Full-Stack Web Foundations

Ages 16-17 | Future Pros

Plan a small app with a screen, data, and a safe release process.

How to use this guide

Read each section, complete its four lab steps, then answer the questions in the online course. The online progress bar advances only after four correct answers.

Section 1: React

LEARNING GOAL: React. Your finished work should demonstrate: A working task list interface.

EXPLAIN THE CONCEPT: React components split an interface into reusable pieces. State lets a component respond to a user's action. Think about where you might use this skill in a real or imaginary project. Describe the problem it helps you solve and how you would recognize a successful result.

ORIGINAL PROJECT LAB: 1. Create a small app with a list component. 2. Add a form to enter a pretend task. 3. Update state so the new task appears. 4. Check keyboard access and empty input. Finished result: A working task list interface.

Understand with an example

The idea. A user interface should render from coherent state. Derived values can be computed from that state rather than stored separately and allowed to drift. Use stable item identity when items can move or be removed.

Get ready. A fictional task list contains A incomplete and B complete.

Worked example 1

Follow the reasoning. Compute remaining count from incomplete items: the count is 1. Mark A complete and recompute: the count becomes 0. The list and summary use the same source of truth.

Worked example 2: change one thing

Compare the result. Store remaining count separately and forget to update it after marking A. The screen can show all tasks complete and still say one remaining, exposing duplicated-state inconsistency.

Try one action at a time

1. Define item identity and state.
2. Derive the remaining count.
3. Trace one state change.
4. Compare all dependent displays.

A mistake to notice

Pause and check. A list position is not a durable identity when items can be reordered.

Your own small challenge

Predict before trying. Remove completed task B from the original list. What is the remaining count?

Check your thinking

It stays 1 because A is still incomplete. Removing a completed item changes total count but not incomplete count.

Show what you know. Point, draw, demonstrate, type, speak or use your usual communication tool. A trusted adult can help without choosing your answer.

GO DEEPER: Identify an assumption, test a safe alternative, and explain the trade-off. Keep a short before-and-after record that someone else could follow.

SAFETY CHECK: Apply identity security, social-engineering defense, secure coding, cloud controls, incident handling, data protection, and professional ethics. Use pretend data, never real passwords or private information. Stop and ask a trusted adult if anything seems unsafe.

REVIEW QUESTIONS: What is the main idea? Why does the first step matter? How would you detect a mistake? What evidence proves the result? What would you improve? Answer in your own words before taking the quiz.

PORTFOLIO REFLECTION: Save a drawing, sanitized screenshot, or short note describing the goal, your approach, your evidence, and one lesson learned. Expected result: A working task list interface.

Investigate with pictures and choices

A UI shows count zero and a button adds one.

What should happen after one action?

1. Only the variable name changes
2. Every component resets

3. State becomes one and the UI reflects it

Worked reasoning

Answer: State becomes one and the UI reflects it. State changes should produce the intended rendered behavior.

Try a changed example

Test repeated actions and a reset from a known state.

Picture and meaning matches

Component state: Controls rendered UI

Database: Stores constrained records

Deployment: Runs an identified release

Ways to respond

Choose, point, draw, demonstrate, type, speak or use your usual communication tool.

A helper can operate the controls while you choose.

Counting support if useful

Use zero to ten counters or dots. Touch each once. The last count names the quantity. Rearrange the same set and check whether anything was added or removed.

Choose this foundation practice only when it fits the learner goal.

My notes: _____

Hands-on lab

1. Create a small app with a list component.
2. Add a form to enter a pretend task.
3. Update state so the new task appears.
4. Check keyboard access and empty input.

Finished result: A working task list interface.

