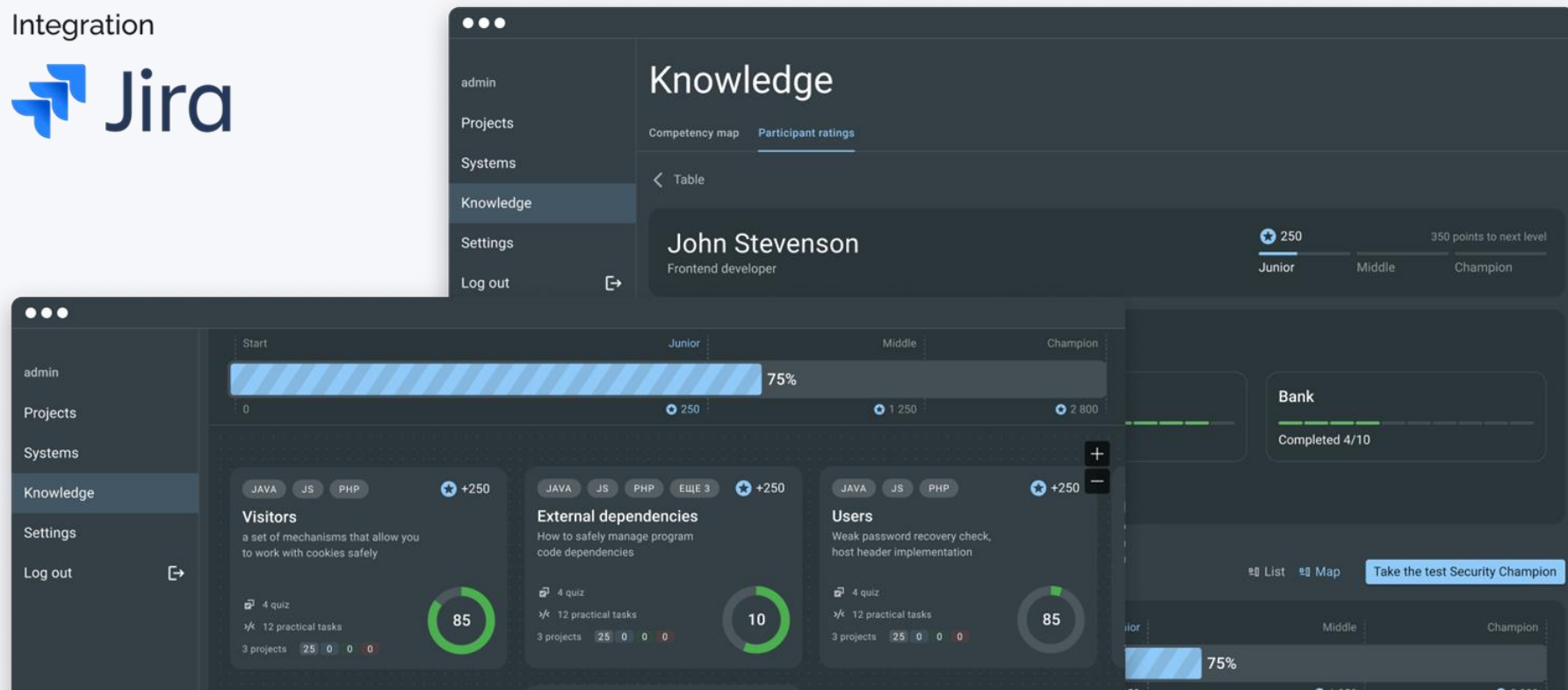


**A platform to teach teams
how to securely develop
and operate products**

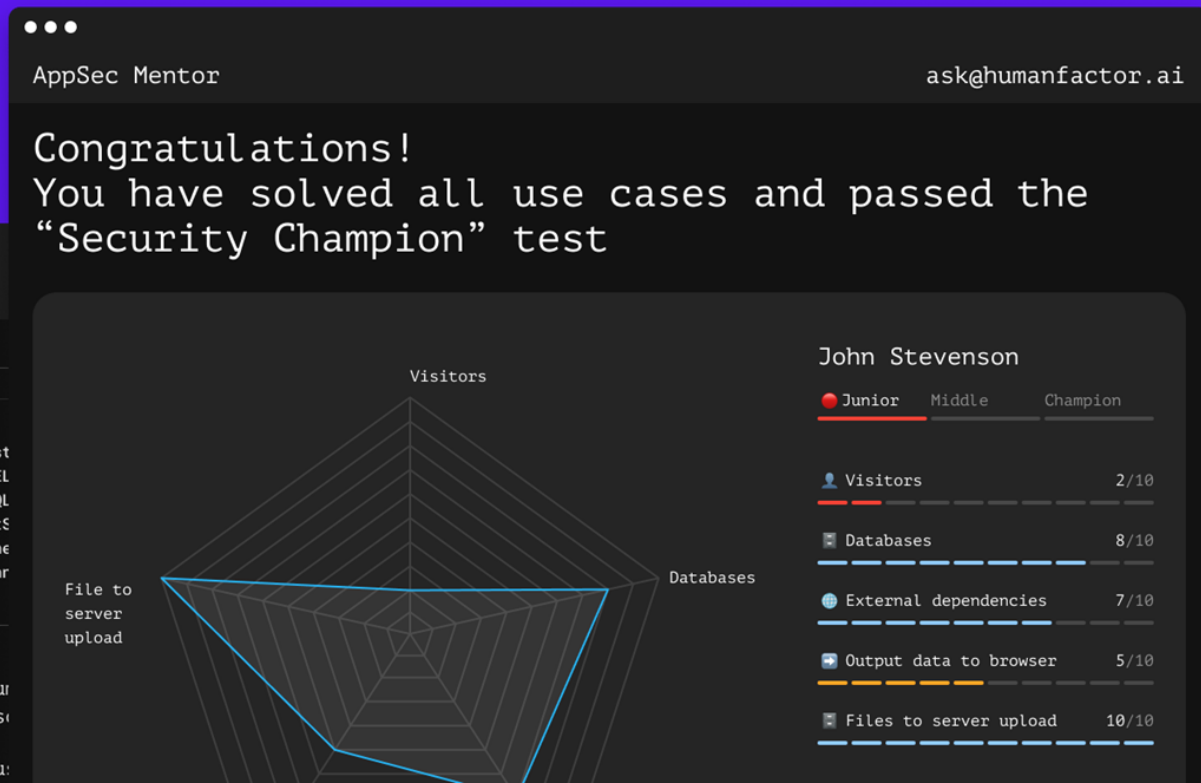
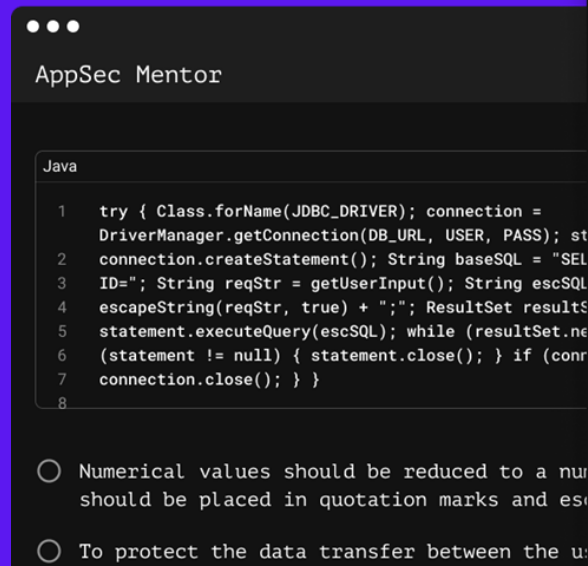


AppSec Mentor — secure code training platform for product teams

Integration



Step zero: Identify your security champions



Step 1: Assemble your project team and establish a knowledge baseline

The screenshot displays a web application interface. On the left is a dark sidebar with a menu containing: 'admin', 'Projects', 'Systems', 'Knowledge' (highlighted), 'Settings', and 'Log out' with an external link icon. The main content area has a top header with 'admin' and 'Warehouse management system'. To the right of the title are 'Delete' and 'Edit' buttons. Below the title is a sub-header with 'Questionnaire', '68 requirements', and 'Training' (underlined). A table lists team members with columns for Name, Role, and Level. The row for 'Juliet Starling' is highlighted. A mouse cursor points to the edit/delete icons at the end of this row.

Name	Role	Level
Kevin Pen	Analyst	Novice
Juliet Starling	Developer	Middle
John Stevenson	AppSec manager	Champion
Oliver Queen	Developer	Novice
Peter Hart	Developer	Novice
Steven Lang	Developer	Novice
Patricia Meldorn	Tester	Novice

Step 2. Define application features and contexts

The screenshot shows a web application configuration interface. On the left is a sidebar with a dark background and white text. It contains a top section with three dots and a 'Logout' button with an external link icon. Below this are menu items: 'admin', 'Projects', 'Systems' (highlighted), 'Knowledge', 'Settings', and 'Logout'. The main content area has a dark background. At the top, it says 'ASM' and 'Name: Information processing system'. There are 'Cancel' and 'Save' buttons in the top right. Below the name, there are three tabs: 'Questionnaire' (active), 'Requirements preview', and 'Training preview'. The 'Questionnaire' tab contains several sections: 'Jira connection' with a 'Disconnected' dropdown; 'Training plan' with a 'Code storing and management' dropdown set to 'git'; 'Code generation' with 'Yes' and 'No' checkboxes; 'Build and delivery' with a 'CI/CD' dropdown; and 'Third-party modules usage' with 'Composer' and 'NPM' buttons. The bottom section is labeled 'Container orchestration'.

