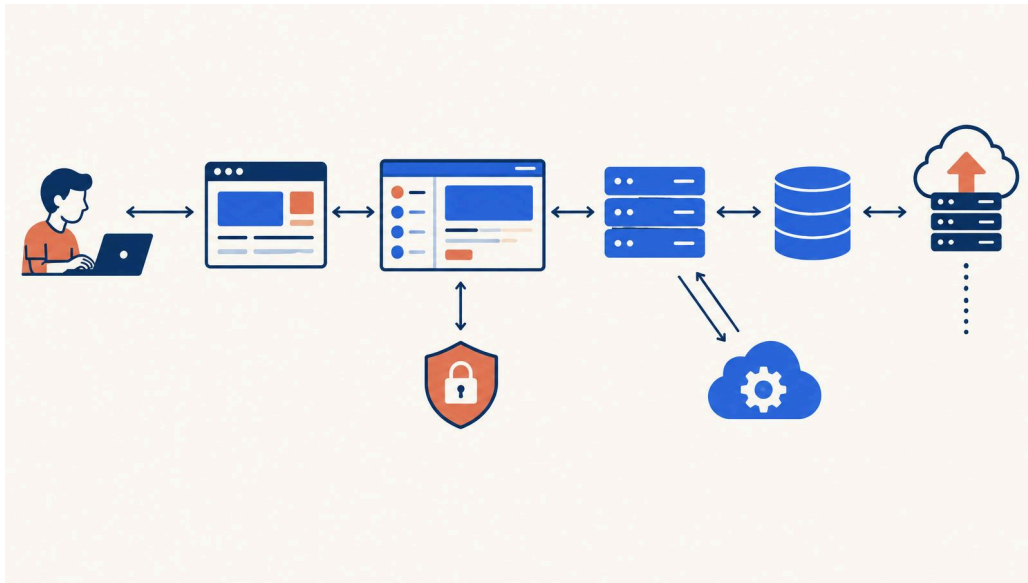


AGILEMINDS HUB

Software Engineering & Vibe Coding

Foundations for AI-Assisted Builders



Understand how software works before you build it.

Concepts • Systems • Responsible AI-assisted development

1 Contents

1	Contents	2
2	Course at a glance	1
2.1	Course promise	1
2.2	What this course does not promise	1
3	Learning outcomes	1
4	The system behind the screen	1
4.1	Concept anchors	2
5	Module map	2
6	Module 00 · Orientation	3
6.1	Lessons	3
7	Module 01 · From Idea to Buildable Product	3
7.1	Lessons	3
8	Module 02 · How Modern Software Works	4
8.1	Lessons	4
9	Module 03 · Understanding AI Coding Agents	4
9.1	Lessons	5
10	Module 04 · Understanding Development Tools	5
10.1	Lessons	5
11	Module 05 · UI/UX and Frontend Product Thinking	6
11.1	Lessons	6
12	Module 06 · Databases and Data Modeling	6
12.1	Lessons	6
13	Module 07 · APIs and Backend Engineering	7
13.1	Lessons	7
14	Module 08 · Git, GitHub, and Version Control	7
14.1	Lessons	7
15	Module 09 · Environment Variables and Application Security	8
15.1	Lessons	8
16	Module 10 · Testing, Debugging, and Quality	8
16.1	Lessons	8
17	Module 11 · Deployment Concepts and Operating Software	9
17.1	Lessons	9
18	Module 12 · Foundations Review and Course Two Bridge	9
18.1	Review sequence	9
19	Assessment and learner support	9
19.1	Suggested resources	10
20	Course Two relationship	10
21	Glossary snapshot	10

2 Course at a glance

This course gives founders, product managers, aspiring builders, and newcomers to technology a clear, high-level mental model of software engineering and vibe coding. It explains how applications are structured, what AI agents do, and why interfaces, APIs, databases, GitHub, security, testing, and deployment matter before Course Two.

