Module 4 / Web

Hosting, Domains, SSL & Deployment

BEGINNER STUDY GUIDE

How a site gets from your laptop to the internet: hosting types, domains and DNS, HTTPS, deploying static sites, apps and servers with cPanel.

Use with the 44-slide presentation. Work through the examples at your own pace, pausing for the exercises. Code blocks are labelled with their required file or context. A fragment needs the surrounding project described in the lesson.

Learning outcomes

- Choose between shared hosting, a VPS, platform hosting and static hosting
- Register a domain and point it to a host with DNS records
- Secure a site with an SSL certificate (Let's Encrypt)
- Deploy a static site and a back-end app
- Find your way around cPanel and basic Linux server commands

Before you begin

Complete the portfolio and API modules. Bring a Git repository and understand environment variables.

How to study

Read the definition, trace the example, predict its result, then run it. Change one input and explain what changes. Keep a record of errors and how you resolved them.

Key terms and setup

Term	Meaning
Hosting	A service that runs or serves a website online.
Domain	A readable name such as example.com.
DNS	Records that connect names to services.
HTTPS	HTTP carried over an encrypted TLS connection.
Build	The process that produces deployable files.
Deployment	Releasing a version of the website to its environment.

Setup

1. Choose the site to publish: Use a working HTML portfolio or React project.
2. Check the project locally: Open it, run its build if needed, and fix visible errors.
3. Keep the project in Git: Commit a working version before deployment.
4. Prepare a hosting account: Use a provider that supports the kind of application.
5. Use practice configuration: Treat example.com and sample IP addresses as placeholders.

How a site becomes reachable

Hosting, domains and the journey from a URL to a page

Slide 6: Local development and the live website

On your laptop

- The files and dev server are local.

- localhost refers to this device.

- Other people cannot use your local file path.

On the web

- A host serves files or runs the app.

- Visitors use a public URL.

- The live environment needs its own configuration.

Sending a friend C:\projects\portfolio\index.html does not publish your website. That path refers to your own computer. A live site needs a service that responds to web requests and a URL that reaches it.

Key point: Deployment copies or builds a release on a reachable host.

Slide 7: The website delivery chain

Stage	What happens	What can go wrong
Domain lookup	DNS finds a service address	Wrong or missing DNS record.
Connection	Browser contacts the host	Wrong port or unreachable service.
TLS	HTTPS identity and encryption are negotiated	Certificate or hostname problem.
HTTP request	Host receives the path	Unknown route or missing file.
Response	Browser loads HTML, CSS and JavaScript	Broken asset path or runtime error.

A site can have working DNS but a broken application. It can also have a healthy application that visitors cannot reach because the domain points elsewhere. Isolate one layer at a time using the browser error, hosting logs and DNS lookup results.

Key point: Check the failing stage before changing unrelated settings.

Slide 8: A domain and hosting are different services

Domain registration

- Reserves the name for a period.

- Provides control of its DNS settings.

- Needs renewal and account protection.

Website hosting

- Stores the published files or runs the server.

- Provides deployment and runtime settings.

- May give you a default hosting URL.

A provider URL is useful for testing before connecting a custom domain. Keep domain account access with the rightful owner and document which account controls registration, DNS, hosting and billing.

Key point: One company can sell both, but changing one does not automatically change the other.

Slide 9: Static sites and server applications

Static hosting

- Serves prepared HTML, CSS, JS and assets.

- Suitable for an HTML site or a built React client.

- Does not run a private Node API by itself.

Application hosting

- Runs a backend process.

- Needs runtime configuration and logs.

- May connect to a separate database.

Static describes how the host delivers files, not whether JavaScript can be interactive. A static frontend can call a remote API. Private database credentials belong with the backend service, never in the browser build.

Key point: A browser React app and a Node API may be deployed as separate services.

Slide 10: cPanel and the hosting account

What cPanel can provide

- A dashboard for a hosting account.

- File, domain, email and database tools.

- Feature availability depends on the host.

What students should distinguish

- The control panel is not the website content.

- The document root is a serving directory.

- A running Node app needs supported runtime hosting.

