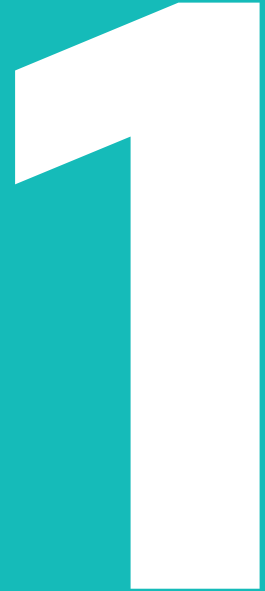


LIVE CODING LESSONS

Customer Experience

Phase one

Stand alone experience



Unknown customer

Customer hears about Live Coding Lessons

- Via tweet
- Via email
- Via developer microsite

Secure Coach | Guidelines | Missions | Research

SECURE CODE COACH

Stay ahead of secure coding best practices.

The cyber-threat landscape is constantly shifting and staying ahead of common software vulnerabilities is more important than ever. With a range of handy secure coding guidelines, hands-on missions, and videos we can help keep you up to date with best practices.

[Start Learning Now >](#)

Our latest secure coding guidelines.

[See all >](#)

- Using Components with Known Vulnerabilities
- Security Misconfiguration
- Server-Side Request Forgery

```
{
  "dependencies": {
    "foo": "1.0.0 - 2.999.9999",
    "bar": ">=1.0.2 < 1.1.2"
  }
}
```

Using Components with Known Vulnerabilities

Most applications make use of large amounts of third-party components. These components provide everything from logging, database access, and more. This makes developing and testing applications and saves a lot of time. But they're also made by people and some will inevitably contain vulnerabilities. Read the full article for more.

[View The Guidelines >](#)

View in browser

Announcing New Live Coding Lessons!

Lorem Ipsum insert introduction copy here, or read this arrested development text generator. Turn this skiff around! We have unlimited juice? This party is going to be off the hook.

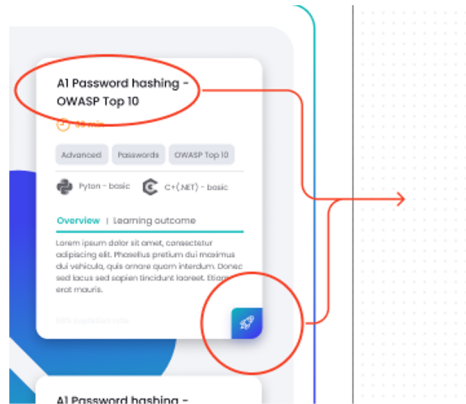
Live Coding Lessons

This is a long paragraph that communicates to the customer their training goal and leads to the call-to-action button below.

[Check it out!](#)

Landing Page

Customer arrives via link to landing page.



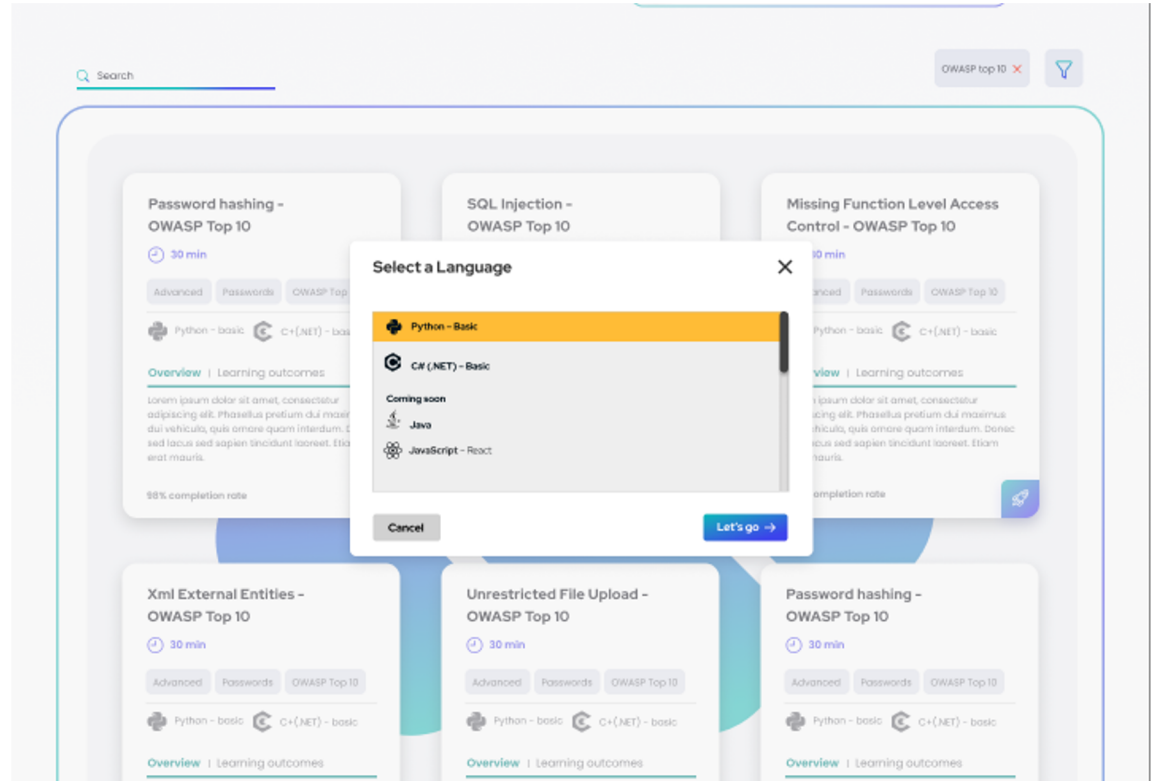
The screenshot shows a landing page for "Live Coding Lessons". At the top, there is a navigation bar with "Tournaments", "Training", "Courses", "Assessments", and "Live coding" (highlighted). The main heading is "TRY NOW Live Coding Lessons". Below this, a sub-heading reads "Learning reimagined. 100% hands-on exercises designed to enhance developers' secure coding skills via exercises in a live coding environment." To the right, there is a "Completion" widget showing a progress circle and a "Benefits" widget. Below the main text, there is a search bar and a filter for "OWASP top 10". The main content area features six cards, each representing a lesson:

- Password hashing - OWASP Top 10**: 30 min, 98% completion rate. Includes tabs for "Advanced", "Passwords", and "OWASP Top 10".
- SQL Injection - OWASP Top 10**: 30 min, 98% completion rate. Includes tabs for "Advanced", "Passwords", and "OWASP Top 10".
- Missing Function Level Access Control - OWASP Top 10**: 30 min, 98% completion rate. Includes tabs for "Advanced", "Passwords", and "OWASP Top 10".
- Xml External Entities - OWASP Top 10**: 30 min, 98% completion rate. Includes tabs for "Advanced", "Passwords", and "OWASP Top 10".
- Unrestricted File Upload - OWASP Top 10**: 30 min, 98% completion rate. Includes tabs for "Advanced", "Passwords", and "OWASP Top 10".
- Password hashing - OWASP Top 10**: 30 min, 98% completion rate. Includes tabs for "Advanced", "Passwords", and "OWASP Top 10".

Each card includes a "Python - basic" and "C+(.NET) - basic" icon, an "Overview | Learning outcomes" section with placeholder text, and a rocket icon at the bottom right.

Select a Language

Customer selects lesson,
and selects language.



Optional create an account

- Information we collect from free trial

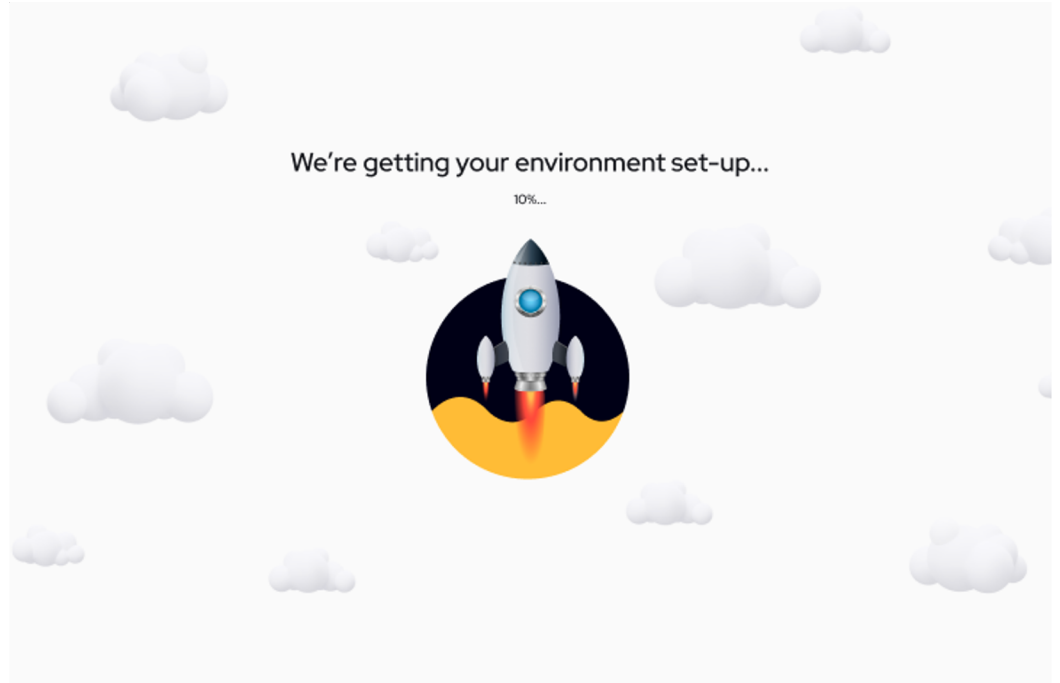
The image shows a 'Create an Account' modal form overlaid on a background of course cards. The modal has a title 'Create an Account' and a close button (X) in the top right corner. It contains the following fields and elements:

- Name:** A text input field.
- Email:** A text input field.
- Company:** A text input field.
- I am a...:** A dropdown menu with the option 'Select your role' visible.
- Terms and Privacy Policy:** A checkbox with the text 'Creating an account means you're okay with our [Terms of Service](#) and [Privacy Policy](#)'.
- Buttons:** A grey 'Cancel' button on the left and a blue 'Let's go →' button on the right.

The background shows several course cards, including 'SQL Injection - OWASP Top 10', 'Missing Function Level Access Control - OWASP Top 10', 'Unrestricted File Upload - OWASP Top 10', and 'Password hashing - OWASP Top 10'. Each card includes a duration (e.g., '30 min') and category tags like 'Advanced', 'Passwords', and 'OWASP Top 10'.

Animated Loading Page

Customer views loading
screen



Introduction

Customer reads
introduction message

The screenshot displays the Visual Studio Code interface during a lesson. On the left, a sidebar contains a 'PRODUCT TOUR' section with an 'INTRODUCTION' message and a 'STEPS' list. The main editor area is titled 'A3 Missing Function Level Access Control in C#' and shows the 'TransactionService.cs' file. The code defines a class 'TransactionService' with a constructor and a 'SearchTransactionAsync' method. The bottom status bar shows the terminal output for installing C# dependencies.