Check your understanding

1. Which idea best describes React?

A. React components split an interface into reusable pieces.

B. A release makes an app available to others.

C. It has no purpose in this project.

D. A database stores structured records.

2. What is the first step in this lab?

A. Update state so the new task appears.

B. Create a small app with a list component.

C. Check keyboard access and empty input.

D. Add a form to enter a pretend task.

3. What should you do after step two?

A. Check keyboard access and empty input.

B. Create a small app with a list component.

C. Update state so the new task appears.

D. Add a form to enter a pretend task.

4. Which result shows that your lab is finished?

A. A copied answer without checking it

B. A skipped lab

C. A blank page with no work

D. A working task list interface.

Answer review

1. A. React components split an interface into reusable pieces.

2. B. Start with: Create a small app with a list component.

3. C. Step three is: Update state so the new task appears.

4. D. The goal is: A working task list interface.

Section 2: Databases

LEARNING GOAL: Databases. Your finished work should demonstrate: A simple data model and sample records.

EXPLAIN THE CONCEPT: A database stores structured records. Design what belongs in each field and validate input before saving. Think about where you might use this skill in a real or imaginary project. Describe the problem it helps you solve and how you would recognize a successful result.

ORIGINAL PROJECT LAB: 1. Sketch a tasks table with id, label, and done fields. 2. Add three sample records without private data. 3. Write the rule for a valid label. 4. Show how one record changes when marked done. Finished result: A simple data model and sample records.

Understand with an example

The idea. A data model defines entities, keys and relationships. Constraints protect rules that must hold regardless of which interface writes data. A successful form submission does not prove persistence until the stored result is checked.

Get ready. Use fictional tables: projects(id,title) and tasks(id,project_id,title).

Project 7 exists.

Worked example 1

Follow the reasoning. Task 12 with project_id 7 points to an existing project. The project key and task foreign key express the relationship; repeating the project title in every task would create unnecessary consistency work.

Worked example 2: change one thing

Compare the result. Try a task with project_id 99 when project 99 does not exist. Under an enforced foreign-key rule the write should be rejected rather than create an orphan relationship.

Try one action at a time

1. Identify entities and keys.
2. Draw the intended relationship.
3. Predict a valid write.
4. Predict a violated constraint.

A mistake to notice

Pause and check. A schema diagram is not proof that constraints exist in a deployed database.

Your own small challenge

Predict before trying. Can a task title alone reliably identify its project?

Check your thinking

No. Titles may repeat or change. Use the declared project key and preserve the relationship explicitly.

Show what you know. Point, draw, demonstrate, type, speak or use your usual communication tool. A trusted adult can help without choosing your answer.

GO DEEPER: Identify an assumption, test a safe alternative, and explain the trade-off. Keep a short before-and-after record that someone else could follow.

SAFETY CHECK: Apply identity security, social-engineering defense, secure coding, cloud controls, incident handling, data protection, and professional ethics. Use pretend data, never real passwords or private information. Stop and ask a trusted adult if anything seems unsafe.

REVIEW QUESTIONS: What is the main idea? Why does the first step matter? How would

you detect a mistake? What evidence proves the result? What would you improve?
Answer in your own words before taking the quiz.
PORTFOLIO REFLECTION: Save a drawing, sanitized screenshot, or short note describing the goal, your approach, your evidence, and one lesson learned. Expected result: A simple data model and sample records.

Investigate with pictures and choices

Two requests try to create the same unique username.
What protects the stored rule?
1. A comment saying unique
2. A database uniqueness constraint
3. Only a browser warning

Worked reasoning

Answer: A database uniqueness constraint. Server-side and storage enforcement are needed even when the UI validates.

Try a changed example

Test competing requests against a sandbox constraint.

Picture and meaning matches

Component state: Controls rendered UI
Database: Stores constrained records
Deployment: Runs an identified release

Ways to respond

Choose, point, draw, demonstrate, type, speak or use your usual communication tool.
A helper can operate the controls while you choose.

Counting support if useful

Use zero to ten counters or dots. Touch each once. The last count names the quantity. Rearrange the same set and check whether anything was added or removed.
Choose this foundation practice only when it fits the learner goal.
My notes: _____

Hands-on lab

1. Sketch a tasks table with id, label, and done fields.

2. Add three sample records without private data.

3. Write the rule for a valid label.

4. Show how one record changes when marked done.

Finished result: A simple data model and sample records.