The curriculum includes cPanel because shared-hosting workflows remain common. Use the host's documentation and account permissions. Do not guess root access or install system services from an ordinary shared-hosting account.

Key point: Do not assume every cPanel plan can run every backend framework.

Slide 11: A basic cPanel publishing workflow

Step	Action	Check
1	Add the intended domain or subdomain	The account is authorised to use it.
2	Identify its document root	Use the path shown by the hosting account.
3	Upload or deploy the correct public files	The entry file is in the served location.
4	Keep private configuration outside public access	No .env or database export can be downloaded.
5	Check the public URL and HTTPS	Assets and intended routes work.

A subdomain can use another directory. Avoid uploading an entire development folder containing secrets, dependency caches or backups into a public document root. Use supported application deployment tools for a backend process.

Key point: public_html is common for a primary site, but use the account's actual document-root setting.

Slide 12: Server basics: processes, logs and access

A running application

- A process listens for requests.

- A service manager can restart it.

- Logs help explain failures.

Access and filesystem

- Use approved SSH or panel access.

- Keep permissions limited to what is needed.

- Separate public files from private configuration.

On managed hosting, many server responsibilities belong to the provider. Learn which settings the account actually controls. Read logs with credentials and personal data protected, and keep production changes documented.

Key point: Do not fix an unfamiliar permission error by making every file world-writable.

Slide 13: Choose hosting by requirements

Question	What to find out	Why it matters
What is deployed?	Static files or a running app	Determines supported hosting type.
Which runtime?	Node, Python, PHP or another	The platform must support it.
Where is stored data?	Database and uploaded files	Needs persistence and backups.
Which settings?	Build and environment variables	The release must be reproducible.
Who operates it?	Owner, logs and renewal access	Someone must maintain the site.

Avoid choosing only from a screenshot of a hosting dashboard. A suitable service must support the application's actual runtime, traffic, data and operational needs. Keep costs and ownership visible to the course project owner.

Key point: Check current provider documentation for setup steps and service limits.

Slide 14: Practice: classify a project

Try it first

- Project A contains HTML, CSS and images.

- Project B is a React client with an external API.

- Project C runs Express and writes to PostgreSQL.

Check your work

- A can use static hosting.

- B needs a frontend build plus the API service.

- C needs application hosting and a database.

Some platforms provide several of these services in one account. The architectural distinction still matters for configuration, secrets, deployment and troubleshooting.

Key point: Draw the frontend, API and database as separate responsibilities.

DNS and HTTPS

Point the name at the correct service and protect the connection

Slide 16: Read the DNS record types

Type	Purpose	Example use
A	Maps a name to an IPv4 address	The host's supplied IPv4 target.
AAAA	Maps a name to an IPv6 address	The host's supplied IPv6 target.
CNAME	Aliases a name to another DNS name	www points to a provider hostname.
TXT	Stores text used by services	Domain verification or email policy.
MX	Directs mail to mail servers	Inbound email for the domain.

DNS records connect names to different services. A website A record does not configure mailbox hosting. Some providers support special root-domain alias behaviour, so follow their documentation for the apex instead of assuming every root can use a standard CNAME.

Key point: Use the exact records supplied for your service; sample values are not deployment instructions.

Slide 17: The root domain and www are different names

Root or apex

- example.com is the main registered name.
- It needs supported DNS configuration.
- It can be the preferred public address.

A subdomain

- www.example.com is another DNS name.
- api.example.com can reach the API.
- Each name needs correct routing and HTTPS.

Both addresses can reach the same website, but they are still different hostnames. Configure the hosting service to recognise each intended hostname. The certificate must also cover the name visitors actually use.

Key point: Choose one preferred website address and redirect the other consistently.

Slide 18: Example DNS change plan

Setting	Placeholder example	Check before saving
Website A	example.com -> 192.0.2.10	Replace with the host's supplied target.
Website alias	www -> site.provider.example	Confirm the provider's exact hostname.
Verification TXT	A service-issued verification value	Copy the full value exactly.
Existing email	Keep required MX and TXT records	Do not erase working mail configuration.

