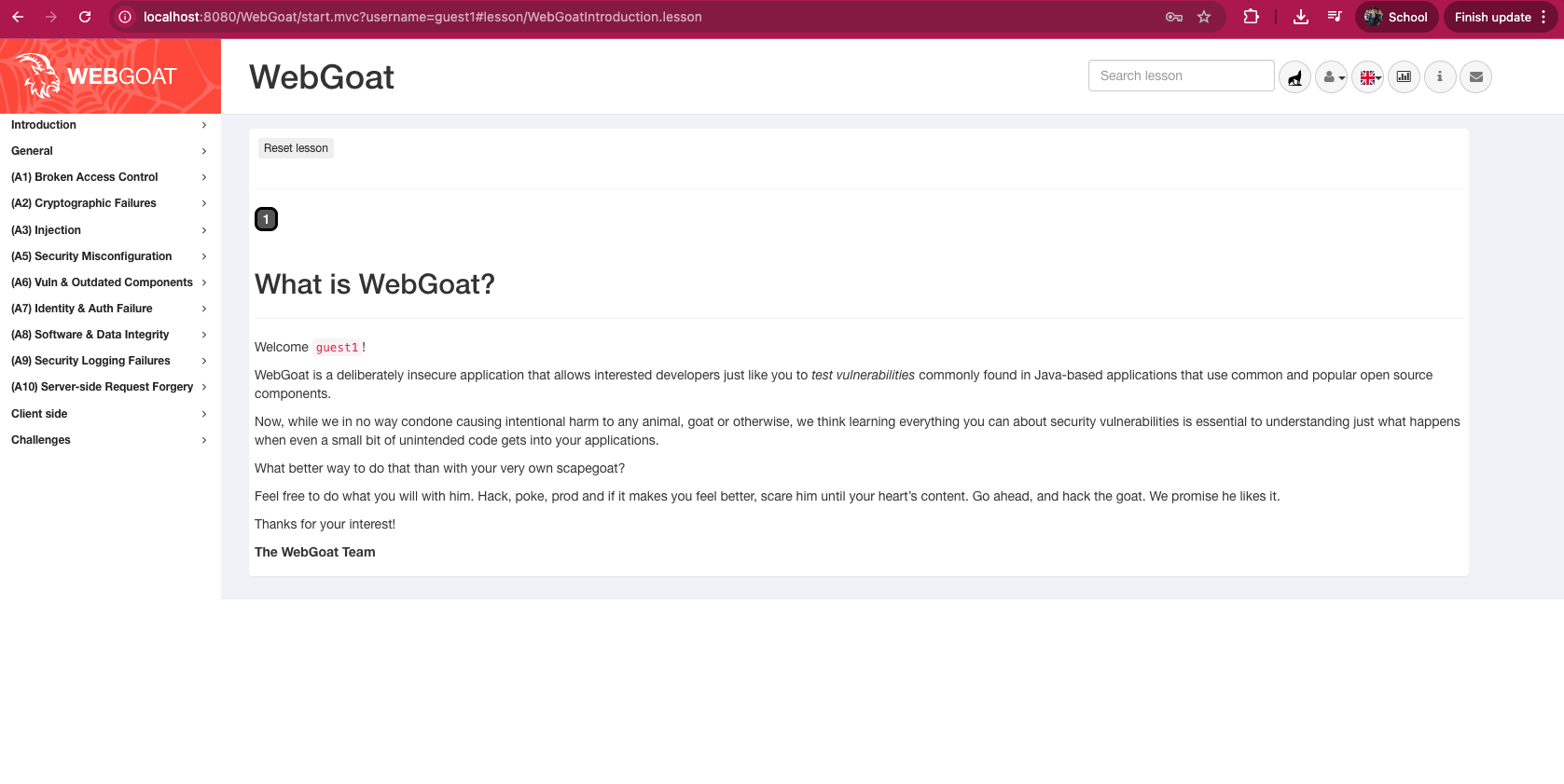
Rezultati testa pokazuju da je većina korisnika prvi klik izvršila na polje za pretragu ili listu oglasa, što ukazuje da početni interfejs aplikacije jasno usmjerava korisnike na glavnu funkcionalnosti.

# Zadatak 3

**WebGoat** je namjerno ranjiva web aplikacija koju je razvila organizacija **OWASP** s ciljem edukacije o web sigurnosti. Koristi se za učenje i praktično vježbanje najčešćih sigurnosnih propusta, poput SQL injectiona, XSS-a, autentikacijskih grešaka i upravljanja sesijama. WebGoat je namijenjen studentima i developerima kako bi u kontrolisanom okruženju razumjeli kako napadi funkcionišu i kako ih spriječiti u stvarnim aplikacijama.

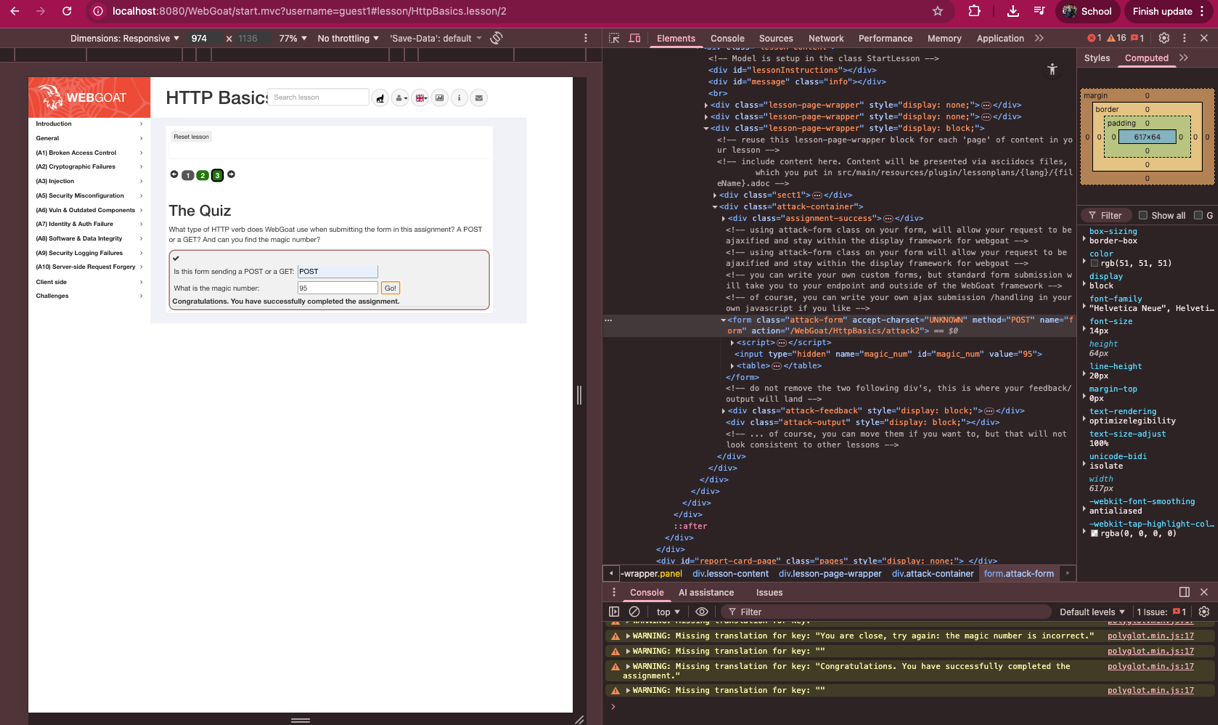
  
Slika 1. Prikaz terminala za instalaciju Webgoat-a

  
Slika 2. Prikaz login forme na webgoat – username i passwor guest1

  
Slika 3. Prikaz WebGoata

## 3.1. HTTP BASICS

Ovdje sam naučio HTTP metode (GET, POST),request / response i headers.