Check your understanding

1. Which idea best describes Databases?

A. A database stores structured records.

B. A release makes an app available to others.

C. It has no purpose in this project.

D. React components split an interface into reusable pieces.

2. What is the first step in this lab?

A. Write the rule for a valid label.

B. Sketch a tasks table with id, label, and done fields.

C. Show how one record changes when marked done.

D. Add three sample records without private data.

3. What should you do after step two?

A. Show how one record changes when marked done.

B. Sketch a tasks table with id, label, and done fields.

C. Write the rule for a valid label.

D. Add three sample records without private data.

4. Which result shows that your lab is finished?

A. A copied answer without checking it

B. A skipped lab

C. A blank page with no work

D. A simple data model and sample records.

Answer review

1. A. A database stores structured records.

2. B. Start with: Sketch a tasks table with id, label, and done fields.

3. C. Step three is: Write the rule for a valid label.

4. D. The goal is: A simple data model and sample records.

Section 3: Deployment

LEARNING GOAL: Deployment. Your finished work should demonstrate: A release-ready checklist for the sample app.

EXPLAIN THE CONCEPT: A release makes an app available to others. Check functionality and remove secrets before sharing. Think about where you might use this skill in a

real or imaginary project. Describe the problem it helps you solve and how you would recognize a successful result.
ORIGINAL PROJECT LAB: 1. List the app's required settings. 2. Check the interface at phone and desktop widths. 3. Run through add and complete actions. 4. Prepare a release checklist with no credentials in source files. Finished result: A release-ready checklist for the sample app.

Understand with an example

The idea. A deployment packages a particular code version for an environment. Readiness includes the complete user flow, required configuration and failure behavior. Keep secrets on the appropriate server side and out of public bundles.
Get ready. A fictional task app has a browser interface, server route and database.

Worked example 1

Follow the reasoning. A preview loads, creates a pretend task and reads it back through the intended flow. These are separate checks: page loading alone does not establish that writes and reads work.

Worked example 2: change one thing

Compare the result. Keep the page available but remove database connectivity in the scenario. The interface should show controlled failure; it must not announce a saved task that was never persisted.

Try one action at a time

1. Identify version and environment.
2. Check page and server response.
3. Verify write then read.
4. Record failures and untested paths.

A mistake to notice

Pause and check. A READY build status does not establish every application workflow.

Your own small challenge

Predict before trying. Why verify a production environment after a preview passed?

Check your thinking

Configuration and connected resources can differ. Verify the released version and

essential production flow rather than assuming environment equivalence.
Show what you know. Point, draw, demonstrate, type, speak or use your usual communication tool. A trusted adult can help without choosing your answer.
GO DEEPER: Identify an assumption, test a safe alternative, and explain the trade-off. Keep a short before-and-after record that someone else could follow.
SAFETY CHECK: Apply identity security, social-engineering defense, secure coding, cloud controls, incident handling, data protection, and professional ethics. Use pretend data, never real passwords or private information. Stop and ask a trusted adult if anything seems unsafe.
REVIEW QUESTIONS: What is the main idea? Why does the first step matter? How would you detect a mistake? What evidence proves the result? What would you improve? Answer in your own words before taking the quiz.
PORTFOLIO REFLECTION: Save a drawing, sanitized screenshot, or short note describing the goal, your approach, your evidence, and one lesson learned. Expected result: A release-ready checklist for the sample app.

Investigate with pictures and choices

Staging tests artifact H1, but production rebuilds H2.
What needs review?
1. Whether the tested identity is what was released
2. Only the project logo
3. Whether both files end in zip

Worked reasoning

Answer: Whether the tested identity is what was released. Release evidence applies to a specific artifact and configuration.

Try a changed example

Record a checksum and reviewed environment differences.

Picture and meaning matches

Component state: Controls rendered UI
Database: Stores constrained records
Deployment: Runs an identified release

Ways to respond

Choose, point, draw, demonstrate, type, speak or use your usual communication tool.

A helper can operate the controls while you choose.

Counting support if useful

Use zero to ten counters or dots. Touch each once. The last count names the quantity. Rearrange the same set and check whether anything was added or removed.

Choose this foundation practice only when it fits the learner goal.