Format	Audience	Positioning
Foundations course	Founders, PMs, aspiring builders, beginners	Conceptual software-engineering literacy for responsible vibe coding
Suggested duration	10 weeks; 5–7 hours weekly	Concepts, demonstrations, guided analysis, and reflection

2.1 Course promise

By the end of the course, learners will understand the concepts, vocabulary, systems, decisions, and responsibilities involved in software engineering and vibe coding. They will be prepared to question AI-generated work, reason about technical choices, and enter Course Two with a strong foundation.

“The goal is not to make learners dependent on AI-generated code. The goal is to help them use AI as a capable engineering assistant while retaining human ownership of the product, architecture, security, quality, and decisions.”

2.2 What this course does not promise

This course does not promise a completed production application, a deployed e-commerce product, or mastery of professional software engineering in ten weeks. Those practical outcomes belong to Course Two, where learners will build a complete e-commerce application in practice.

3 Learning outcomes

Learners will be able to:

1. Understand how modern software works across frontend, backend, databases, APIs, authentication, and hosting.
2. Use vibe coding and AI agents responsibly with clear instructions, context, constraints, and verification.
3. Transform product ideas into problem statements, user journeys, requirements, data models, and implementation plans.
4. Understand GitHub, environment variables, security, testing, debugging, deployment, and maintenance.
5. Decide what to build, what to validate, and when a simpler solution is better than software.

4 The system behind the screen

The diagrams in this course are designed as visual anchors. Use them to explain the relationships between the parts of a product before discussing implementation details.

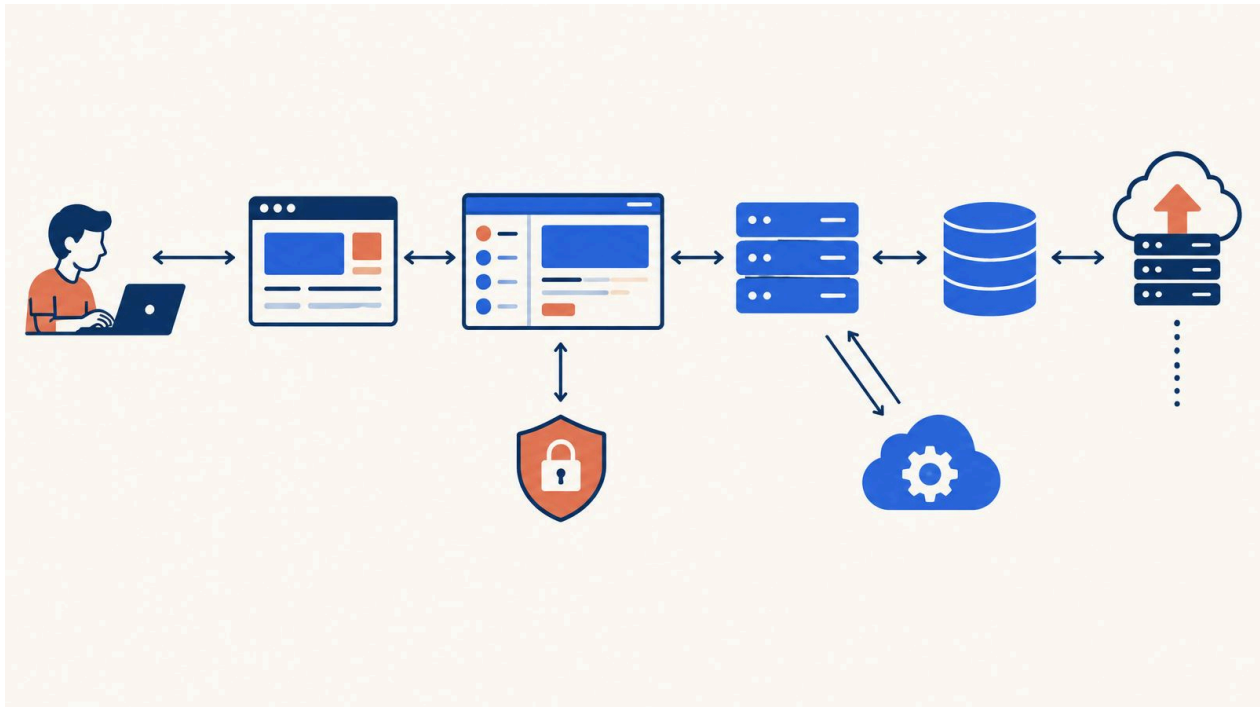


Figure 1: A modern software system connects the user interface to services, data, security, and hosting.