  
Slika 4. Uradjen dio HTTP Basics

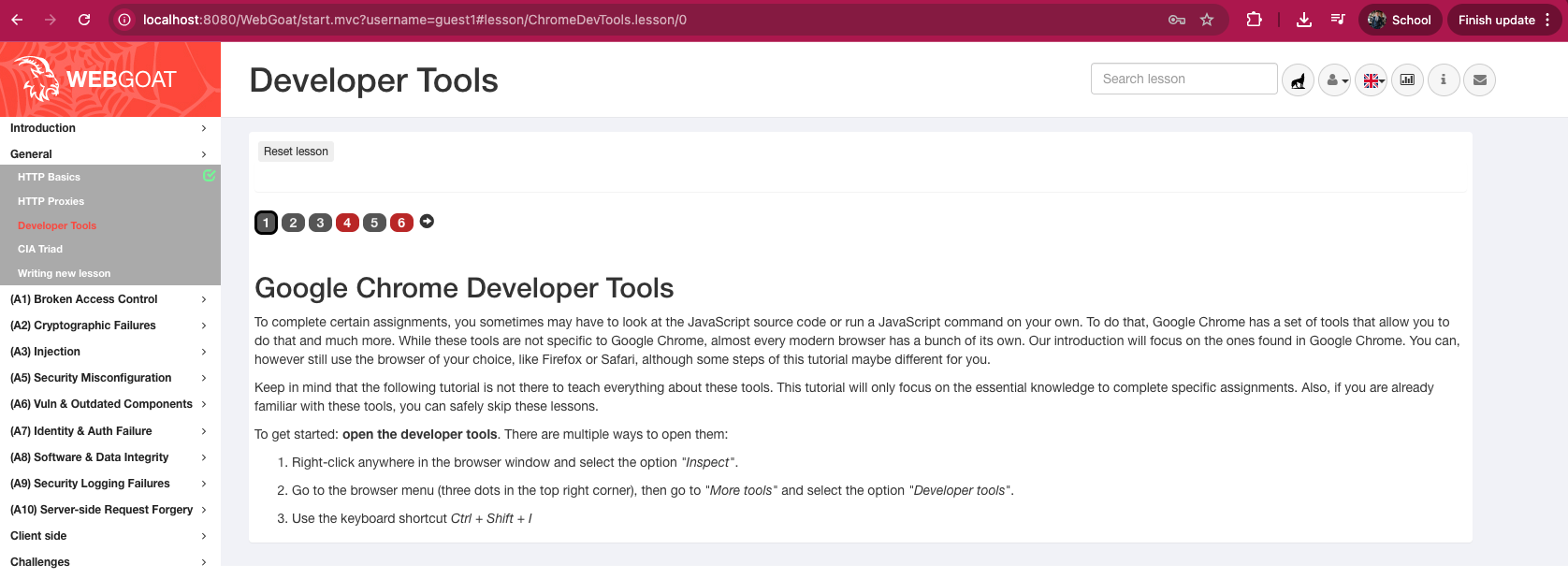
Korištena je metoda POST i magični broj je 95 kao što se vidi na slici 4.

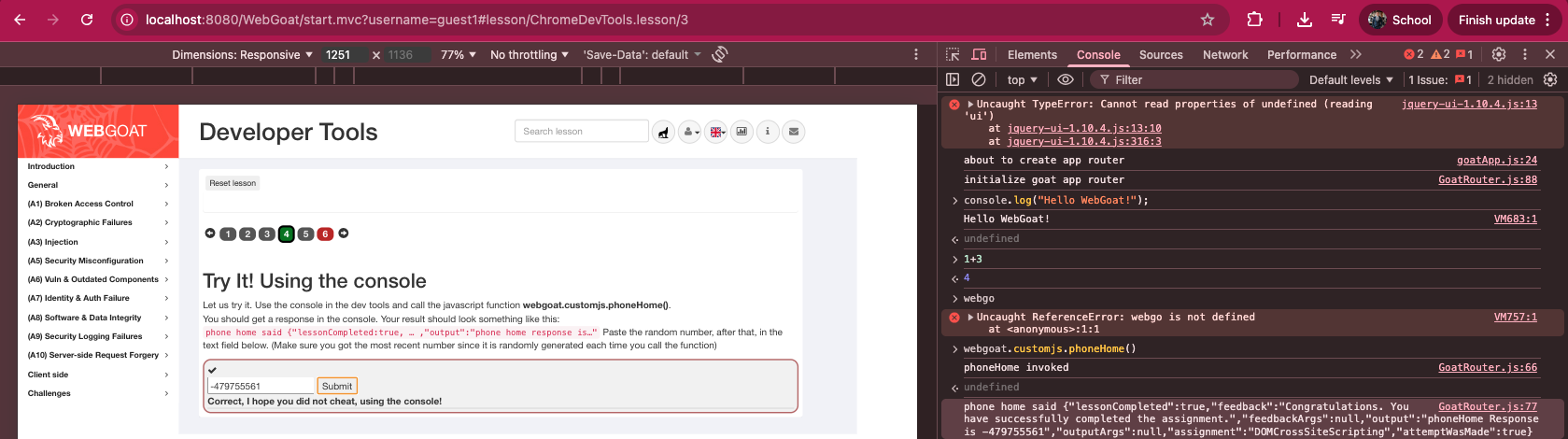
## 3.2. HTTP Proxies

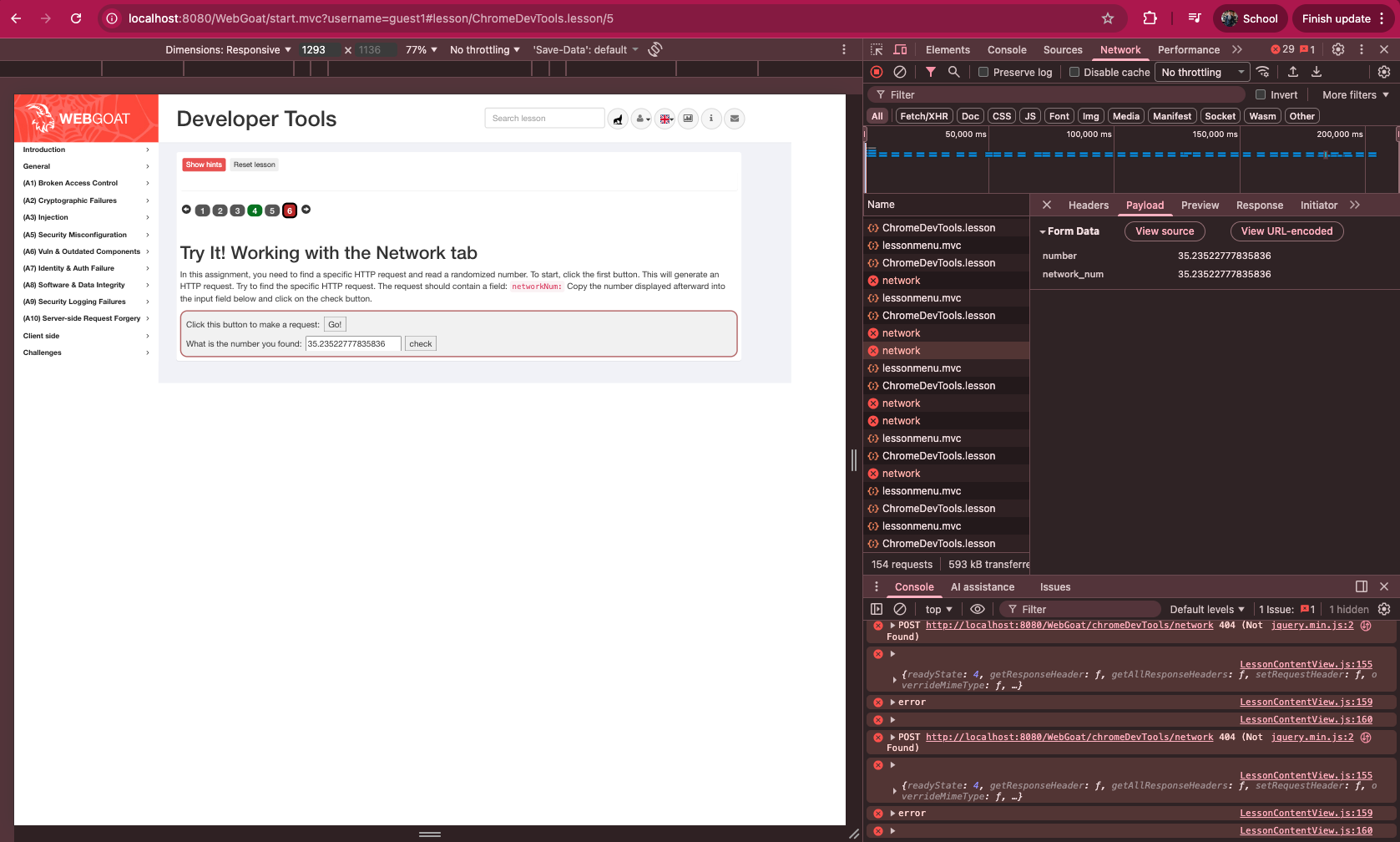
Imam grešku prilikom pokretanja OWASP ZAP-a. Nisam ga mogao instalirati pa ovaj dio nisam ni uradio.

## 3.3. Developer Tools

Ovdje saam naučio browser dev tools, inspect element i network tab.