My notes: _____

Hands-on lab

1. List the app's required settings.
2. Check the interface at phone and desktop widths.
3. Run through add and complete actions.
4. Prepare a release checklist with no credentials in source files.

Finished result: A release-ready checklist for the sample app.

Check your understanding

1. Which idea best describes Deployment?

A. A database stores structured records.

B. It has no purpose in this project.

C. React components split an interface into reusable pieces.

D. A release makes an app available to others.

2. What is the first step in this lab?

A. List the app's required settings.

B. Prepare a release checklist with no credentials in source files.

C. Check the interface at phone and desktop widths.

D. Run through add and complete actions.

3. What should you do after step two?

A. List the app's required settings.

B. Run through add and complete actions.

C. Check the interface at phone and desktop widths.

D. Prepare a release checklist with no credentials in source files.

4. Which result shows that your lab is finished?

A. A skipped lab

B. A blank page with no work

C. A release-ready checklist for the sample app.

D. A copied answer without checking it

Answer review

1. D. A release makes an app available to others.
2. A. Start with: List the app's required settings.
3. B. Step three is: Run through add and complete actions.
4. C. The goal is: A release-ready checklist for the sample app.

Section 4: Guided Project & Worked Examples

COURSE INVESTIGATION: Authorization belongs at the resource boundary

YOUR PRACTICE INPUT: A synthetic app displays project P1 for learner U1 and project P2 for U2. A caller changes a URL from P1 to P2.

WHY IT WORKS: Hiding a link does not restrict the server route. The server must verify the caller's permission to the requested project.

PLAN BEFORE YOU ACT: Draw or write the starting state, the question you need to answer, and the evidence you expect to see. A paper trace is acceptable; label it as a prediction unless you actually run a test.

WORK THE CASE: Create an allow/deny matrix for authenticated owner, another user, and signed-out caller. Trace the server decision before data retrieval.

COMPARE WITH THE EXPECTED RESULT: Only an authorized identity receives the project. Tests include changing both URL and body values.

EXPLAIN YOUR REASONING: Show the connection between the supplied input, each decision, and your result. If a number, label, or observation differs, return to the first differing step rather than changing several things together.

KEEP YOUR EVIDENCE: Save the initial plan, a drawing or sanitized output, and one explanation of what the result does and does not prove. Use pointing, speech, drawing, typing, or your usual communication method.

LEARNING GOAL: Guided Project & Worked Examples. Your finished work should demonstrate: A completed guided project with documented evidence: Architecture diagram, Application/repository, Test/security checklist, Deployment/readme evidence.

EXPLAIN THE CONCEPT: Review the course concepts through two applied examples. Trace a request from UI to server/API to database and back, identifying trust boundaries. Compare client-only validation with server-side validation and authorization. Guided project: Build a small database-backed app with UI, server route, validated input, authorization concept, safe error handling, and deployment plan. Work with synthetic

or fictional information and ask an adult for help when appropriate. Think about where you might use this skill in a real or imaginary project. Describe the problem it helps you solve and how you would recognize a successful result.
ORIGINAL PROJECT LAB: 1. Read both worked examples and explain the reasoning. 2. Build a small database-backed app with UI, server route, validated input, authorization concept, safe error handling, and deployment plan. 3. Record the steps you completed and the result you observed. 4. Compare your result to the goal and correct one issue. Finished result: A completed guided project with documented evidence: Architecture diagram, Application/repository, Test/security checklist, Deployment/readme evidence.

Understand with an example

The idea. A full-stack task flow joins UI state, server validation and stored data. Define where each rule is enforced and what evidence proves the user-visible result. Get ready. Plan adding task Read for existing fictional project 7. Empty titles are invalid.

Worked example 1

Follow the reasoning. The server validates the title and project, stores the task and returns its identifier. A subsequent read supplies the list displayed by the UI. Announce success only after the defined save outcome.

Worked example 2: change one thing

Compare the result. Submit an empty title. The server rejects it and the visible list stays unchanged. Client-side checks can help users, but the server must enforce the rule independently.