4.1 Concept anchors

APIs are contracts

An API describes how one part of a system requests something from another part, what input is accepted, and what response or error is returned.

AI needs supervision

AI agents can accelerate implementation, but the human still provides context, defines correctness, reviews changes, and owns the outcome.

Data needs structure

Databases are not just storage boxes; they preserve the information, relationships, rules, and history a product depends on.

Shipping includes responsibility

Security, tests, secrets, deployment, monitoring, and maintenance are part of understanding software—not optional extras.

5 Module map

The course is organized as a progression from vocabulary and mental models to responsible AI collaboration and readiness for Course Two. Open each module in the interactive companion for the full lesson list.

No.	Module	Core question
00	Orientation	What are we actually learning?
01	From Idea to Buildable Product	Is the problem clear enough to think about?
02	How Modern Software Works	What happens behind the screen?
03	Understanding AI Coding Agents	What are AI agents and how can they assist?
04	Understanding Development Tools	What role does each tool play?
05	UI/UX and Frontend Thinking	How does a product help users act?

06	Databases and Data Modeling	What must the product remember?
07	APIs and Backend Engineering	How do systems communicate?
08	Git, GitHub, and Version Control	How do we manage change safely?
09	Environment Variables and Security	What can go wrong and how do we reduce risk?
10	Testing, Debugging, and Quality	How do we know it works?
11	Deployment Concepts	What happens when software leaves the laptop?
12	Foundations Review and Course Two Bridge	Are we ready to build in practice?

6 Module 00 · Orientation

CORE IDEA

Vibe coding is not simply asking an AI to write code. It is an AI-assisted way of participating in software creation, where the learner must still define the problem, establish constraints, understand the system, verify the output, and take responsibility for the result.

6.1 Lessons

1. Welcome to AgileMinds Hub: understand the course philosophy and learning journey.
2. What is software engineering? See why software engineering is more than writing code.
3. What is vibe coding? Define AI-assisted development and its limits.
4. The future of vibe coding: explore how AI may change workflows and technical judgment.
5. The vibe-coding responsibility model: own the problem, definition of correct, security, verification, and results.

Learning artifact: A personal learning goal and course map.

7 Module 01 · From Idea to Buildable Product

Before a learner asks an AI agent to build, they need to know what they are trying to solve and what evidence would justify moving forward.

7.1 Lessons

1. Start with the problem and separate enthusiasm from evidence.
2. Define the user and user journey.
3. Scope the first version and make out-of-scope decisions.
4. Write a product brief containing the problem, user, workflow, constraints, and success signal.
5. Convert the brief into requirements, acceptance criteria, edge cases, and open questions.

Learning artifact: A product brief and first-version requirements document.