PRODUCT TOUR

INTRODUCTION

Now that you've found the secure solution, drop what you're doing and let's join forces. Follow the steps in the assignment to learn how to secure the `SearchTransactionAsync(System.String)` method from SQL injection as seen in the 'pick a solution' stage. Stick with Entity Framework and use LINQ's `Where` method to solve the assignment.

[Watch video to learn more](#)

STEPS

- 1 Investigate Transaction** [Reset lesson](#)
Investigate the `SearchTransactionAsync(System.String)` method.
[Visit Entity Framework to learn more](#)
[Get a hint](#)
- 2 Update transactions**
- 3 Return transactions**

[Continue](#) →

TransactionService.cs

```
1 using Microsoft.EntityFrameworkCore;
2 using SecureCodeWarriorBank.Data;
3
4 namespace SecureCodeWarriorBank;
5 public class TransactionService
6 {
7
8     2 references
9     private readonly SecureCodeWarriorBankContext _context;
10
11     0 references
12     public TransactionService(SecureCodeWarriorBankContext context)
13     {
14         _context = context;
15
16     0 references
17     public async Task<IEnumerable<Transaction>> SearchTransactionAsync(
18         string searchTerm)
19     {
20         var query = $"SELECT * FROM Transactions T WHERE T.owner = @searchTerm";
21         var transactionQuery =
22             _context.Transactions.FromSqlRaw(query);
23         return await transactionQuery.ToListAsync();
24     }
25 }
26
```

PROBLEMS 16 **OUTPUT** **DEBUG CONSOLE** **TERMINAL** C#

Installing C# dependencies...
Platform: linux, x86_64, name=debian, version=11

Finished

master 0 14 project Ln 1, Col 1 Spaces: 4 UTF-8 LF C#

View Hint

Customer is able to view a hint.

The screenshot displays the Visual Studio Code interface. On the left, a 'STEPS' panel is overlaid on top of the code editor. The first step, '1 Investigate Transaction', is active and contains the following text:

Investigate the `SearchTransactionAsync` (`System.String`) method.

[Visit Entity Framework to learn more](#)

[`AllowAnonymous`] takes priority over any other authorization attribute. So if you place it on controller level, all its endpoints will be accessible for non-authenticated users.

Below the text are three empty input fields, and a 'Continue →' button at the bottom.

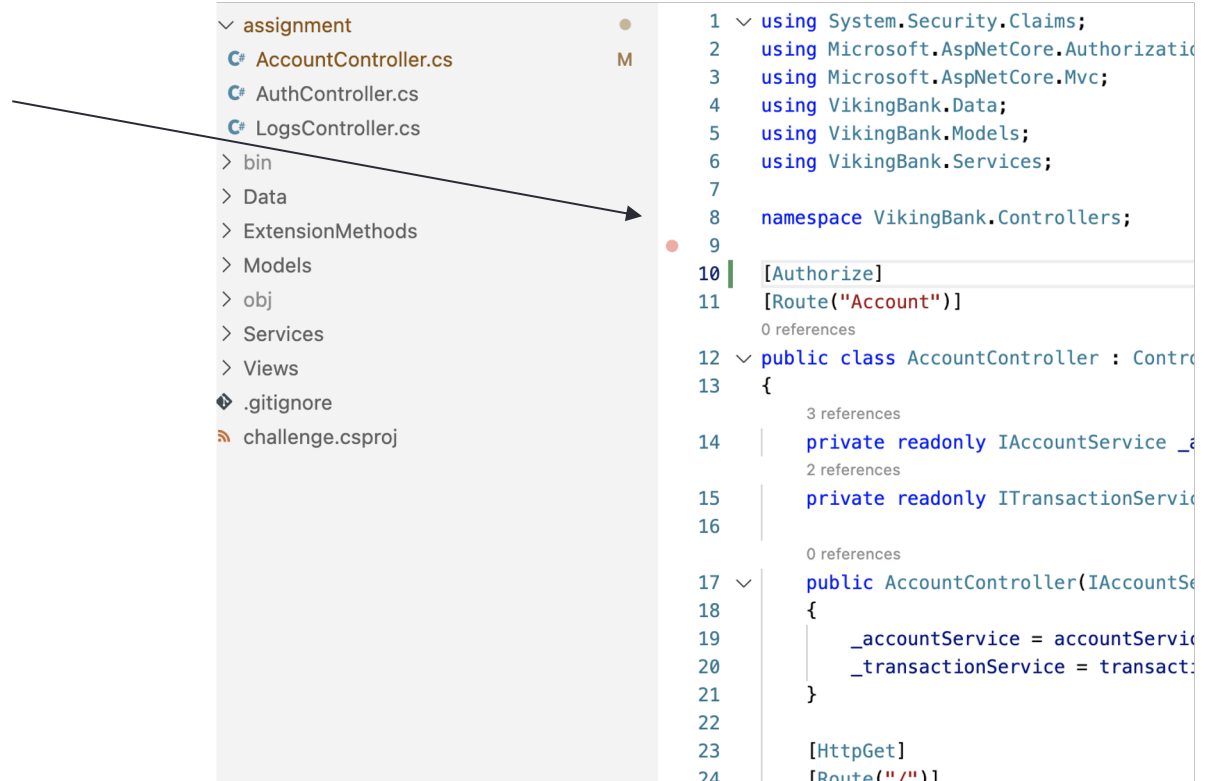
The background shows the Visual Studio Code editor with a C# file named `TransactionService.cs` open. The code includes the following:

```
1 using Microsoft.EntityFrameworkCore;
2 using SecureCodeWarriorBank.Data;
3
4 namespace SecureCodeWarriorBank;
5 public class TransactionService
6 {
7
8     private readonly SecureCodeWarriorBankContext _context;
9
10    public TransactionService(SecureCodeWarriorBankContext context)
11    {
12        _context = context;
13    }
14
15    public async Task<IEnumerable<Transaction>> SearchTransaction
16    {
17        var query = $"SELECT * FROM Transactions T WHERE T.owner
18
19        var transactionQuery =
20            _context.Transactions.FromSqlRaw(query);
21
22        return await transactionQuery.ToListAsync();
23    }
24 }
25
26
```

The bottom status bar shows the file path `14 project` and the current cursor position `Ln 1, Col 1`.