admin

Projects

Systems

Knowledge

Settings

Logout

ASM

Name

Information processing system

Questionnaire Requirements preview Training preview

Jira connection

Disconnected

Training plan

Code storing and management

git

Code generation

☐ Yes

☐ No

Build and delivery

CI/CD

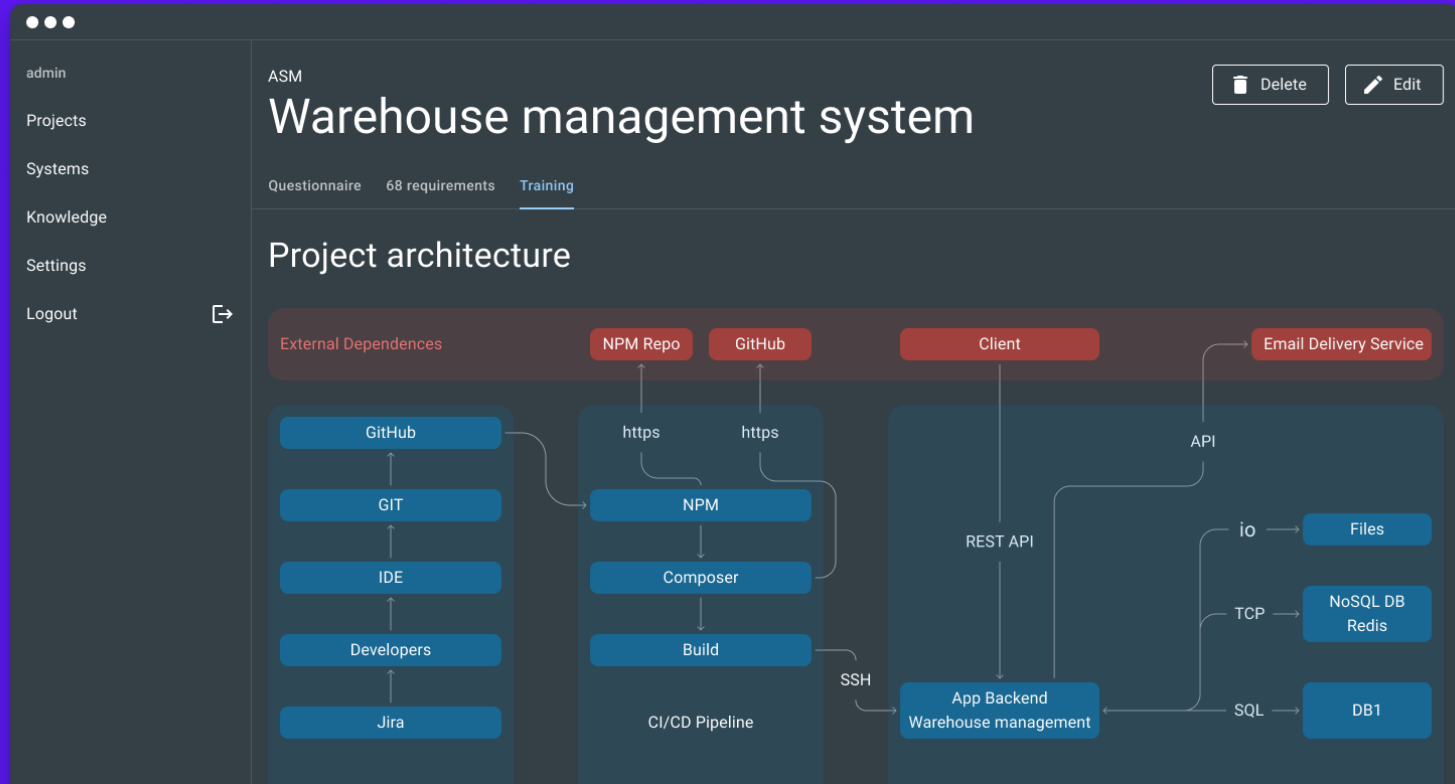
Third-party modules usage

Composer NPM

Container orchestration

Cancel Save

Step 3. Result: project architecture



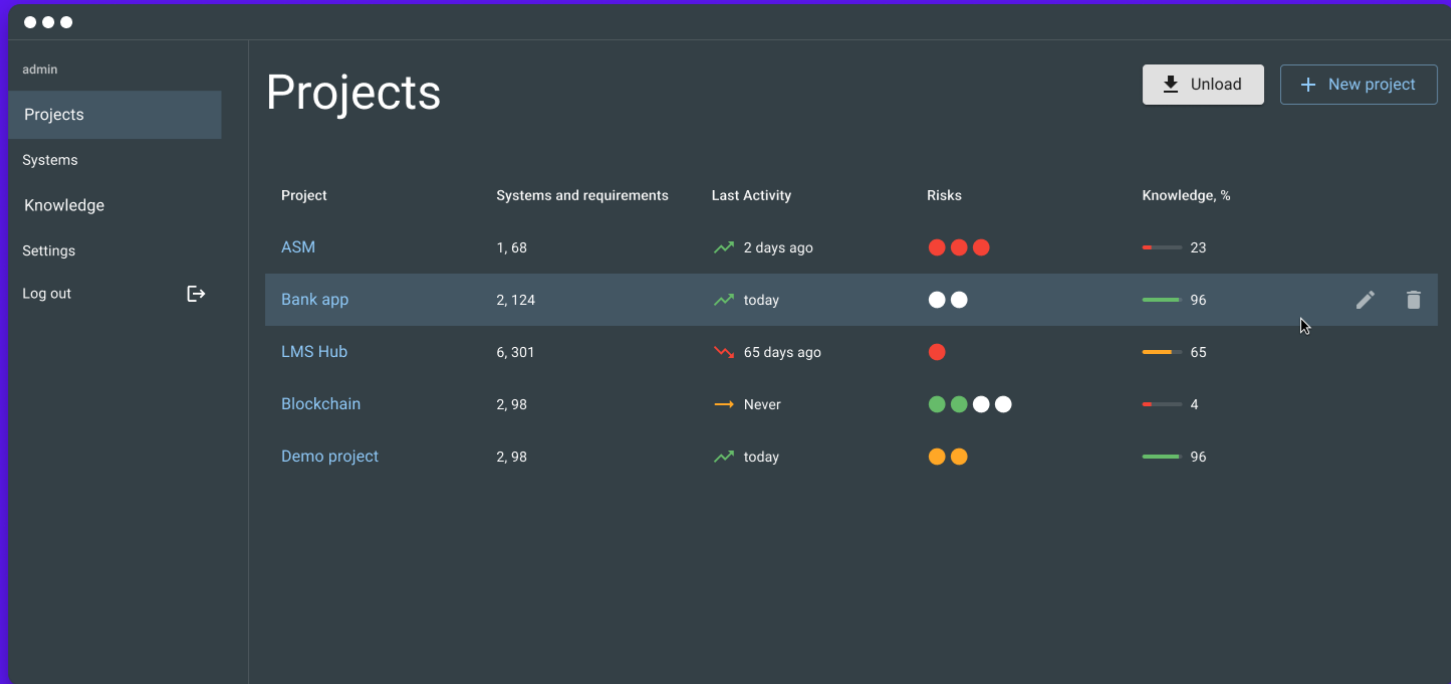
Step 4. Training modules are automatically selected for each context