8 Module 02 · How Modern Software Works

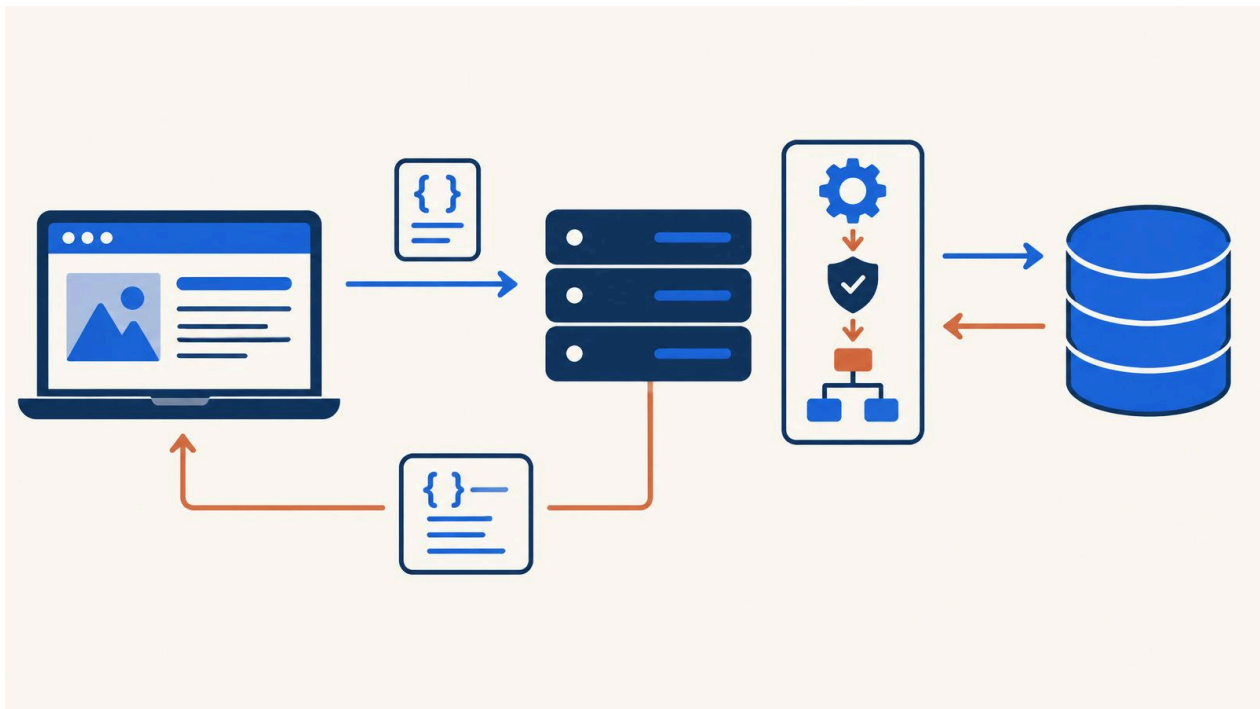


Figure 2: A request moves through the interface, server logic, validation, data, and response.

8.1 Lessons

1. The anatomy of a web application: browser, frontend, backend, database, APIs, authentication, hosting, and external services.
2. Request and response flow: follow a user action from the browser to the server and back.
3. Frontend, backend, and full-stack responsibilities.
4. Monoliths, services, and architecture choices.
5. Technical trade-offs involving speed, cost, quality, complexity, and risk.

Learning artifact: An annotated architecture map.

9 Module 03 · Understanding AI Coding Agents

AI agents can help people explore ideas, explain code, suggest solutions, and speed up software work. They are assistants, not authorities. This module introduces their role at a high level and explains why human judgment remains necessary.