Enters Code

Customer enters code



```
1  using System.Security.Claims;
2  using Microsoft.AspNetCore.Authorization;
3  using Microsoft.AspNetCore.Mvc;
4  using VikingBank.Data;
5  using VikingBank.Models;
6  using VikingBank.Services;
7
8  namespace VikingBank.Controllers;
9
10 [Authorize]
11 [Route("Account")]
12 0 references
13 public class AccountController : Controller
14 {
15     3 references
16     private readonly IAccountService _accountService;
17     2 references
18     private readonly ITransactionService _transactionService;
19     0 references
20     public AccountController(IAccountService accountService, ITransactionService transactionService)
21     {
22     }
23     [HttpGet]
24     [Route("/")]
```

Reviewing Code

Reviewing Code button
state change

The screenshot displays the Visual Studio Code interface during a code review session. On the left, the 'PRODUCT TOUR' sidebar is open, showing the 'INTRODUCTION' and 'STEPS' sections. The 'STEPS' section is active, with the first step 'Investigate Transaction' selected. The main editor area shows the 'EXPLORER' view with the 'TransactionService.cs' file selected. The code in the editor is as follows:

```
1 using Microsoft.EntityFrameworkCore;
2 using SecureCodeWarriorBan;
3
4 namespace SecureCodeWarriorBan
5 {
6     public class TransactionService
7     {
8         private readonly SecureCodeWarriorBan _context;
9
10        public TransactionService(SecureCodeWarriorBan context)
11        {
12            _context = context;
13        }
14
15        public async Task<IEnumerable<Transaction>> GetTransactions()
16        {
17            var query = $"SELECT * FROM Transactions";
18
19            var transactionQuery = _context.Transactions
20                .Where(t => t.Query == query);
21
22            return await transactionQuery.ToListAsync();
23        }
24    }
25 }
26
```

At the bottom of the interface, there is a 'Reviewing code' button with a '00:10' timer. The 'PROBLEMS' view at the bottom right shows 16 problems, with the first one being 'Installing C# dependencies... Platform: linux, x86_64, nameid'. The 'OUTPUT' and 'DEBUG CONSOLE' views are also visible.

Step Complete

Customer completes a step, and moves to second step.

The screenshot shows the Visual Studio Code interface with a tutorial for 'A3 Missing Function Level Access Control in C#'. The interface is divided into several sections:

- PRODUCT TOUR:** Includes an 'INTRODUCTION' section with text: 'Now that you've found the secure solution, drop what you're doing and let's join forces. Follow the steps in the assignment to learn how to secure the `SearchTransactionAsync(System.String)` method from SQL Injection as seen in the 'pick a solution' stage. Stick with Entity Framework and use LINQ's `Where` method to solve the assignment.' Below this is a 'Watch video to learn more' button.
- STEPS:** A list of three steps:
 1. Investigate Transaction (Completed, indicated by a blue checkmark)
 2. Update transactions (Active, indicated by a blue circle with '2'). Below this step, it says: 'Investigate the `SearchTransactionAsync(System.String)` method. Visit Entity Framework to learn more. [AllowAnonymous] takes priority over any other authorization attribute. So if you place it on controller level, all its endpoints will be accessible for non-authenticated users.'
 3. Return transactions

At the bottom of the steps section is a 'Continue' button with a right-pointing arrow.

