CERTLABZ CERTIFICATION BOARD

# Certified Full Stack Engineer (CFSE)

## Ten Sample Questions

These ten questions are written at the difficulty of the live exam. Six carry a figure, a data table, a configuration or a code listing; four are plain scenario questions, as on the live paper. Every one is a scenario a working full stack engineer would meet, and none of them appears in the live question bank.

The live exam is multistage adaptive: how you score on one stage selects the difficulty of the next, so no two candidates sit the same paper. It is a performance-based exam: each form can carry up to 5 to 12 performance-based questions alongside the multiple-choice questions, which means you build, sequence or complete something rather than only choosing from a list. Answers and full explanations follow each question here; the live exam does not disclose item-level answers.

---

### Question 1 — Domain 1: Frontend Fundamentals — DATA

Emma Wilson matches each frontend task to the tool that owns it.

| Task | Tool |
|---|---|
| Page structure | (to choose) |
| Visual styling | (to choose) |
| Interactivity | (to choose) |

Table 1.1 Tasks and their tools.

Which pairing is correct?

A. CSS, HTML, JS
B. JS, CSS, HTML
C. HTML, CSS, JS
D. CSS, JS, HTML

**Correct answer: C**

HTML provides page structure, CSS provides visual styling, and JavaScript provides interactivity. Only option C maps each task to the tool that actually owns it. The other orders swap structure, styling or behavior onto the wrong layer.

## Question 2 Domain 1: Frontend Fundamentals

A page uses a styled div for every clickable control, so keyboard users cannot reach them. What is the soundest fix?

- A Use button elements
- B Add more div wrappers
- C Remove all styling
- D Raise the font size

**Correct answer: A**
Native button elements are focusable and operable by keyboard and screen readers by default, which restores access. Extra div wrappers keep the same problem, removing styling does not add semantics, and font size is unrelated to reachability.

## Question 3 Domain 2: Backend and APIs CONFIG

James Carter reviews an API that returns 200 for every response, including errors.

```
GET /users/999  : 200 OK
body : { error : not found }
DELETE /orders  : 200 OK
on failure      : still 200
```

Listing 3.1 The current responses.

What is the soundest change?

- A Return 200 for all
- B Use correct codes
- C Remove the endpoint
- D Log the error only

**Correct answer: B**
HTTP status codes let clients react correctly, so a missing record should return 404 and a server fault 500 rather than a blanket 200. Keeping 200 everywhere hides failures, deleting the endpoint removes the feature, and logging alone does not inform the client.

## Question 4 Domain 2: Backend and APIs

David Brown finds an endpoint that returns the entire orders table on every call, and it grows slower each week. What is the best fix?

- A Add more servers
- B Increase the timeout
- C Cache the whole table

D Add pagination

**Correct answer: D**
Pagination returns a bounded page of rows per request, so response size and latency stay stable as the table grows. More servers and longer timeouts only mask the cost, and caching a table that keeps growing does not bound the payload.

## Question 5 Domain 3: Data and Persistence DATA

Olivia Davis studies a query that scans a whole table to find one row by key.

| Symptom | Detail |
| --- | --- |
| Lookup by a single key | Full scan |
| Rows keep growing | Yes |
| Key values are unique | Yes |
| No structure on the key | Confirmed |

Table 5.1 A slow key lookup.

What is the best remedy?

A Add an index
B Drop the table
C Add more columns
D Disable the cache

**Correct answer: A**
An index on the key lets the database find the row without scanning every record, which keeps lookups fast as rows grow. Dropping the table removes the data, extra columns add nothing, and the cache is not the cause of the scan.

## Question 6 Domain 3: Data and Persistence CONFIG

Liam Anderson reviews a transfer that writes two rows with no safeguard between them.

```
debit  account A : write
credit account B : write
no wrapper around both
crash between : funds lost
```

Listing 6.1 The current writes.

What is required?

A Retry each write twice
B Ignore the failures
C Wrap in a transaction

D Cache both writes

**Correct answer: C**
A transaction makes the two writes all-or-nothing, so a crash between them rolls both back and money is never lost. Retrying each write alone can double-apply one side, ignoring failures corrupts balances, and caching does not give atomicity.

## Question 7 Domain 4: Integration and Architecture

Ava Thompson finds two services reading and writing each other private database tables directly. What is the better practice?

A Share all the tables
B Use defined APIs
C Merge both services
D Copy the database

**Correct answer: B**
Each service should own its data and expose a defined API, so callers depend on a stable contract rather than a private schema. Sharing tables couples the services tightly, merging them removes separation, and copying the database creates two sources of truth.

## Question 8 Domain 5: Testing, Quality and Developer Experience DATA

Noah Clark charts how many of each test type the team has today.

| Test type | Count today |
|---|---|
| Unit | Few |
| Integration | Some |
| End to end | Many |
| Shape | Inverted |

Table 8.1 Test counts today.

What should change?

A Add more end to end
B Delete all the tests
C Only test by hand
D Grow the unit base

**Correct answer: D**
A healthy suite has many fast unit tests and fewer slow end to end tests, so growing the unit base fixes the inverted shape. Adding more end to end tests makes the suite slower and more brittle, and dropping or manual-only testing removes coverage.

## Question 9 Domain 5: Testing, Quality and Developer Experience

Sarah Miller sees tests that pass locally but fail at random in the shared pipeline, blocking every branch. What is the best first step?

A. Delete failing tests
B. Ignore the pipeline
C. Fix the flaky tests
D. Rerun until green

**Correct answer: C**
Flaky tests must be diagnosed and stabilised, often by removing shared state or timing races, so results become trustworthy again. Deleting them or ignoring the pipeline loses real coverage, and rerunning until green hides the defect and wastes time.

## Question 10 Domain 6: Security, Performance and Deployment CONFIG

A deployment config is checked in with secrets in plain text and traffic over an open channel.

```
db_password = hunter2
api_key     = 1234567890
protocol    = http only
secrets committed to repo
```

Listing 10.1 The current config.

What is required?

A. Hard-code more keys
B. Use env vars and TLS
C. Store secrets in code
D. Turn off encryption

**Correct answer: B**
Secrets belong in environment variables or a managed secret store, never in the repository, and traffic should move over TLS. Hard-coding more keys and storing secrets in code widen the exposure, and turning off encryption removes the protection entirely.

Published by the CertLabz Certification Board, a division of CertLabz LLC. Companion to objectives document CFSE-001. https://certlabz.com/certifications/certified-full-stack-engineer.html