  
Slika 1. Prikaz sekcije Developer Tools

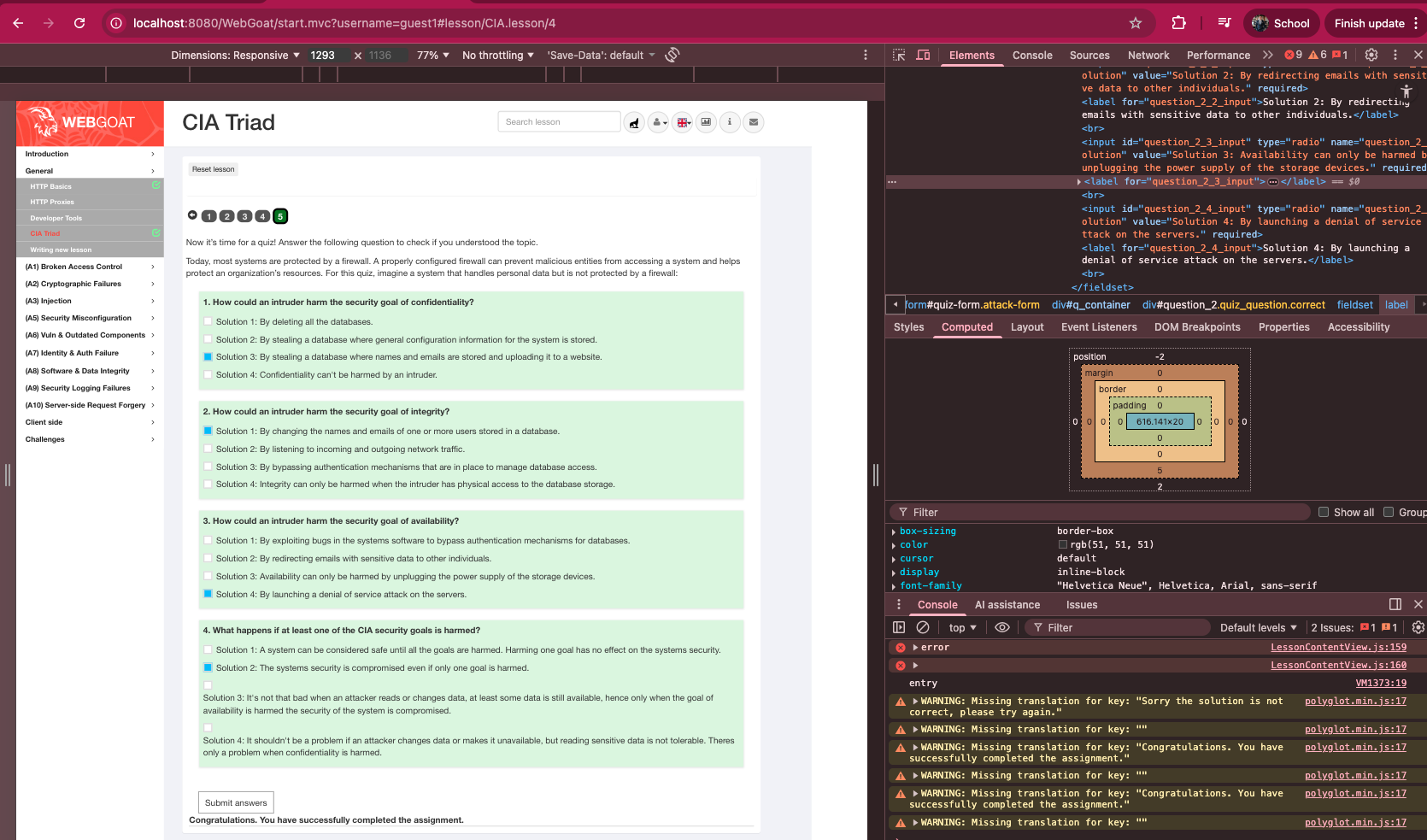
  
Slika 2. Prikaz urađenog 4-koraka

  
Slika 3. Prikaz greške u koraku 6

Pritisnuo sam button Go, napravio request i dobio sam neke greške u konzoli. Pronašao sam broj 35.23522777835836 međutim ništa se ne dešava kad pritisnem check button.

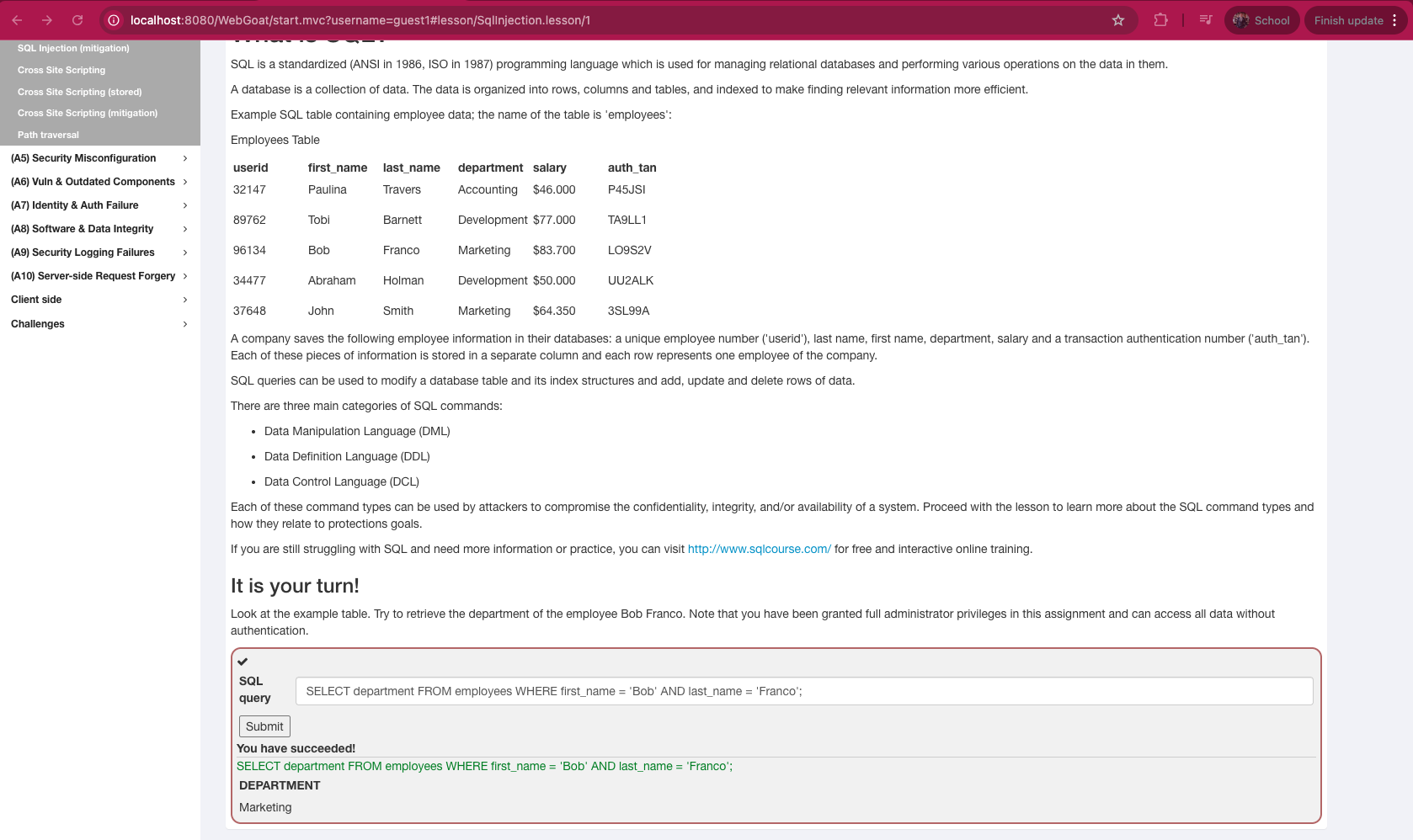
## 3.4. CIA Triad

Ovdje sam naučio Confidentialtiy, Integrity i Availability.