Try one action at a time

1. Define validation and relationship rules.
2. Trace a valid save and read.
3. Trace an invalid request.
4. Compare visible and stored state.

A mistake to notice

Pause and check. Do not collapse different stages into a single unsupported success claim.

Your own small challenge

Predict before trying. A save succeeds but the follow-up read fails. What can the UI honestly say?

Check your thinking

It may report the confirmed save and the separate refresh failure, with a safe retry path. Do not imply the list is current when that read failed.

Show what you know. Point, draw, demonstrate, type, speak or use your usual communication tool. A trusted adult can help without choosing your answer.

GO DEEPER: Identify an assumption, test a safe alternative, and explain the trade-off. Keep a short before-and-after record that someone else could follow.

SAFETY CHECK: Apply identity security, social-engineering defense, secure coding, cloud controls, incident handling, data protection, and professional ethics. Use pretend data, never real passwords or private information. Stop and ask a trusted adult if anything seems unsafe.

REVIEW QUESTIONS: What is the main idea? Why does the first step matter? How would you detect a mistake? What evidence proves the result? What would you improve? Answer in your own words before taking the quiz.

PORTFOLIO REFLECTION: Save a drawing, sanitized screenshot, or short note describing the goal, your approach, your evidence, and one lesson learned. Expected result: A completed guided project with documented evidence: Architecture diagram, Application/repository, Test/security checklist, Deployment/readme evidence.

Investigate with pictures and choices

A hidden admin button has no server permission check.

Is the operation protected?

1. Yes, hidden means secure
2. Only on a large screen
3. No, the server must authorize it

Worked reasoning

Answer: No, the server must authorize it. UI visibility does not stop a direct unauthorized request.

Try a changed example

Test a forbidden server request in an approved mock.

Picture and meaning matches

Component state: Controls rendered UI

Database: Stores constrained records

Deployment: Runs an identified release

Ways to respond

Choose, point, draw, demonstrate, type, speak or use your usual communication tool.

A helper can operate the controls while you choose.

Counting support if useful

Use zero to ten counters or dots. Touch each once. The last count names the quantity. Rearrange the same set and check whether anything was added or removed.

Choose this foundation practice only when it fits the learner goal.

My notes: _____

Hands-on lab

1. Read both worked examples and explain the reasoning.
2. Build a small database-backed app with UI, server route, validated input, authorization concept, safe error handling, and deployment plan.
3. Record the steps you completed and the result you observed.
4. Compare your result to the goal and correct one issue.

Finished result: A completed guided project with documented evidence: Architecture diagram, Application/repository, Test/security checklist, Deployment/readme evidence.

Check your understanding

1. Which idea best describes Guided Project & Worked Examples?

A. React components split an interface into reusable pieces.

B. Review the course concepts through two applied examples.

C. A database stores structured records.

D. It has no purpose in this project.

2. What is the first step in this lab?

A. Build a small database-backed app with UI, server route, validated input, authorization concept, safe error handling, and deployment plan.

B. Record the steps you completed and the result you observed.

C. Read both worked examples and explain the reasoning.

D. Compare your result to the goal and correct one issue.

3. What should you do after step two?

A. Build a small database-backed app with UI, server route, validated input, authorization concept, safe error handling, and deployment plan.

B. Compare your result to the goal and correct one issue.

C. Read both worked examples and explain the reasoning.

D. Record the steps you completed and the result you observed.

4. Which result shows that your lab is finished?

A. A completed guided project with documented evidence: Architecture diagram, Application/repository, Test/security checklist, Deployment/readme evidence.

B. A copied answer without checking it

C. A skipped lab

D. A blank page with no work

Answer review

1. B. Review the course concepts through two applied examples.

2. C. Start with: Read both worked examples and explain the reasoning.

3. D. Step three is: Record the steps you completed and the result you observed.

4. A. The goal is: A completed guided project with documented evidence: Architecture diagram, Application/repository, Test/security checklist, Deployment/readme evidence.

Section 5: Independent Challenge & Mastery

INDEPENDENT TRANSFER CASE: Authorization belongs at the resource boundary

KEEP THE ORIGINAL BASELINE: A synthetic app displays project P1 for learner U1 and project P2 for U2. A caller changes a URL from P1 to P2.