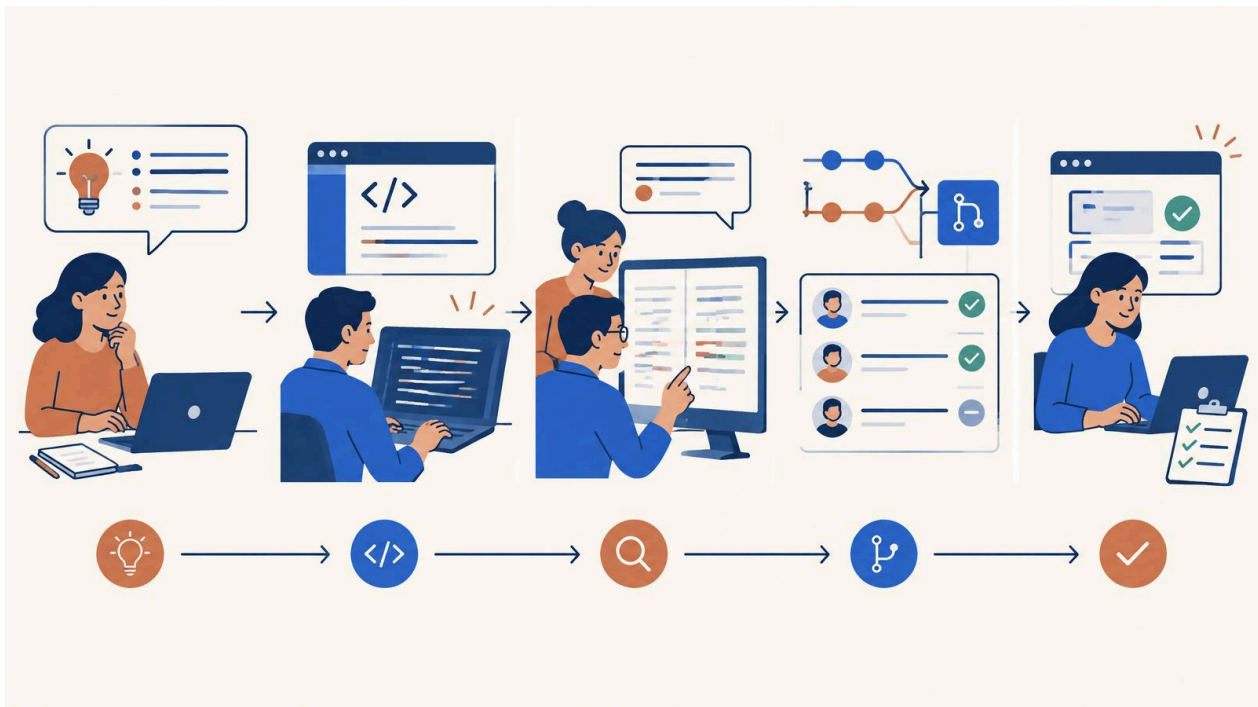


Figure 3: AI can support the journey from idea to code, but people remain responsible for decisions and review.

9.1 Lessons

1. What AI coding agents are and how they differ from ordinary chat tools.
2. Common ways agents assist with explanations, ideas, code suggestions, and debugging.
3. Why clear instructions and useful context improve an agent's response.
4. Why AI-generated code should be read, questioned, and reviewed by a person.
5. Common risks, including incorrect answers, insecure suggestions, missing context, and overconfidence.
6. The human responsibilities AI does not remove: deciding what to build, defining quality, protecting users, and owning the result.

Learning artifact: A simple AI-agent responsibility checklist.

10 Module 04 · Understanding Development Tools

This module introduces the tools people use to work with software and explains the role each tool plays at a high level.

10.1 Lessons

1. Code editors and where software files are written.
2. The command line as a way to communicate with development tools.
3. Programming languages, runtimes, and packages.
4. Frameworks such as Next.js and why teams use them.
5. Configuration files and why projects need agreed settings.

Learning artifact: A simple development-tools map.

11 Module 05 · UI/UX and Frontend Product Thinking

A user interface is a system for helping users understand, decide, and act. This module teaches the concepts that make an interface useful before asking AI to produce one.

11.1 Lessons

1. Hierarchy, clarity, affordance, consistency, feedback, and accessibility.
2. Translating user journeys into pages, screens, states, and transitions.
3. Design tokens, components, spacing, typography, and UI consistency.
4. Responsive behavior and mobile-first decisions.
5. Forms, validation, loading, errors, and success feedback.
6. Semantic HTML, keyboard access, focus, contrast, and alternative text.
7. Using a design system as the source of truth for AI-generated UI.

Learning artifact: A screen map and interface-state inventory.

12 Module 06 · Databases and Data Modeling



Figure 4: Database concepts include storage, relationships, indexes, validation, and controlled access.

12.1 Lessons

1. What databases remember: records, fields, queries, indexes, and backups.
2. Relational and document databases.
3. MongoDB concepts: databases, collections, documents, ObjectIds, and indexes.
4. Identifying entities, ownership, required fields, optional fields, and timestamps.
5. Database credentials, permissions, backups, and server-only access.
6. CRUD operations, pagination, filtering, and safe errors.

Learning artifact: A conceptual data model and database glossary.

13 Module 07 · APIs and Backend Engineering

An API is a contract between systems. Learners study the language of requests, responses, validation, authentication, authorization, and failure.

13.1 Lessons

1. What an API is and why contracts matter.
2. HTTP methods, status codes, headers, parameters, and JSON.
3. The endpoint sequence: authenticate, authorize, validate, execute, respond.
4. Server actions and route handlers.
5. Schema validation, safe errors, logging, and failure states.
6. Authentication, authorization, sessions, roles, permissions, and ownership.
7. Third-party APIs, keys, OAuth, webhooks, retries, timeouts, and idempotency.