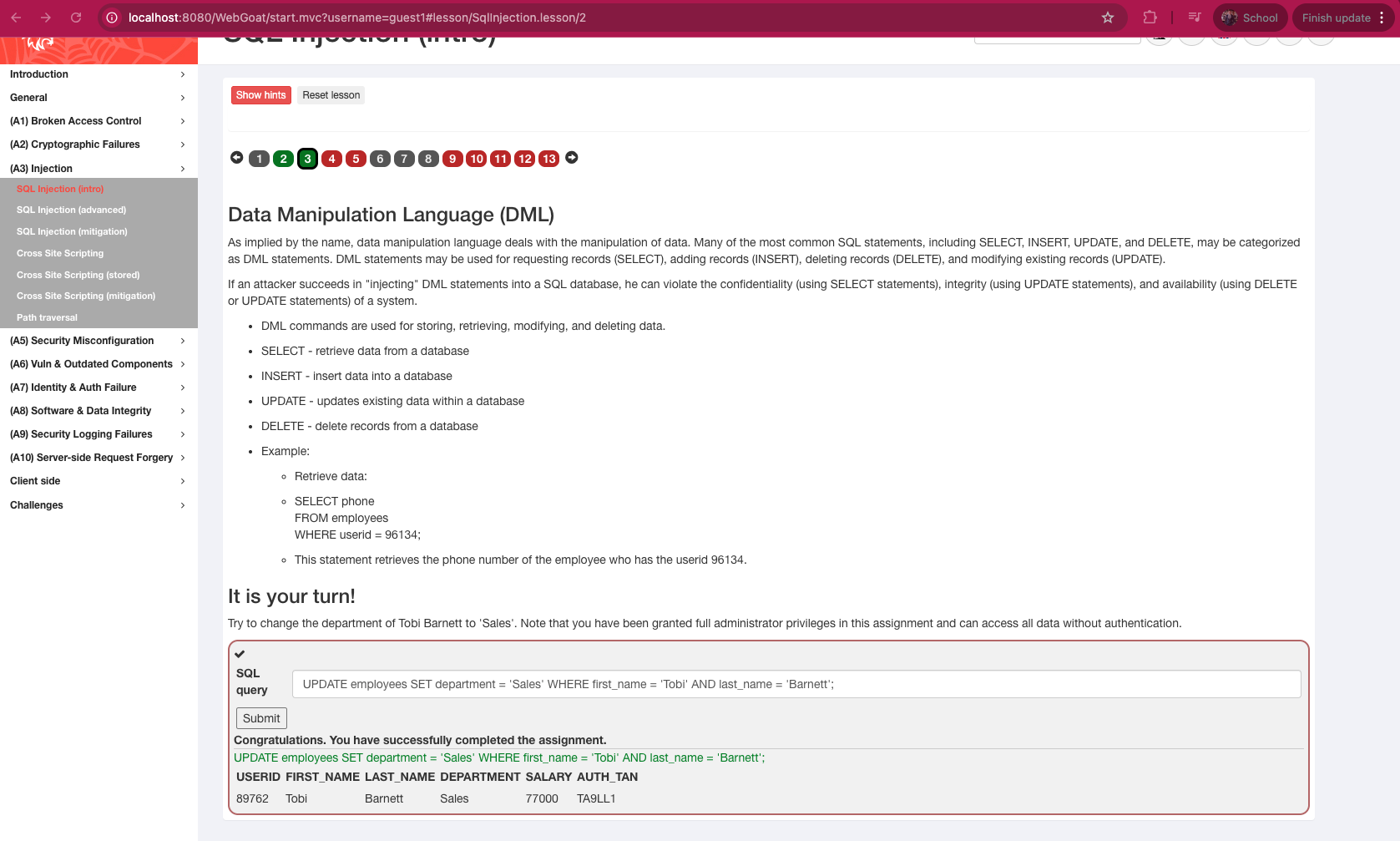
  
Slika 1. Prikaz urađene lekcije

## 3.5. SQL INJECTION(intro)

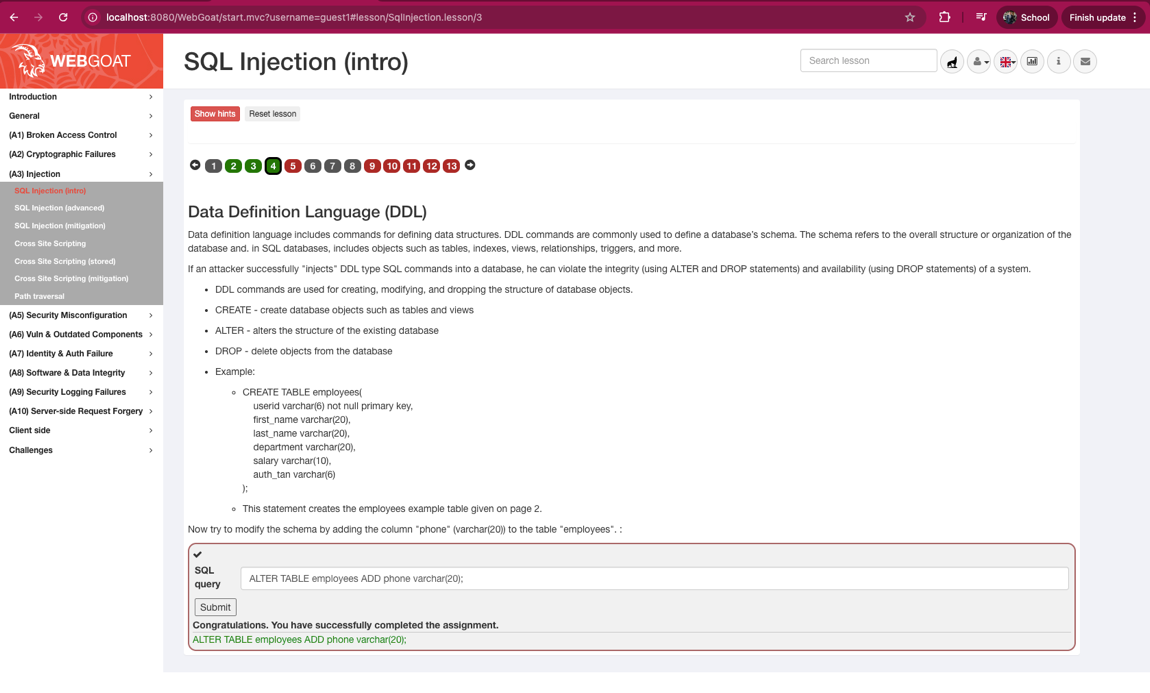
Ovdje sam naučio DML, DDL, DCL, Stringove SQL injection, Brojeve SQL injection.

  
Slika 1. Prikaz lekcije 2. What is SQL?

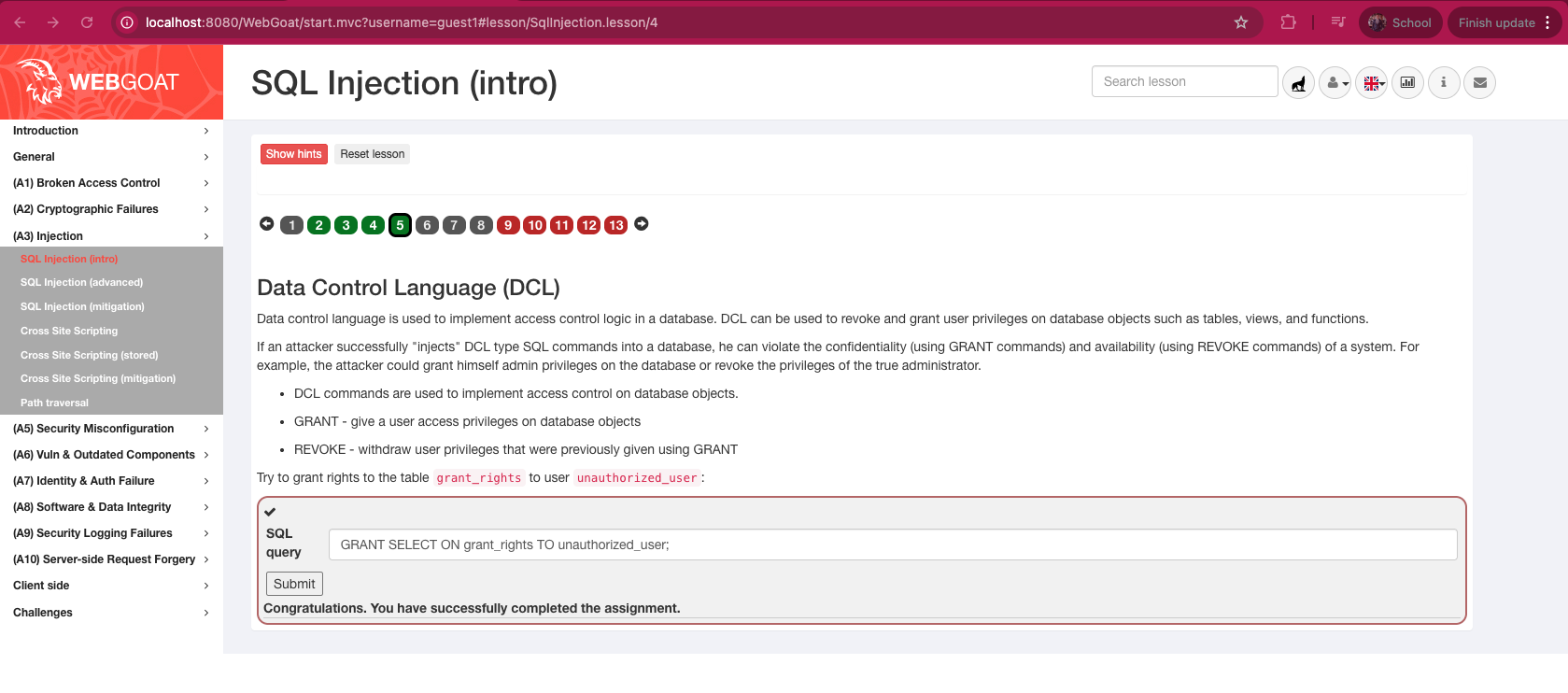
SQL query: SELECT department FROM employees WHERE first\_name = 'Bob' AND last\_name = 'Franco';

  
Slika 2. Prikaz lekcije 3. Dana Manipulation Language(DML)

SQL query: UPDATE employees SET department = 'Sales' WHERE first\_name = 'Tobi' AND last\_name = 'Barnett';