The screenshot displays a web application interface for a 'Warehouse management system'. On the left is a sidebar menu with options: admin, Projects, Systems, Knowledge, Settings, and Logout. The main content area has a top bar with 'admin', 'Warehouse management system', and 'Delete'/'Edit' buttons. Below this is a sub-header 'Team Competency Map' and a breadcrumb trail: 'Questionnaire' > '68 requirements' > 'Training'.
















The 'Team Competency Map' is organized into three columns, each with a title, a progress bar, and a percentage. The 'Visitors' column is 20% complete, 'External dependencies' is 100%, and 'Databases' is 40%.

Visitors	20%	External dependencies	100%	Databases	40%
Cross-site forgery protection	✓	Securing code dependencies	✓	SQL injection	✓
XML parsers protection	40%	Working with files and directories	✓	noSQL injection	✓
Cookies and HTTP headers	40%	SSRF, remote file inclusion	✓	GraphQL: injection/bypass/dos	✓
Handling visitor requests	0%	File functions and streams	✓	DB clustering 37/50	0%
XXE, xpath injection	0%	LDAP injection	✓	Serialization/deserialization	0%
		CRLF injection	✓		
		SMTP injection	✓		
		Default sharing permissions	✓		

Step 5. Result: clear training plan within the context of the project and the product teams

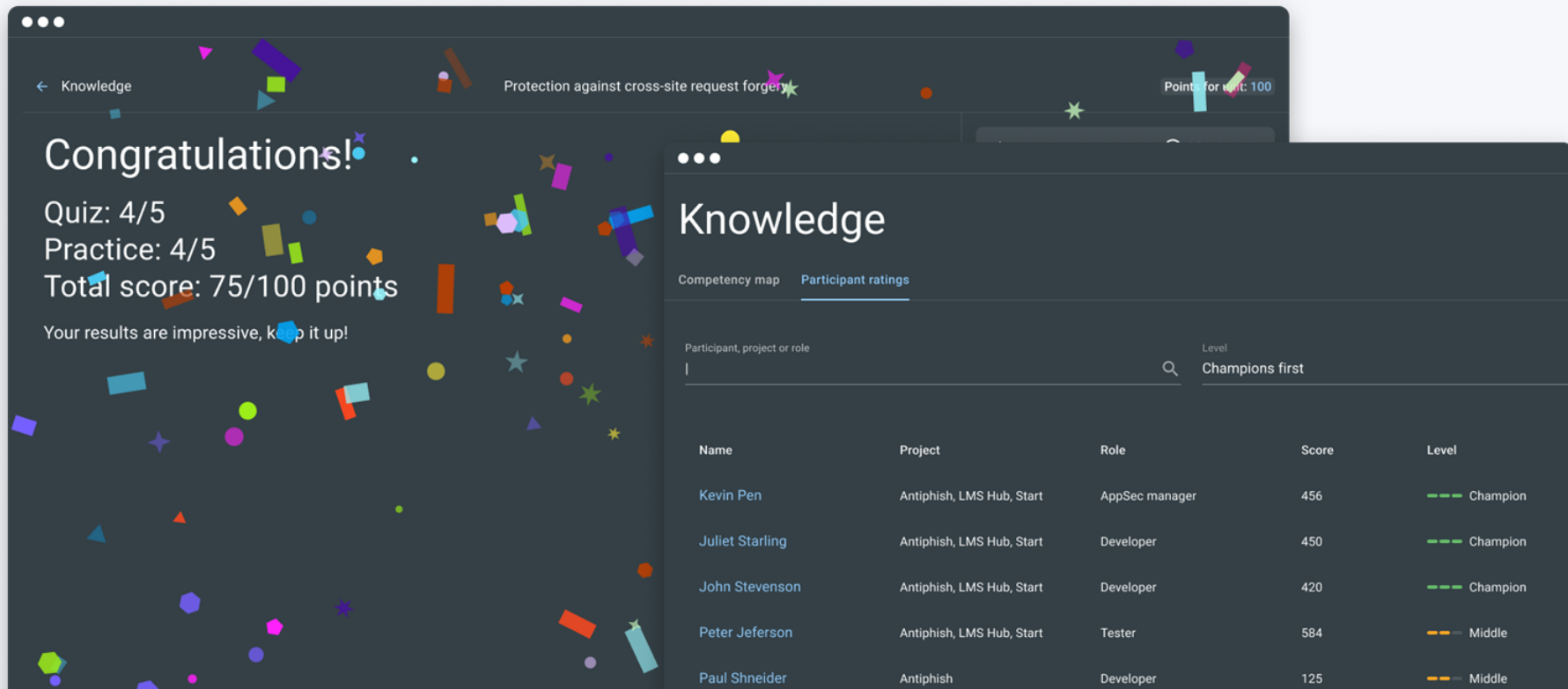


The screenshot displays a web application interface for managing projects. The main content area is titled 'Projects' and contains a table with the following data:

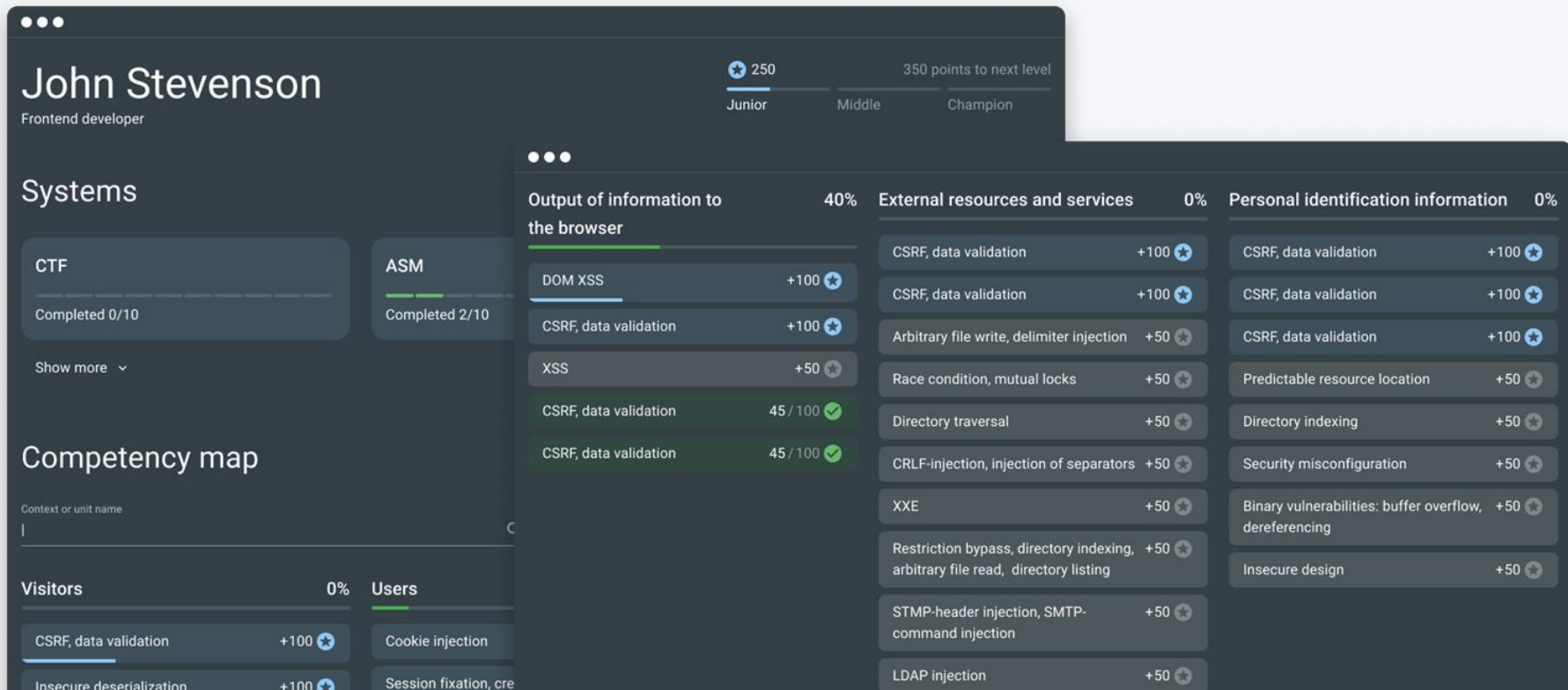
Project	Systems and requirements	Last Activity	Risks	Knowledge, %
ASM	1, 68	 2 days ago		 23
Bank app	2, 124	 today		 96
LMS Hub	6, 301	 65 days ago		 65
Blockchain	2, 98	 Never		 4
Demo project	2, 98	 today		 96

The interface also features a sidebar with navigation links: admin, Projects (selected), Systems, Knowledge, Settings, and Log out. The top bar includes a 'Unload' button and a '+ New project' button. The 'Bank app' row is highlighted, and a mouse cursor is visible over the edit and delete icons in the Knowledge column.

Motivation



Personal competency map



Theory and code samples

Employees gain access to hands-on training modules

Title 2

CSRF defense enhancement techniques

In this section, we collected additional measures to improve your anti-CSRF protection.

Configuring the SameSite attribute

The SameSite attribute of the Set-Cookie HTTP header prevents the browser from sending cookies with cross-site requests. The main purpose of the parameter is to reduce the risk of CSRF attacks.