Learning artifact: A conceptual API map and request/response analysis.

14 Module 08 · Git, GitHub, and Version Control

Version control is the memory and collaboration layer of software work. Learners understand how changes are recorded, reviewed, merged, and recovered.

14.1 Lessons

1. Why version control matters.
2. Repositories, staging, commits, branches, remotes, and history.
3. GitHub repositories, READMEs, ignore files, visibility, and collaborators.
4. Meaningful commits and reviewing diffs.
5. Branches, pull requests, checks, and merge strategies.
6. Reverting, restoring, and resolving conflicts.
7. Issues and project tracking.

Learning artifact: An annotated GitHub workflow.

15 Module 09 · Environment Variables and Application Security

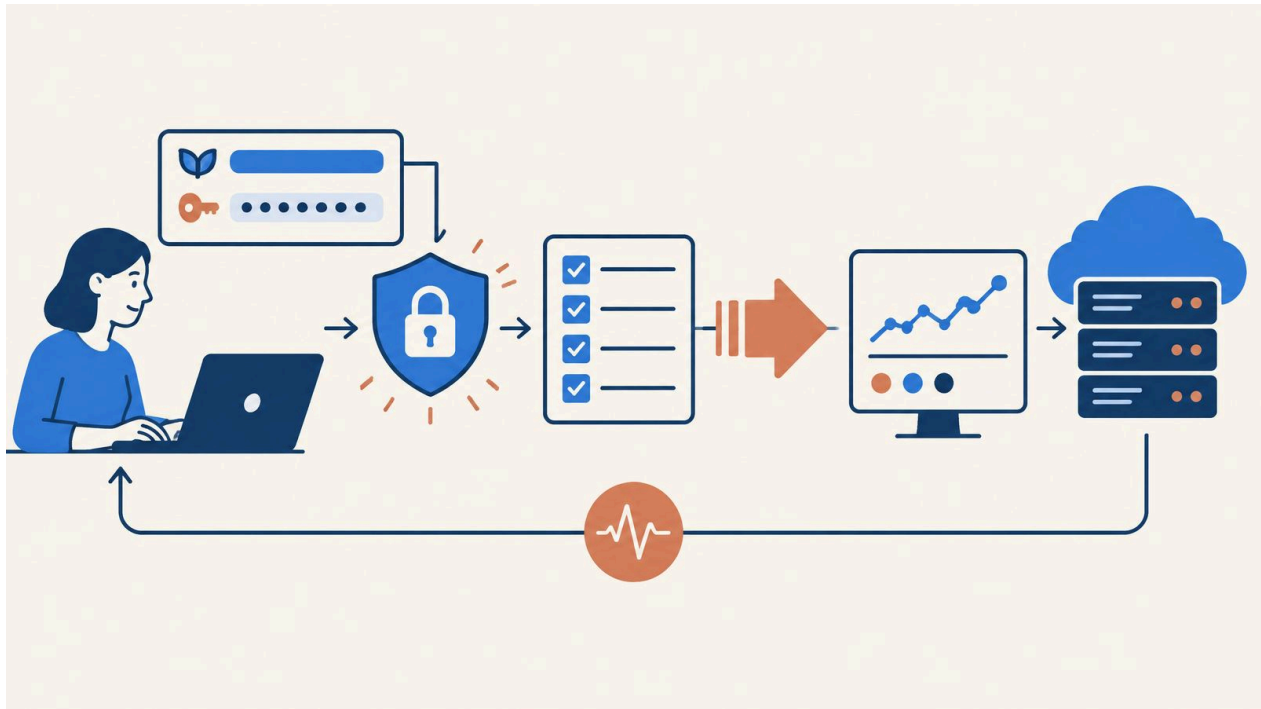


Figure 5: Responsible software delivery connects secrets, security review, testing, deployment, monitoring, and maintenance.