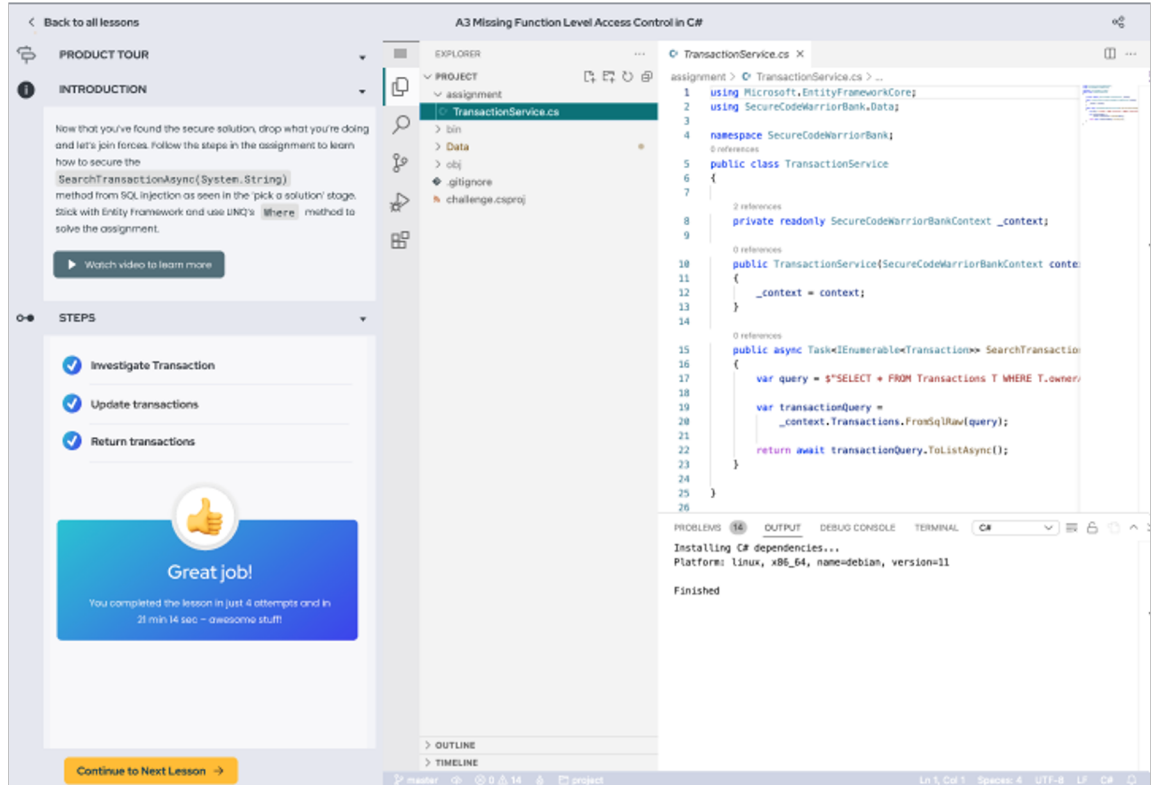
The right side of the IDE shows the Explorer view with a project structure including 'assignment', 'TransactionService.cs', 'bin', 'Data', 'obj', '.gignore', and 'challenge.csproj'. The main editor displays the code for 'TransactionService.cs':

```
1 using Microsoft.EntityFrameworkCore;
2 using SecureCodeWarriorBank.Data;
3
4 namespace SecureCodeWarriorBank;
5
6 public class TransactionService
7 {
8     private readonly SecureCodeWarriorBankContext _context;
9
10    public TransactionService(SecureCodeWarriorBankContext conte
11    {
12        _context = context;
13    }
14
15    public async Task<IEnumerable<Transaction>> SearchTransaction
16    {
17        var query = $"SELECT * FROM Transactions T WHERE T.owner
18
19        var transactionQuery =
20            _context.Transactions.FromSqlRaw(query);
21
22        return await transactionQuery.ToListAsync();
23    }
24
25
26 }
```

At the bottom right, the Output window shows the message: 'Installing C# dependencies... Platform: linux, x86_64, name=debian, version=11 Finished'.

Lesson completed

Customer submits the correct answer, and has the option to move on immediately to the next language.



The screenshot shows a coding IDE with a lesson completion overlay on the left and a code editor on the right.

Lesson Completion Overlay:

- Back to all lessons
- PRODUCT TOUR
- INTRODUCTION: Now that you've found the secure solution, drop what you're doing and let's join forces. Follow this steps in the assignment to learn how to secure the `SearchTransactionAsync(System.String)` method from SQL injection as seen in the 'pick a solution' stage. Stick with Entity framework and use UNQ's `Where` method to solve the assignment. [Watch video to learn more](#)
- STEPS:
 - Investigate Transaction
 - Update transactions
 - Return transactions
- Great job!**
You completed the lesson in just 4 attempts and in 21 min 14 sec - awesome stuff!
- [Continue to Next Lesson](#)

Code Editor (TransactionService.cs):

```
1 using Microsoft.EntityFrameworkCore;
2 using SecureCodeWarriorBank.Data;
3
4 namespace SecureCodeWarriorBank;
5
6 public class TransactionService
7 {
8     private readonly SecureCodeWarriorBankContext _context;
9
10    public TransactionService(SecureCodeWarriorBankContext context)
11    {
12        _context = context;
13    }
14
15    public async Task<IEnumerable<Transaction>> SearchTransaction
16    (
17        var query = $"SELECT * FROM Transactions T WHERE T.owner"
18    )
19    {
20        var transactionQuery =
21            _context.Transactions.FromSqlRaw(query);
22        return await transactionQuery.ToListAsync();
23    }
24 }
25
26
```

Terminal Output:

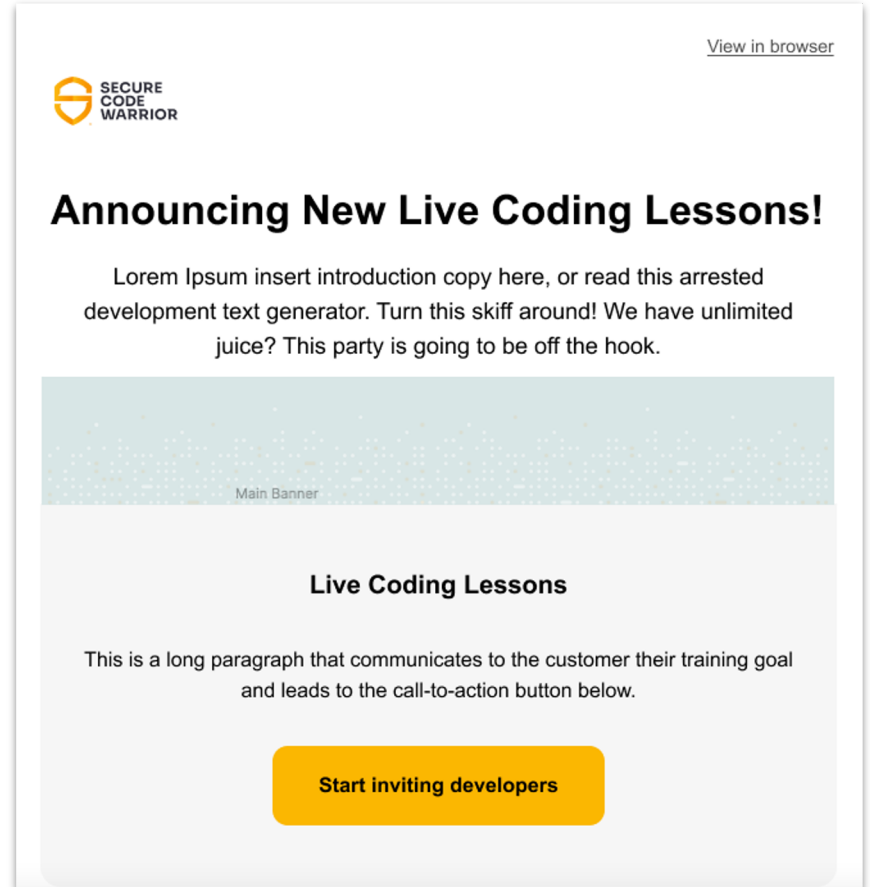
```
Installing C# dependencies...
Platform: linux, x86_64, name=debian, version=11
Finished
```