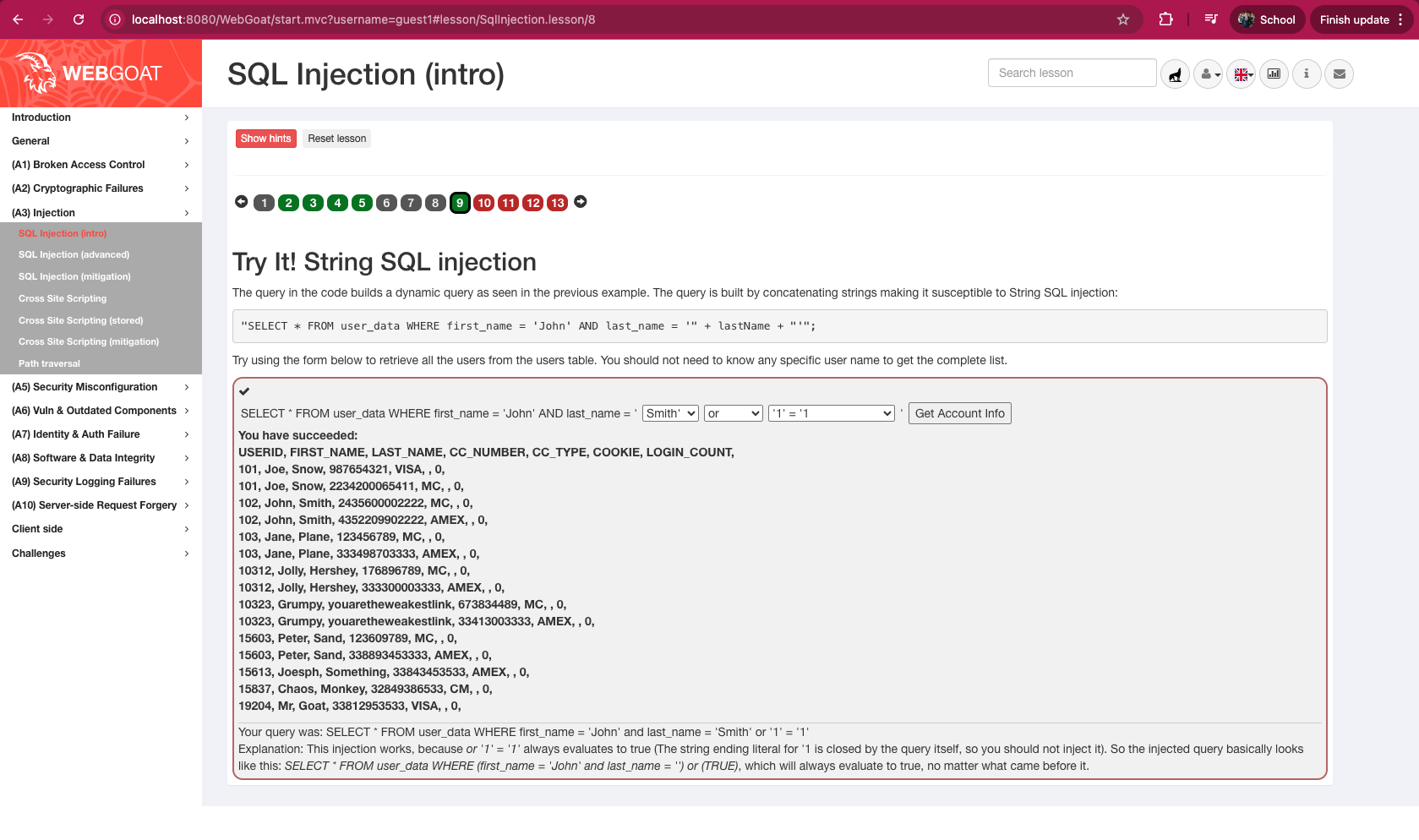
  
Slika 3. Prikaz lekcije 4. Dana Definition Language(DDL)

SQL query: ALTER TABLE employees ADD phone varchar(20);

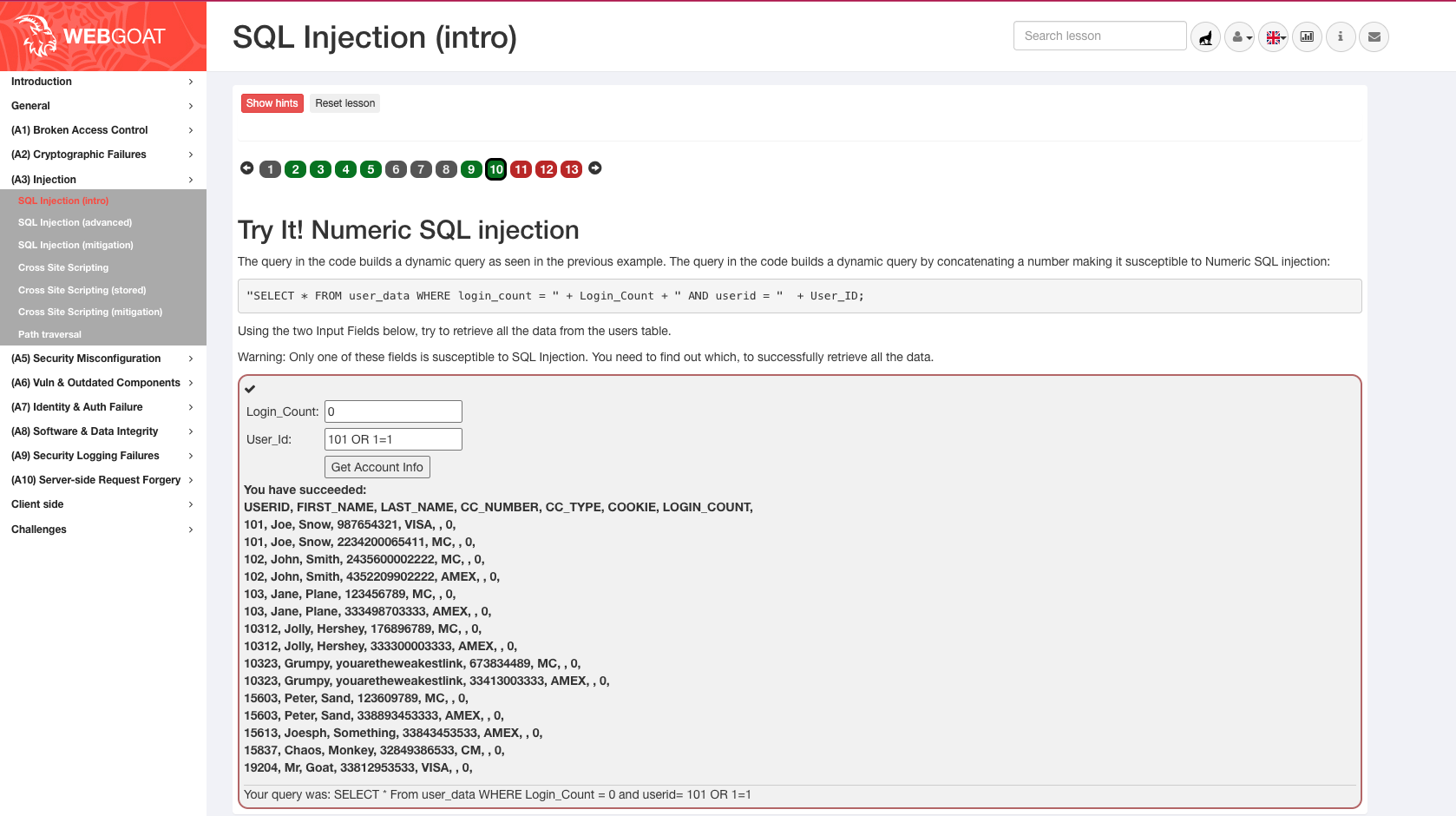


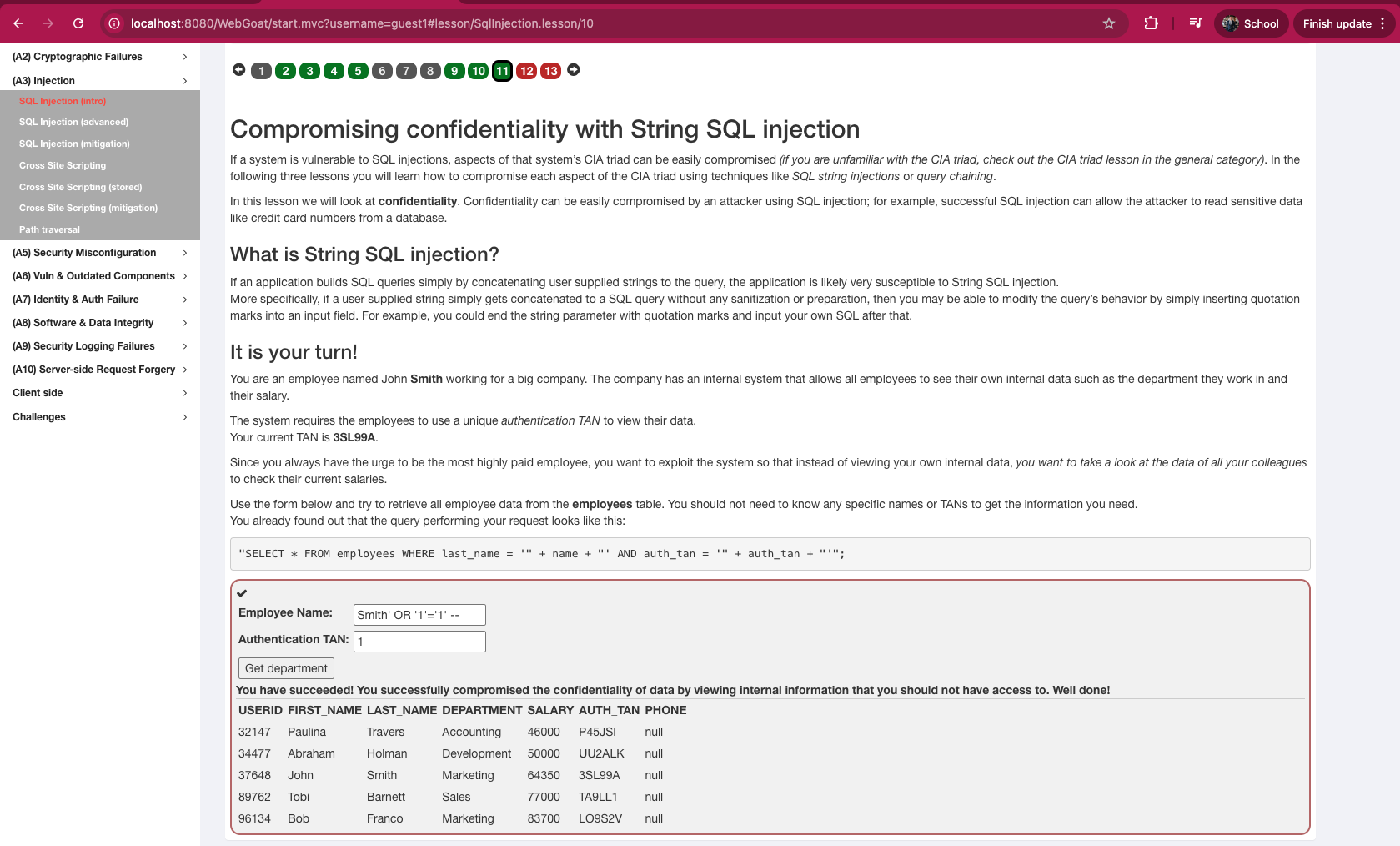
Slika 4. Prikaz lekcije 5. Dana Control Language(DCL)