15.1 Lessons

1. Configuration versus code, and public versus private variables.
2. Secrets, keys, passwords, rotation, and accidental exposure.
3. Authentication, authorization, injection, XSS, CSRF, uploads, and dependency risk.
4. Secure input, output encoding, rate limits, and safe errors.
5. Ownership checks and user-data isolation.
6. Payment verification, signatures, idempotency, and audit logs.
7. Evidence-based security review with an AI agent.

Learning artifact: A security review and risk-analysis worksheet.

16 Module 10 · Testing, Debugging, and Quality

Learners study how teams gain confidence that software behaves as intended and how they reason through failure.

16.1 Lessons

1. Confidence, regression prevention, edge cases, and refactoring safety.
2. Manual testing, test data, browser checks, and failure paths.
3. Unit and integration tests.
4. End-to-end user journeys.
5. Reproduction, isolation, logs, network inspection, and controlled fixes.

6. AI-assisted debugging with evidence and root-cause explanations.
7. Definition of done across functionality, design, accessibility, security, tests, and documentation.

Learning artifact: A test plan and definition-of-done checklist.

17 Module 11 · Deployment Concepts and Operating Software

This module explains what happens when software leaves the laptop. It is conceptual and demonstrative; the full deployment exercise belongs to Course Two.

17.1 Lessons

1. Build artifacts, hosting, domains, DNS, environment variables, production databases, logs, and release flow.
2. Local, preview, and production environments.
3. Build scripts, type checks, linting, configuration, and error reporting.
4. How repositories connect to deployment providers and how deployment logs work.
5. Domains, DNS, HTTPS, cookies, and environment-specific URLs.
6. Smoke tests, mobile verification, and checking for exposed secrets.
7. Monitoring, backups, updates, support, and incident response.

Learning artifact: A deployment-readiness checklist and environment comparison.

18 Module 12 · Foundations Review and Course Two Bridge

The final module connects the concepts and prepares learners for the practical full e-commerce build in Course Two.

18.1 Review sequence

1. Explain the product system: connect problem, interface, backend, database, APIs, services, and deployment.
2. Critique an AI-built feature: identify assumptions, risks, and questions before acceptance.
3. Create a build-readiness map: prepare product, architecture, data, API, security, GitHub, and testing maps.
4. Prepare for Course Two: identify gaps and understand the practical e-commerce build ahead.

Learning artifact: A foundations portfolio and Course Two readiness review.

19 Assessment and learner support

The first course assesses understanding and responsible reasoning, not production polish. Learners submit thinking artifacts rather than a completed application.

Assessment area	Weight
Problem clarity and scope	15%

Product and user-flow reasoning	10%
UI/UX reasoning and accessibility awareness	15%
Frontend and backend conceptual understanding	15%
API and database reasoning	15%
GitHub and version-control understanding	10%
Security and environment awareness	15%
Testing and deployment understanding	10%
Quality of final explanation and readiness plan	10%

19.1 Suggested resources

Product brief template, problem-validation interview guide, user journey template, requirements template, AI-agent project-context template, code-review checklist, architecture diagram template, database model template, API design template, GitHub workflow guide, environment-variable checklist, security review checklist, testing checklist, deployment concepts checklist, and Course Two preparation checklist.

20 Course Two relationship

“Course One teaches learners how software works and how to think while vibe coding. Course Two teaches learners how to apply that understanding by building a full e-commerce application from requirements to deployment.”

Course One is complete when learners can explain the major parts of software, ask better questions of AI agents, identify risks, and create a responsible build-readiness plan. Course Two begins when they are ready to apply that foundation in a full practical implementation.

Understand the system. Then build with confidence.

21 Glossary snapshot

Term	Plain-language meaning
AI agent	A tool that can inspect context, plan actions, change files, and help execute a software task.
API	A contract that defines how systems request and return information.
Backend	The server-side logic that validates input, applies rules, and communicates with data or services.
Database	A system for storing and retrieving information that an application needs to remember.

Environment variable	Configuration supplied outside the source code, often used for URLs, settings, or secrets.
GitHub	A platform for storing repositories, reviewing changes, and collaborating on software.
Vibe coding	An AI-assisted approach to creating software through natural-language instructions, iteration, and verification.