CHANGE ONE CONDITION: The caller sends ownerId U2 in the request body. User-controlled ownership fields cannot grant access.

YOUR TASK: Create an allow/deny matrix for authenticated owner, another user, and signed-out caller. Trace the server decision before data retrieval.

PREDICT, THEN CHECK: State which part of your original reasoning should remain true and which part must be recalculated or retested. Keep the original and changed results side by side.

REVIEW CRITERIA: Only an authorized identity receives the project. Tests include changing both URL and body values.

MENTOR DISCUSSION: Explain why the changed condition matters. Identify a tempting wrong conclusion, show the observation that would challenge it, and choose one next

test. A statement of uncertainty is better than inventing a result.
PORTFOLIO NOTE: Keep your baseline, changed input, prediction, observed or paper-traced result, explanation, and one improvement. These practice prompts do not replace the course's existing quiz or final assessment.
LEARNING GOAL: Independent Challenge & Mastery. Your finished work should demonstrate: An independent demonstration with evidence: Architecture diagram, Application/repository, Test/security checklist, Deployment/readme evidence.
EXPLAIN THE CONCEPT: Apply what you learned without copying the guided solution. Independent challenge: Threat-model the app for broken access control, injection, exposed secrets, unsafe data, and dependency risk; fix at least two findings. Safety and responsibility: Never place service credentials in client code; minimize stored personal data and require authorization for every protected server action. Reflect on your work: What did you build or discover? What did you test or change after feedback? What safe or responsible choice did you make? Think about where you might use this skill in a real or imaginary project. Describe the problem it helps you solve and how you would recognize a successful result.
ORIGINAL PROJECT LAB: 1. Explain the challenge and make a short plan. 2. Threat-model the app for broken access control, injection, exposed secrets, unsafe data, and dependency risk; fix at least two findings. 3. Test the result and record evidence of your work. 4. Review the safety scenario and describe one improvement. Finished result: An independent demonstration with evidence: Architecture diagram, Application/repository, Test/security checklist, Deployment/readme evidence.

Understand with an example

The idea. A project demonstration should cover the entire path with sanitized evidence and explicit limits. State what the user does, what the server accepts and what the stored record proves.
Get ready. Design a fictional project-and-task app with valid, invalid and unavailable-service scenarios.

Worked example 1

Follow the reasoning. One test adds a valid task, reads it back and checks its project key. Another sends an invalid title and checks that no new task appears. These verify different requirements.

Worked example 2: change one thing

Compare the result. Simulate an unavailable database. Expect controlled feedback and no invented saved identifier. Record any actual retry behavior separately from the intended design.

Try one action at a time

1. Write three acceptance scenarios.
2. Predict state at every layer.
3. Collect evidence when implemented.
4. Explain one limitation or recovery path.

A mistake to notice

Pause and check. An attractive interface does not prove data integrity or service availability.

Your own small challenge

Predict before trying. What should a reviewer receive besides a screenshot of the homepage?

Check your thinking

Provide the relevant version, setup assumptions, input/output checks and sanitized evidence of the complete flow, marking any unrun checks clearly.
Show what you know. Point, draw, demonstrate, type, speak or use your usual communication tool. A trusted adult can help without choosing your answer.
GO DEEPER: Identify an assumption, test a safe alternative, and explain the trade-off. Keep a short before-and-after record that someone else could follow.
SAFETY CHECK: Apply identity security, social-engineering defense, secure coding, cloud controls, incident handling, data protection, and professional ethics. Use pretend data, never real passwords or private information. Stop and ask a trusted adult if anything seems unsafe.
REVIEW QUESTIONS: What is the main idea? Why does the first step matter? How would you detect a mistake? What evidence proves the result? What would you improve? Answer in your own words before taking the quiz.
PORTFOLIO REFLECTION: Save a drawing, sanitized screenshot, or short note describing the goal, your approach, your evidence, and one lesson learned. Expected result: An

independent demonstration with evidence: Architecture diagram, Application/repository, Test/security checklist, Deployment/readme evidence.