Question 1

The following code snippet will output:

WITHOUT CODE

PYTHON

JAVASCRIPT

PHP

ASP.NET

PHP

```
1 mt_srand(time()); // Using a random initialization vector for a random number generator
```

```
2 echo mt_rand(); // Using random number generation
```

```
3
```

☐ Random number

Protection against attacks on XML parsers (protection against XXE)

In this section, we will examine the methods to protect against cross-site request forgery:

- Protection based on a secret token
- Built-in framework protection features
- Self-protection: for stateful and stateless applications; using the Encryption Based Token pattern; using the HMAC Based Token pattern

Java

Python

JavaScript

PHP

ASP.NET

Quiz based on real cases

Quiz

Congratulations! You answered 4 out of 5 questions and earned 4 points.

You have mastered the theory and are now ready to practice.

Question 1

The following code snippet will output:

WITHOUT CODE

JAVA

PYTHON

JAVASCRIPT

PHP

ASP.NET

```
PHP
1  mt_srand(time()); // Using a random initialization vector for a random
   number generator
2  echo mt_rand(); // Using random number generation
3
```

Theory30 min

Quiz30 min

Question 1
The following code snippet will output:

Question 2
Encryption based Token

Question 3
Required factors to enable the risk of a CSRF attack

Question 4
Encryption based Token

Practice30 min

Results

Code review practice

Project

> .git

> templates

app.py

Dockerfile

requirements.txt

app.py

1 from flask import Flask, request, url_for

2 from flask_login import

3 LoginManager, UserMixin, current_user, login_required,

4 login_user, logout_user

5 from flask_wtf.csrf import CSRFProtect

6

7 app = Flask(__name__)

8 app.config.update(

9 DEBUG=True,

10 SECRET_KEY="secret_antiCSRF",

11)

12

13 login_manager = LoginManager()

14 login_manager.init_app(app)

15

16 csrf = CSRFProtect()

17 csrf.init_app(app)

18

19 # database

20 users = [

21 {

22 "id": 1,

23 "username": "test",

24 "password": "test"

25 }

26]

app.py

1 from flask import Flask, request, url_for

2 from flask_login import

3 LoginManager, UserMixin, current_user, login_required,

4 login_user, logout_user

5 from flask_wtf.csrf import CSRFProtect

6

7 app = Flask(__name__)

8 app.config.update(

9 DEBUG=True,

10 SECRET_KEY="secret_antiCSRF",

11)

12

13 login_manager = LoginManager()

14 login_manager.init_app(app)

15

16 csrf = CSRFProtect()

17 csrf.init_app(app)

18

19 # database

20 users = [

21 {

22 "id": 1,

23 "username": "test",

24 "password": "test"

25 }

26]

Option 1

Option 2

Option 3

✓ Approve

✗ Deny

Course Creator, Theory

Encoding and escaping in browsers (XSS)

✓ Settings

2 Theory delete

3 Quiz delete

4 Practice delete

5 Result

To draft

Preview


Publish

Clear all

Unit title

Enter the unit title

Image



Titles

Title 1 3 points

+ add subtitle

Title 2 3 points

+ add subtitle

Title 3 3 points

How to use....

3 top methods ...

+ add subtitle

Title 4 3 points

+ add subtitle

Title 5 3 points

+ add subtitle

Tools

Text

Subtitle 1

Encoding and escaping in browsers (XSS)

✓ Settings

2 Theory delete

3 Quiz delete

4 Practice delete

5 Result

System

Unit name

Encoding and escaping in browsers (XSS)