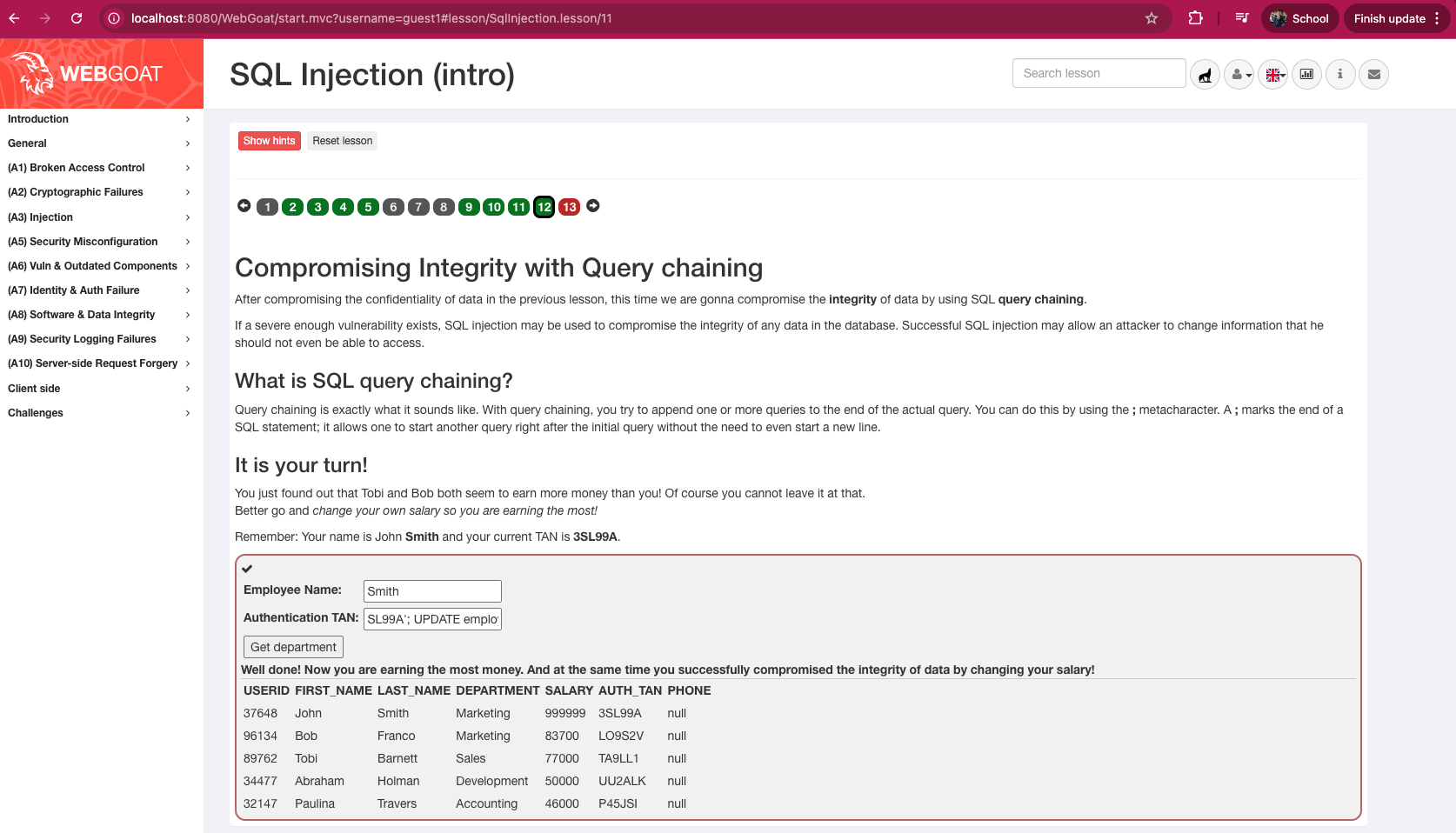
SQL query: GRANT SELECT ON grant\_rights TO unauthorized\_user;



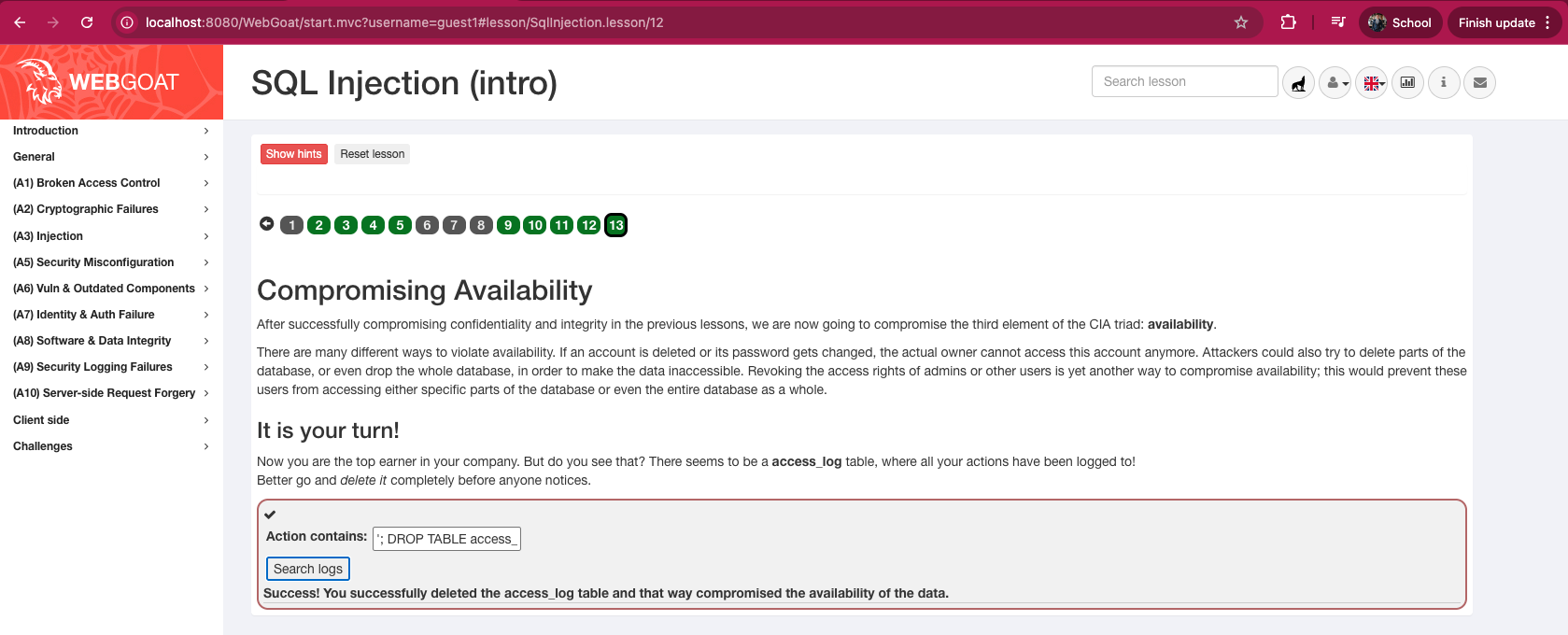
Slika 5. Prikaz lekcije 9. String SQL injection