Investigate with pictures and choices

An update removes a database field older code still reads.

What should a rollout consider?

1. Deleting all older tests
2. Backward compatibility and staged migration
3. Only the newest screenshot

Worked reasoning

Answer: Backward compatibility and staged migration. Old and new releases may overlap, so data changes need a compatible transition.

Try a changed example

Write a rollback plan that includes schema compatibility.

Picture and meaning matches

Component state: Controls rendered UI

Database: Stores constrained records

Deployment: Runs an identified release

Ways to respond

Choose, point, draw, demonstrate, type, speak or use your usual communication tool. A helper can operate the controls while you choose.

Counting support if useful

Use zero to ten counters or dots. Touch each once. The last count names the quantity. Rearrange the same set and check whether anything was added or removed.

Choose this foundation practice only when it fits the learner goal.

My notes: _____

Hands-on lab

1. Explain the challenge and make a short plan.
2. Threat-model the app for broken access control, injection, exposed secrets, unsafe data, and dependency risk; fix at least two findings.
3. Test the result and record evidence of your work.

4. Review the safety scenario and describe one improvement.

Finished result: An independent demonstration with evidence: Architecture diagram, Application/repository, Test/security checklist, Deployment/readme evidence.

Check your understanding

1. Which idea best describes Independent Challenge & Mastery?

A. It has no purpose in this project.

B. React components split an interface into reusable pieces.

C. Apply what you learned without copying the guided solution.

D. A database stores structured records.

2. What is the first step in this lab?

A. Review the safety scenario and describe one improvement.

B. Threat-model the app for broken access control, injection, exposed secrets, unsafe data, and dependency risk; fix at least two findings.

C. Test the result and record evidence of your work.

D. Explain the challenge and make a short plan.

3. What should you do after step two?

A. Test the result and record evidence of your work.

B. Threat-model the app for broken access control, injection, exposed secrets, unsafe data, and dependency risk; fix at least two findings.

C. Review the safety scenario and describe one improvement.

D. Explain the challenge and make a short plan.

4. Which result shows that your lab is finished?

A. A blank page with no work

B. An independent demonstration with evidence: Architecture diagram, Application/repository, Test/security checklist, Deployment/readme evidence.

C. A copied answer without checking it

D. A skipped lab

Answer review

1. C. Apply what you learned without copying the guided solution.

2. D. Start with: Explain the challenge and make a short plan.

3. A. Step three is: Test the result and record evidence of your work.

4. B. The goal is: An independent demonstration with evidence: Architecture diagram, Application/repository, Test/security checklist, Deployment/readme evidence.

Mastery & safe-use plan

Ages 16-17

Digital safety: Apply identity security, social-engineering defense, secure coding,

cloud controls, incident handling, data protection, and professional ethics.
AI literacy: Use AI as an accountable engineering assistant: validate outputs, protect data, document use, test security, and keep humans responsible.
Challenge: Work through a realistic ticket, incident, architecture, data, or application scenario with professional evidence.
Portfolio: Produce a career-ready artifact such as a repository, incident report, architecture diagram, dashboard, automation, or technical brief.
Mastery check: Present the work as if to a mentor or interviewer: problem, approach, evidence, tradeoffs, security, and next improvement.

Project worksheet

My project: Launch a database-powered app
Who will use or review it? _____
My plan: _____
Checkpoint 1: A working task list interface.
Checkpoint 2: A simple data model and sample records.
Checkpoint 3: A release-ready checklist for the sample app.
Checkpoint 4: A completed guided project with documented evidence: Architecture diagram, Application/repository, Test/security checklist, Deployment/readme evidence.
Checkpoint 5: An independent demonstration with evidence: Architecture diagram, Application/repository, Test/security checklist, Deployment/readme evidence.
One thing I tested: _____
One thing I changed after testing: _____
What I learned: _____
Trusted adult or mentor feedback: _____

Family review

Show your finished work to a trusted adult. Explain one thing you learned and one thing you would improve.

Beat IT Kids | For learning at home. Do not put real passwords or private details in practice projects.