Phase Two
Existing Customer



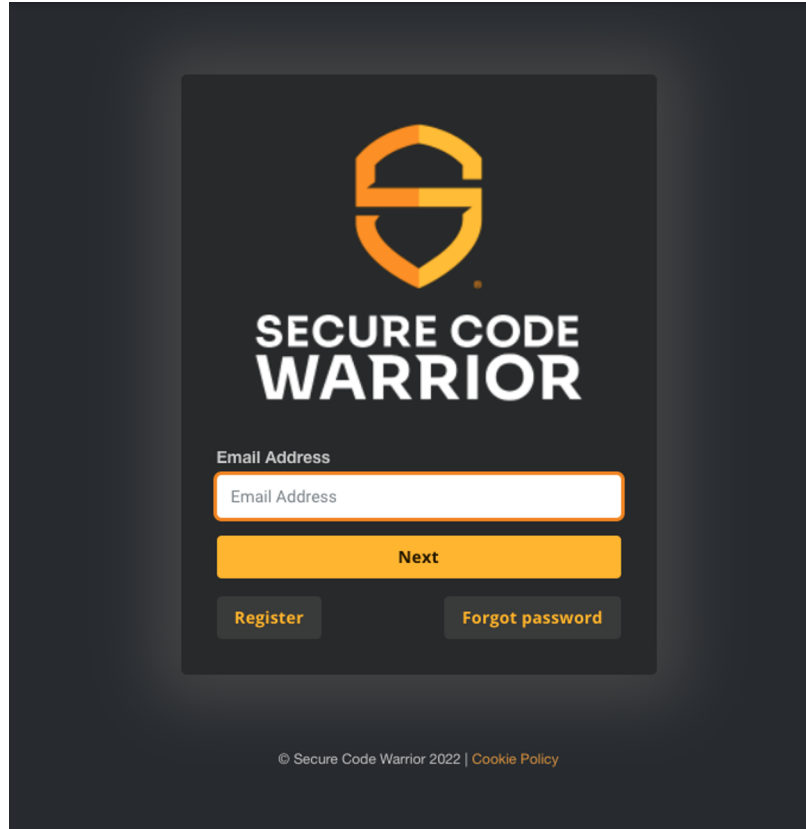
Customer invite via email

Customer receives an invite via email



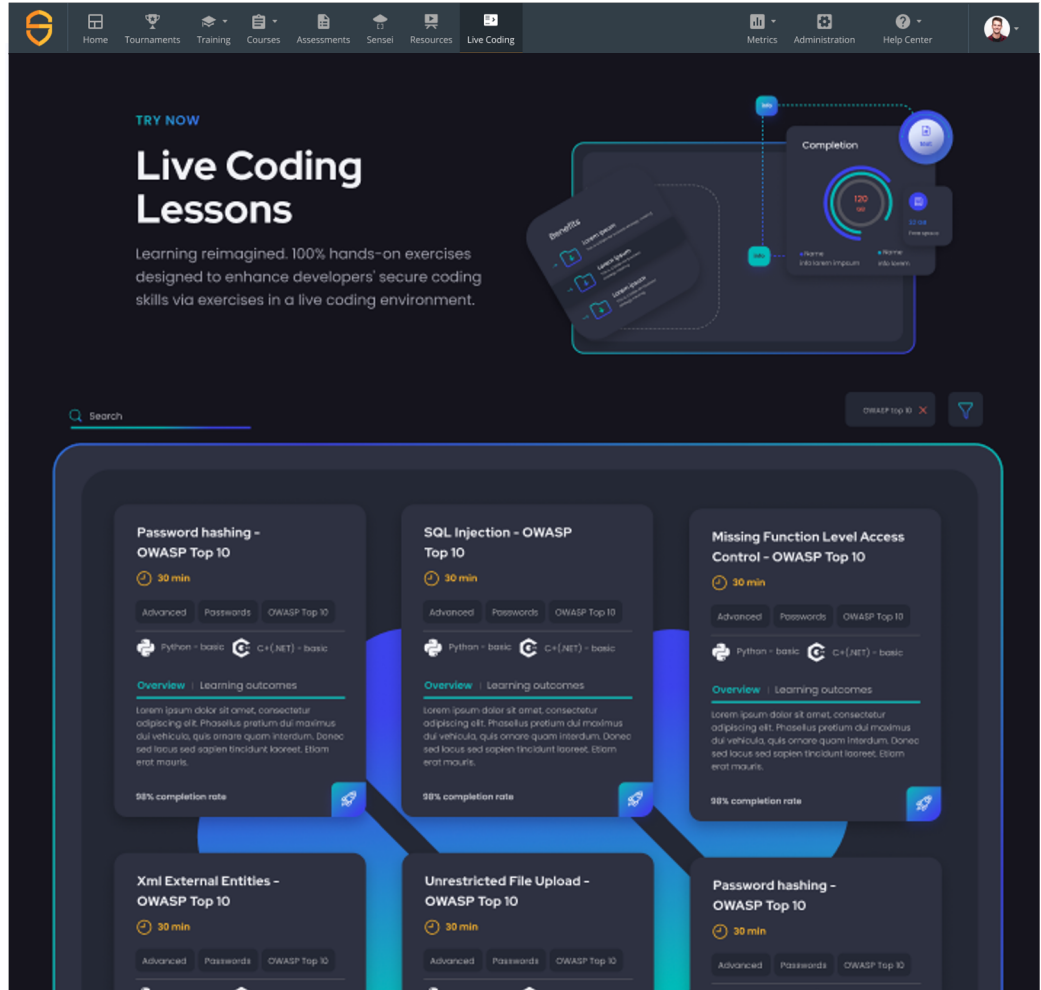
Log In

Customer is directed to a login screen



Landing Page

Customer is logged in and arrives at landing page



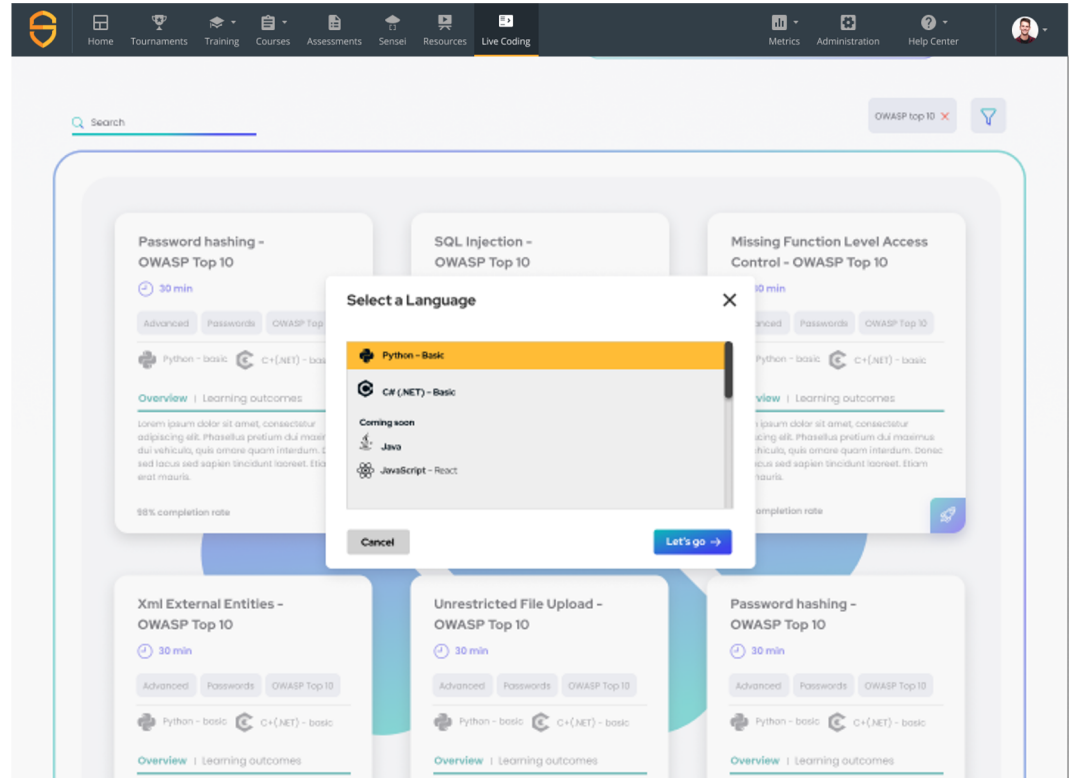
Landing Page - Dark theme

Customer is logged in and arrives at landing page



Select a Language

Customer selects a language, and goes through the same phase as phase one, with the exception of logged in state.



Repeats LCL

Customer repeats challenge, and moves on to next lesson, or can go back to landing paged and browse the lessons.

The screenshot displays a learning management system interface. On the left, a sidebar shows a 'PRODUCT TOUR' section with an 'INTRODUCTION' and 'STEPS' section. The 'STEPS' section lists three tasks: 'Investigate Transaction', 'Update transactions', and 'Return transactions', all of which are marked as completed with blue checkmarks. Below the steps is a 'Great job!' notification with a thumbs-up icon, stating 'You completed the lesson in just 4 attempts and in 21 min 14 sec - awesome stuff!'. A 'Continue to Next Lesson' button is visible at the bottom of the sidebar.