  
Slika 6. Prikaz lekcije 10. Numeric SQL injection

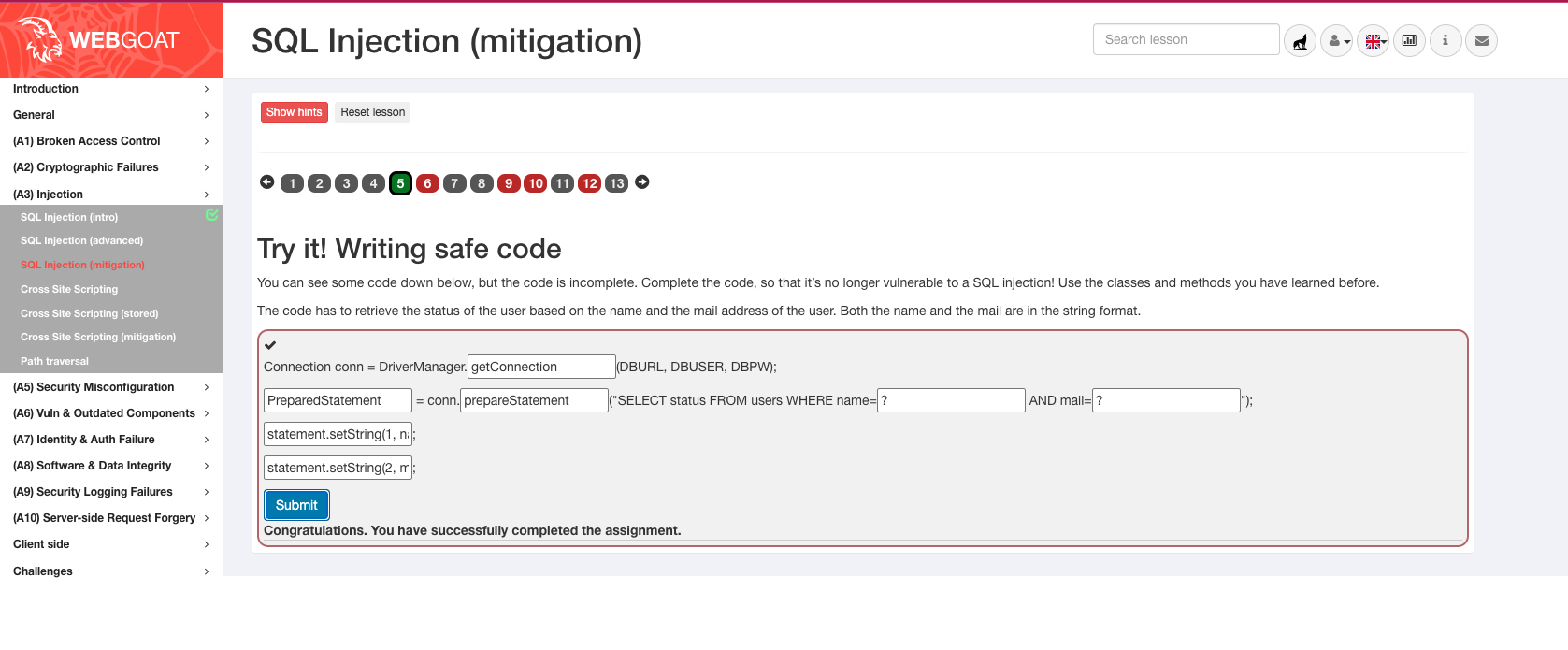
  
Slika 7. Prikaz lekcije 11. Compromising confidentiality with String SQL injection

  
Slika 8. Prikaz lekcije 12. Compromising Integrity with Query chaining

Authentication TAN: SL99A'; UPDATE employees SET salary = 999999 WHERE last\_name = 'Smith

  
Slika 9. Prikaz lekcije 13. Compromising Availability

## 3.6. SQL Injection (mitigation)

  
Slika 1. Prikaz urađene lekcije 5. Writing safe code

Connection conn = DriverManager.getCinnectuin(DBURL, DBUSER, DBPW);

 PreparedStatement= conn.prepareStatement("SELECT status FROM users WHERE name=? AND mail=?");

statement.setString(1, name);

statement.setString(2, mail);

Zaa lekciju 6(slika 2) korišten je sljedeći kod:

try {

Connection conn = DriverManager.getConnection(DBURL, DBUSER, DBPW);

String query = "SELECT \* FROM users WHERE name = ?";

PreparedStatement statement = conn.prepareStatement(query);

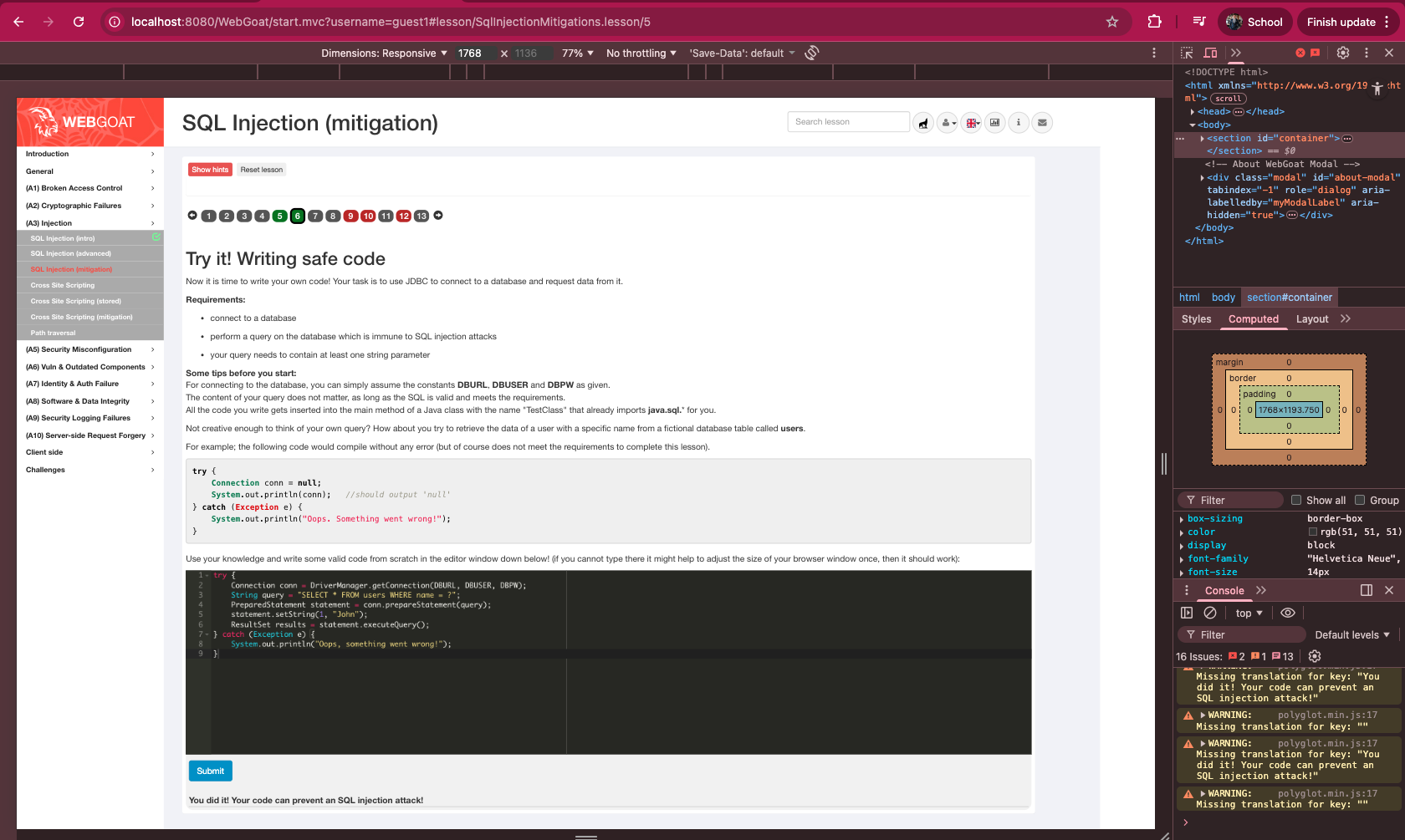
statement.setString(1, "John");