Characteristics

CI/CD

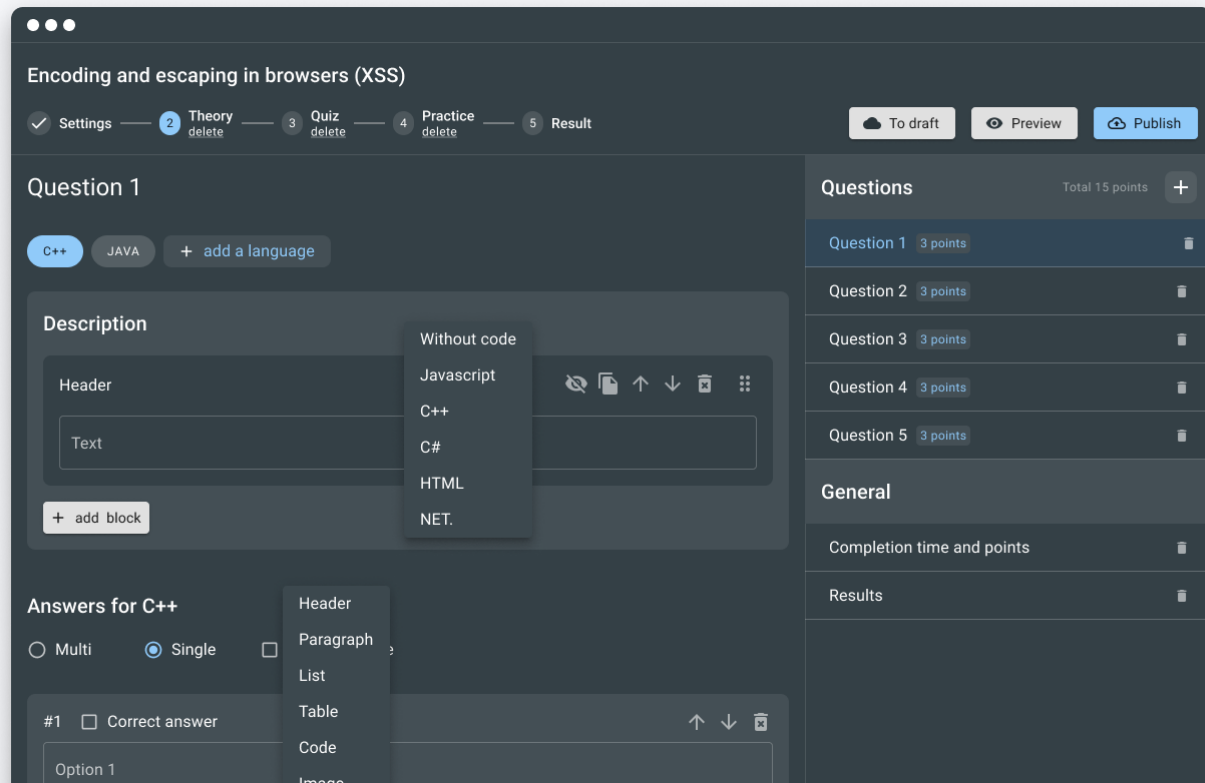
Privacy

Roles

Frontend

Backend

Course Creator, Quiz



Course Creator, Practice

Encoding and escaping in browsers (XSS)

✓ Settings

2 Theory delete

3 Quiz delete

4 Practice delete

5 Result

To draft

Preview

Publish

Question 1

C++

JAVA

+ add a language

☒ Answer as a code

☐ Answer as a text

Question header

Description

1. Section

Text

+ add block

Reset all changes

Upload project

Project

cutils

> Header Files

> Source Files

Graph.cpp

MathImpl.cpp

TestGraph.cpp

> External Files

build.gradle

> New Folder

> New Folder

TestGraph.cpp

```
65 def homepage():
66     if request.method == "POST":
67         username =
68         request.form.get("username")
69         password =
70         request.form.get("password")
71
72         for user in users:
73             if user["username"] ==
74             username and user["password"] ==
75             password:
76                 user_model = User()
77                 user_model.id =
78                 user["id"]
79                 login_user(user_model)
80                 return
81             redirect(url_for("accounts"))
82             return abort(401)
83
84             if current user is authenticated:
```

TestGraph.cpp

Option 1 ✕

Option 2 ✕

Option 3 ✕

+ □ Correct

```
65 def homepage():
66     if request.method == "POST":
67         username =
68         request.form.get("username")
69         password =
70         request.form.get("password")
71
72         for user in users:
73             if user["username"] ==
74             username and user["password"] ==
75             password:
76                 user_model = U
77                 user_model.id =
78                 user["id"]
79                 login_user(user_model)
80                 return
81             redirect(url_for("accounts"))
82             return abort(401)
83
84             if current user is authenticated:
```