The main content area is titled 'A3 Missing Function Level Access Control in C#' and shows a code editor for 'TransactionService.cs'. The code is as follows:

```
1 using Microsoft.EntityFrameworkCore;
2 using SecureCodeWarriorBank.Data;
3
4 namespace SecureCodeWarriorBank;
5 public class TransactionService
6 {
7
8     private readonly SecureCodeWarriorBankContext _context;
9
10    public TransactionService(SecureCodeWarriorBankContext context)
11    {
12        _context = context;
13    }
14
15    public async Task<IEnumerable<Transaction>> SearchTransactions(
16    {
17        var query = $"SELECT * FROM Transactions T WHERE T.owner
18
19        var transactionQuery =
20            _context.Transactions.FromSqlRaw(query);
21
22        return await transactionQuery.ToListAsync();
23    }
24
25
26
```

At the bottom of the code editor, there is a 'PROBLEMS' section with 14 items, an 'OUTPUT' section, a 'DEBUG CONSOLE' section, and a 'TERMINAL' section. The 'TERMINAL' section shows the output: 'Installing C# dependencies...' and 'Platform: Linux, x86_64, name=debian, version=11'. The 'Finished' status is also visible.

Phase two

Deep Course Integration



Customer arrives via course

Customer selects a live coding lesson from course



OWASP 1-5
Introduction to OWASP Top 10 Awareness (with latest updates from the Web top 10 2021)

Hide Activities

- 1 **Video: Missing Function Level Access Control**
Not Watched
- 2 **Walkthrough Mission: Missing Function Level Access Control**
Not Completed
- 3 **Live Coding Lesson: A3 Missing Function Level Access Control** NEW
Not Viewed Yet
- 4 **Video: Plain Text Storage of Passwords**
Not Watched
- 5 **Walkthrough Mission: Plain Text Storage of Passwords**
Not Completed
- 6 **Challenge: Plain Text Storage of Passwords**
0/2 stages completed
- 7 **Video: SQL Injection**
Not Watched
- 8 **Walkthrough Mission: SQL Injection**

NEW IN: Live Coding Lessons are now available in Courses for you to try out!

Note that points earned through a Live Coding Lesson will not be part of your Course score and only time spent doing the lesson will be saved.

Repeat Live Coding Lessons

Customer repeats challenge, and moves on to next lesson, or can go back to landing page and browse the lessons.

The screenshot displays a web-based coding environment. On the left, a sidebar contains a 'PRODUCT TOUR' section with an 'INTRODUCTION' card. The introduction text reads: 'Now that you've found the secure solution, drop what you're doing and let's join forces. Follow the steps in the assignment to learn how to secure the SearchTransactionAsync(System, String) method from SQL injection as seen in the 'pick a solution' stage. Stick with Entity Framework and use LINQ's Where method to solve the assignment.' Below this is a 'Watch video to learn more' button. The 'STEPS' section below shows three completed tasks: 'Investigate Transaction', 'Update transactions', and 'Return transactions', each with a blue checkmark. A large blue and white notification box with a thumbs-up icon says 'Great job! You completed the lesson in just 4 attempts and in 21 min 14 sec - awesome stuff!'. At the bottom of the sidebar is a 'Continue to Next Lesson' button.

The main area is a code editor titled 'A3 Missing Function Level Access Control in C#'. The Explorer pane on the left shows a project structure with folders for 'bin', 'Data', 'obj', and 'challenge.csproj', and a file named 'TransactionService.cs'. The code editor shows the following C# code:

```
1 using Microsoft.EntityFrameworkCore;
2 using SecureCodeWarriorBank.Data;
3
4 namespace SecureCodeWarriorBank;
5 public class TransactionService
6 {
7
8     [references]
9     private readonly SecureCodeWarriorBankContext _context;
10
11     [references]
12     public TransactionService(SecureCodeWarriorBankContext context)
13     {
14         _context = context;
15     }
16
17     [references]
18     public async Task<IEnumerable<Transaction>> SearchTransaction
19     {
20         var query = $"SELECT * FROM Transactions T WHERE T.ownerId =
21         _context.Transactions.FromSqlRaw(query);
22         var transactionQuery =
23         _context.Transactions.FromSqlRaw(query);
24         return await transactionQuery.ToListAsync();
25     }
26 }
```

At the bottom of the editor, the 'PROBLEMS' pane shows 16 errors, with the first one being 'Installing C# dependencies... Platform: linux, x86_64, name=debian, version=11'. The 'OUTPUT' pane shows 'Finished'.

Survey Customer

The screenshot displays a live coding environment with a dark-themed header containing navigation icons and labels: Home, Tournaments, Training, Courses, Assessments, Sensel, Resources, and Live Coding. On the right side of the header, there are icons for Metrics, Administration, Help Center, and a user profile.

The main content area is titled "A3 Missing Function Level Access Control in C#" and features a sidebar on the left with sections for "PRODUCT TOUR", "INTRODUCTION", and "STEPS". The "STEPS" section includes three completed items: "Investigate Transaction", "Update transactions", and "Return transactions". A large blue notification box with a thumbs-up icon says "Great job! You completed the lesson in just 4 attempts and in 21 min 14 sec - awesome stuff!".

The central focus is a "Congratulations!" modal window. It features a circular icon with a beaker and a lightbulb. The text inside the modal reads: "Congratulations! You've just completed your first live coding lesson. You needed 5 hints. This took 25 minutes to complete. How was it?" Below the text is a progress bar and three buttons: "Save result", "Live coding", and a refresh icon.

The background shows a code editor with a file explorer on the left displaying a project structure with folders like "bin", "Data", "obj", ".gitignore", and "challenge.csproj". The code editor itself shows C# code for a class named "TransactionService" within a namespace "SecureCodeWarriorBank". The code includes using statements for "Microsoft.EntityFrameworkCore" and "SecureCodeWarriorBank.Data", and a partial class definition for "TransactionService".

At the bottom of the interface, there is a terminal window showing the command "dotnet run" and the output "Installing C# dependencies... Platform: Linux, x86_64, name".

Thankyou