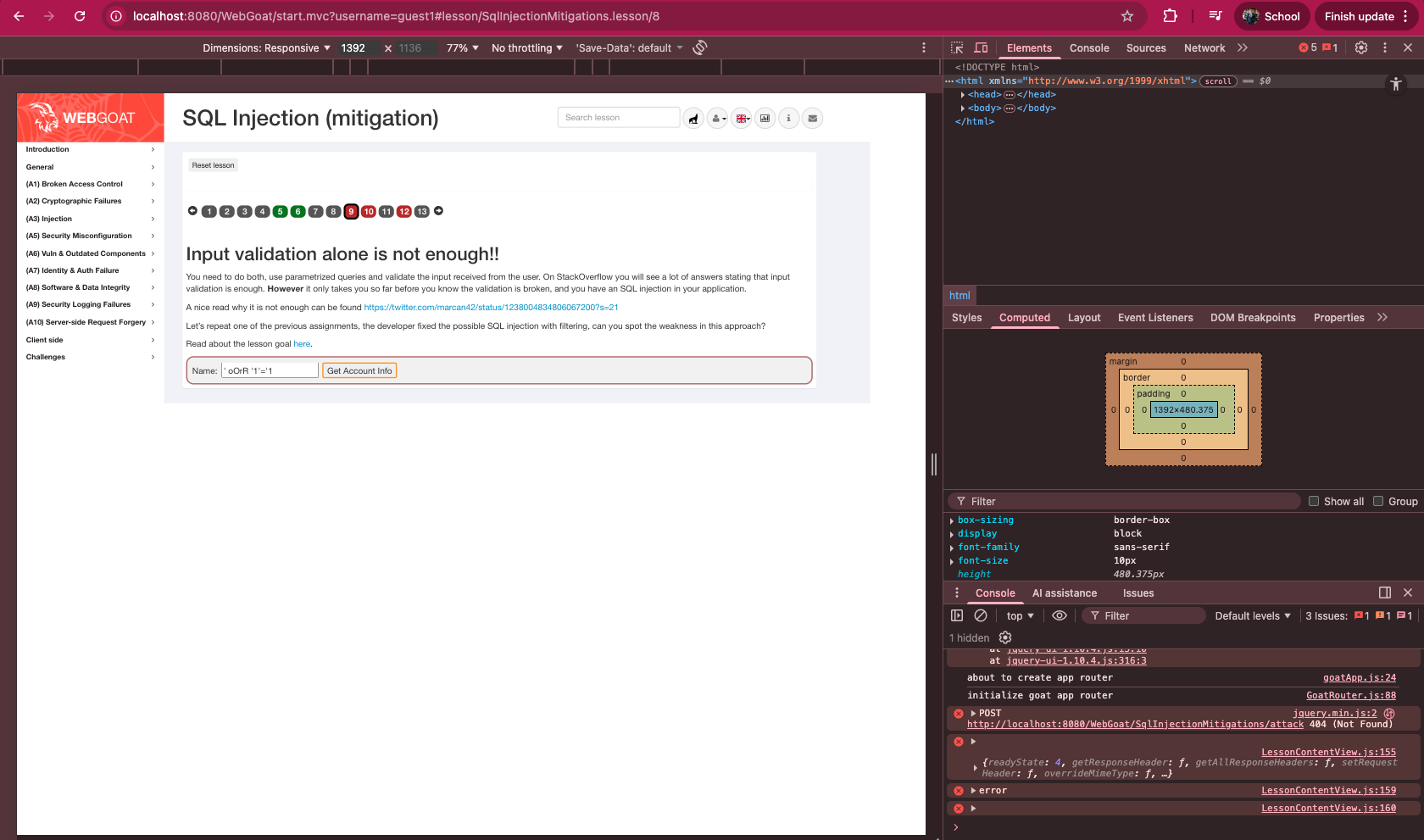
ResultSet results = statement.executeQuery();

} catch (Exception e) {

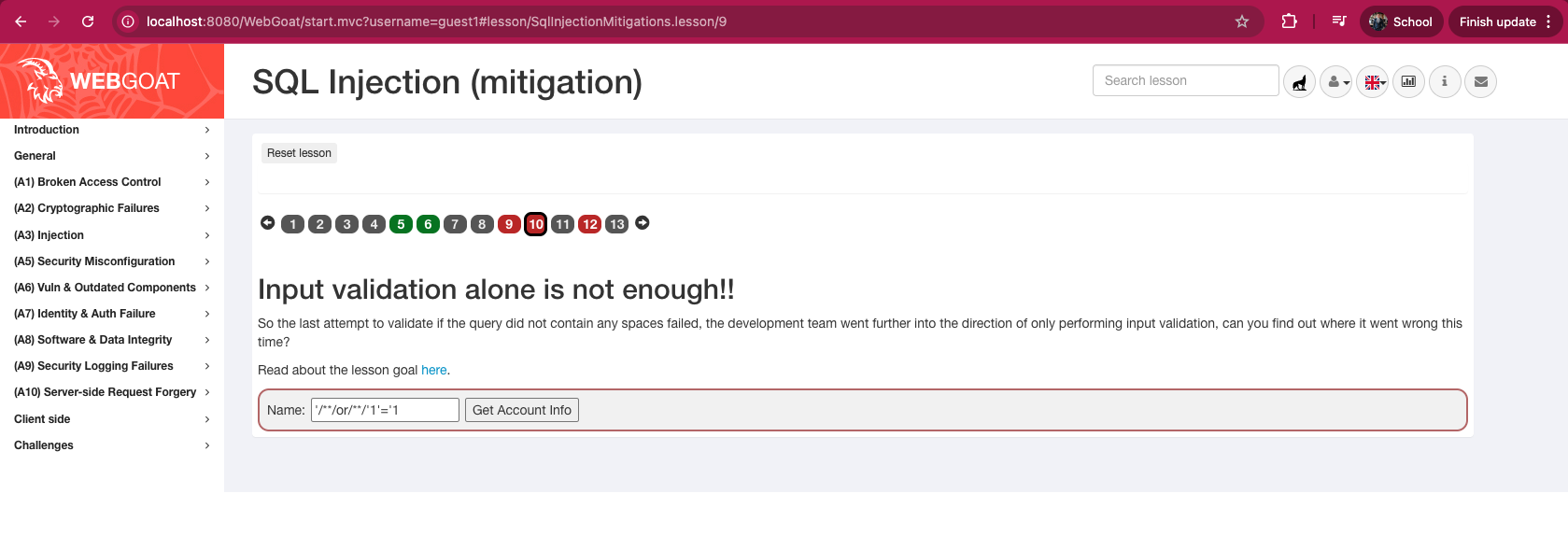
System.out.println("Oops, something went wrong!");

}

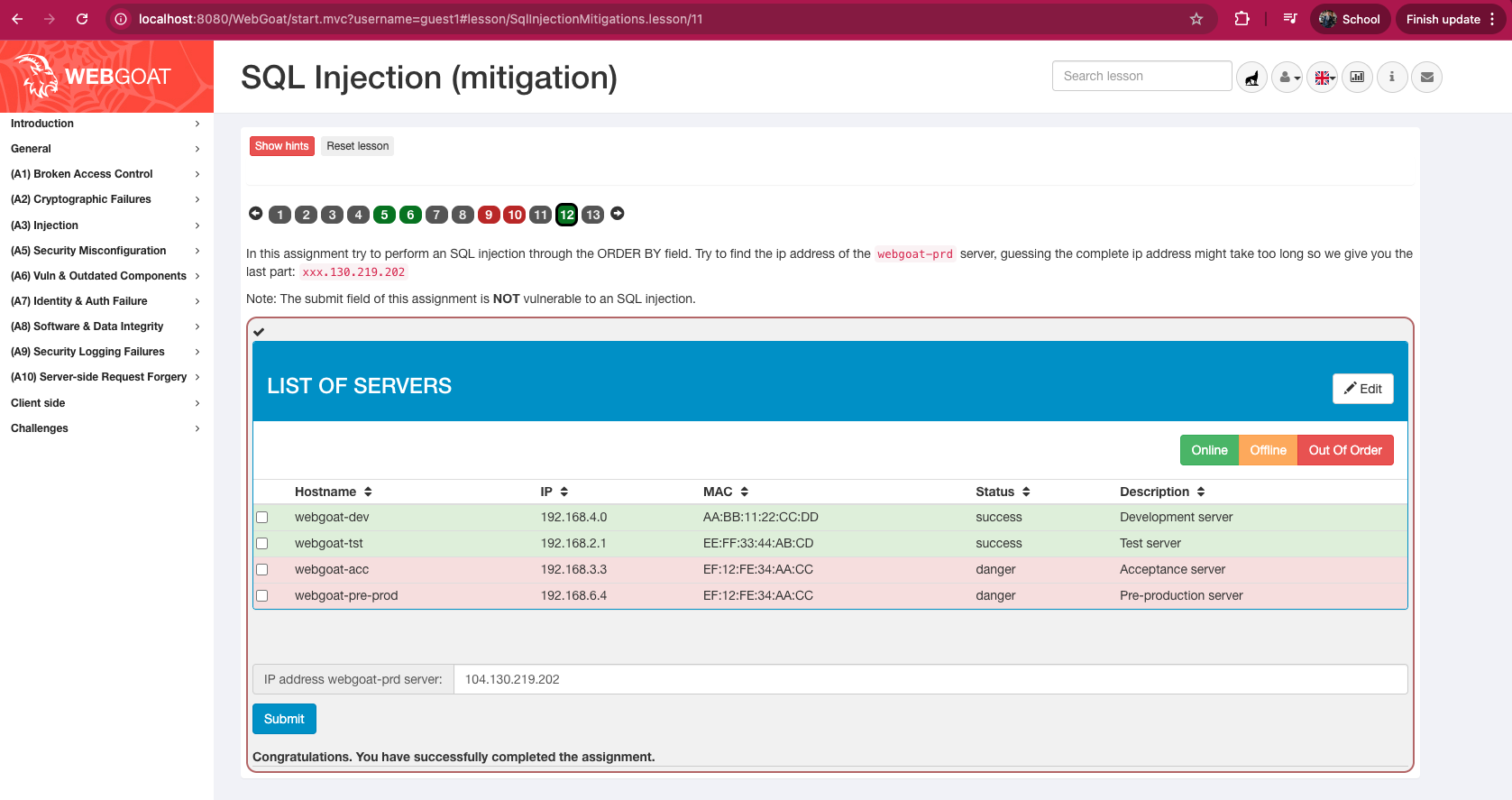
  
Slika 2. Prikaz urađene lekcije 6. Writing safe code

  
Slika 3. Prikaz greške u lekciji 9

Ova greška sugeriše da postoji problem sa komunikacijom između mog pregledača i WebGoat servera, a ne sa mojim SQL kodom.



Istu grešku imam i sa lekcijom 10.

  
Slika 4. Prikaz lekcije 12 – IP adresa je 104.130.219.202