Undo

Redo

Cut

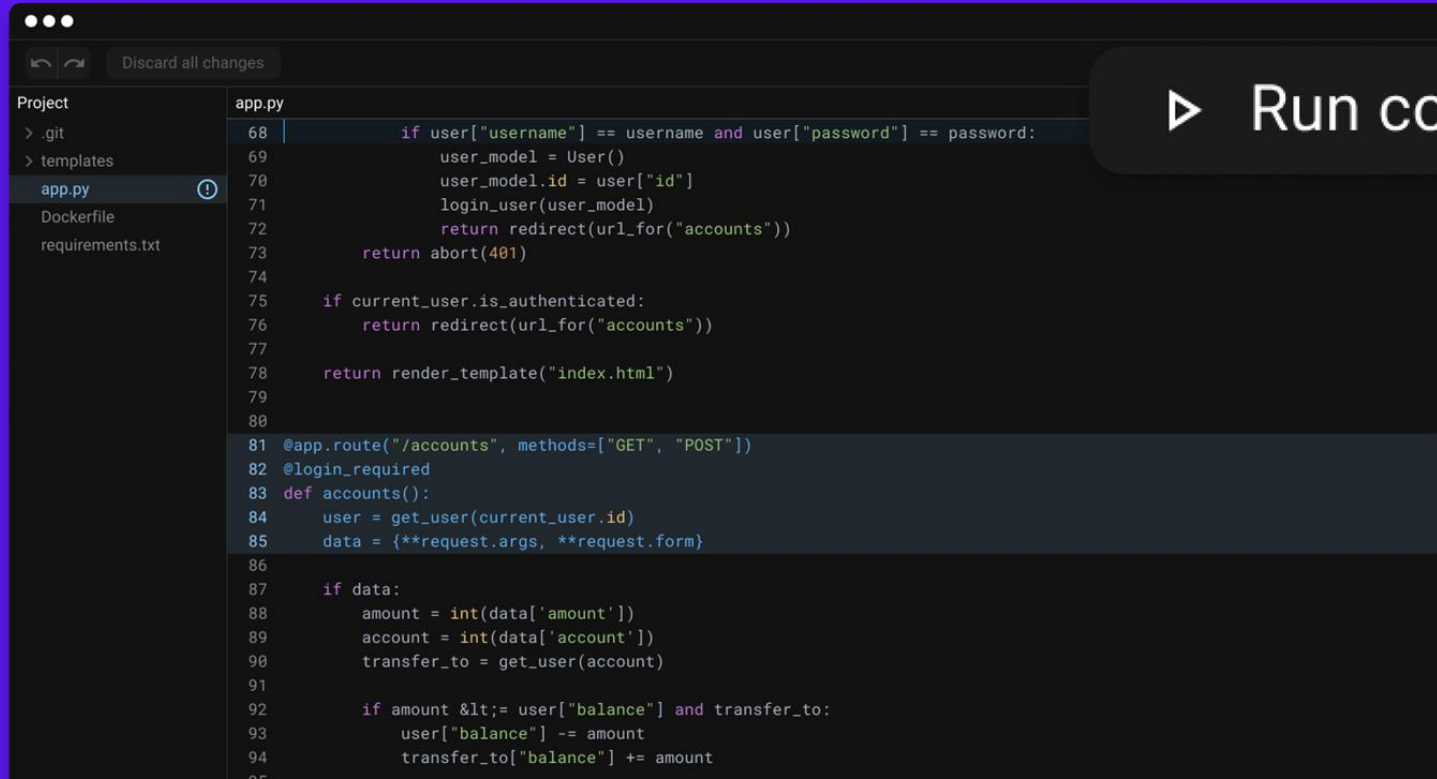
Copy

Paste

Delete

Select all

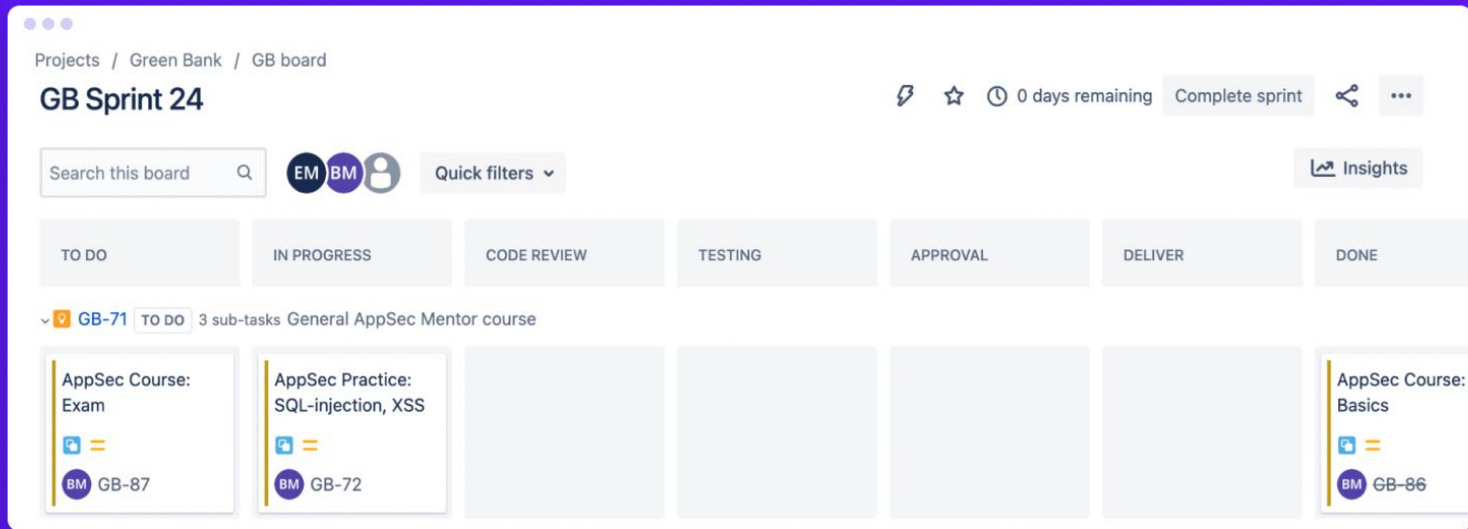
Coming Soon — Practicing writing code



```
68     if user["username"] == username and user["password"] == password:
69         user_model = User()
70         user_model.id = user["id"]
71         login_user(user_model)
72         return redirect(url_for("accounts"))
73     return abort(401)
74
75     if current_user.is_authenticated:
76         return redirect(url_for("accounts"))
77
78     return render_template("index.html")
79
80
81 @app.route("/accounts", methods=["GET", "POST"])
82 @login_required
83 def accounts():
84     user = get_user(current_user.id)
85     data = {**request.args, **request.form}
86
87     if data:
88         amount = int(data['amount'])
89         account = int(data['account'])
90         transfer_to = get_user(account)
91
92         if amount &lt;= user["balance"] and transfer_to:
93             user["balance"] -= amount
94             transfer_to["balance"] += amount
95
```

▶ Run code

Coming Soon — Training tasks delivered to a single task management environment



Everything your developers need to know to write secure code

✓ Protection against cross-site request forgery (CSRF)

✓ Protection against attacks on XML parsers (protection against XXE)

✓ Encoding and escaping when working with browsers (XSS)

✓ Path Manipulation

✓ Secure work with dependencies in the code

✓ Working with SQL

✓ Insecure Direct Object Reference

✓ Secure Kubernetes Setup

✓ Working with Cookies and HTTP headers

✓ Processing of input data

✓ Working with secrets (passwords, keys, tokens)

Convenient practical training in different programming languages

Programming languages

Java Python PHP SQL C#
Go HTML Javascript

Average completion time

Theory	Quiz	Practice
30-60 minutes	10-15 minutes	15-25 minutes
	5-8 questions	

How does AppSec Mentor differ from traditional courses



Automatic training customization

Testing developer skills, automatic training modules selections for the specific project and level of developer experience (within the context of information security).



Training recommendations according to the project map

Create a project map based on project parameters (contexts-situations), with linked vulnerability categories and recommended training modules.



Employee competency map

Build a competency map for each employee

Journey