Before editing DNS, capture the current records and identify which service each record supports. A website migration can accidentally break email when unrelated records are deleted. Plan the change and retain a way to restore the previous configuration.

Key point: These reserved example values must not be copied into a real domain.

Slide 19: DNS changes are cached

What caching means

- Resolvers can reuse an earlier answer.
- A TTL influences how long an answer may be cached.
- Two devices may briefly see different answers.

How to check

- Confirm the authoritative record first.
- Use a lookup tool and compare results.
- Wait for relevant caches instead of editing repeatedly.

Repeatedly switching targets can make diagnosis harder while caches hold different values. Confirm the intended target, the nameservers controlling the domain and the hosting configuration before assuming a delay is the only problem.

Key point: A DNS change is not guaranteed to appear everywhere at the same moment.

Slide 20: Check name resolution from your computer

Context: Windows terminal

```
nslookup example.com
nslookup -type=CNAME www.example.com
nslookup -type=MX example.com
```

Part	What it means
nslookup	Queries DNS information.
Default lookup	Shows address information for the name.
-type=CNAME	Checks a name alias.
-type=MX	Checks mail-routing records.

A lookup shows what the selected resolver returns. It does not prove the web application is healthy. If DNS resolves correctly, continue to HTTPS and the HTTP response instead of changing the record again.

Key point: Replace example.com with a domain you are authorised to inspect or use.

Slide 21: What HTTPS protects

TLS provides

- Encryption for data in transit.
- Integrity against transport modification.
- Authentication of the server hostname.

HTTPS does not guarantee

- That all website content is trustworthy.
- That the app has no security bugs.
- That stolen credentials cannot be misused.

People often call this SSL, but modern deployments use TLS. Use the hosting provider's supported certificate flow. Test the exact hostname and avoid teaching students to click through certificate warnings as a normal solution.

Key point: A valid certificate is necessary, but it is only one part of a secure site.

Slide 22: HTTPS checks after connecting a domain

Check	Expected behaviour	Failure clue
Certificate name	Covers the visited hostname	Hostname mismatch warning.
Certificate lifetime	Valid and renewable	Expired certificate warning.
HTTP redirect	Leads to the preferred HTTPS URL	Visitors remain on unencrypted HTTP.
Assets	Load over HTTPS	Mixed-content warnings.
Alternate hostname	Redirects consistently	Duplicate or broken www address.

A certificate may fail to issue or renew if DNS or validation routing is wrong. Check the provider status and logs rather than repeatedly requesting new certificates. Update hard-coded http asset URLs when they cause mixed content.

Key point: Automated renewal still needs monitoring and correct domain configuration.

Slide 23: Practice: explain a certificate warning

Try it first

- The site opens on a provider URL.

- The custom domain shows a hostname warning.

- DNS points to the expected service.

Check your work

- Confirm the domain is added to the host.

- Check which names the certificate covers.

- Complete the host's certificate setup.

DNS pointing at a service does not always tell that service to serve the new hostname. Domain binding and certificate issuance are separate configuration steps on many platforms.

Key point: Do not disable browser certificate checks to make the warning disappear.

Build and release the application

Publish the right files with the right configuration

Slide 25: Source files and build output

Source project

- Contains editable code and dependencies.

- Uses development scripts and settings.

- Belongs in version control.

Build output

- Contains deployable frontend files.

- Usually generated into dist for Vite.

- Can be recreated from source and configuration.

An HTML-only project may not need a build step. A React/Vite project usually does. Publishing src alone will not make the browser interpret every source format correctly. Confirm the build command and output directory for the actual tool.

Key point: Deploy the folder your build actually produces.

Slide 26: Build a React/Vite frontend

Context: Terminal: project folder

```
npm ci
npm run build
npm run preview
```

Part	What it means
npm ci	Installs from the committed lockfile.
build	Creates the production output.
preview	Serves that output for a local check.
dist	Vite's usual output folder unless configured otherwise.

npm ci expects a compatible package-lock.json. Run the local preview to catch missing assets and configuration differences. A preview server is a checking tool, not an instruction to use the development machine as permanent hosting.

Key point: Fix a failed build before trying to deploy the same broken source.

Slide 27: Basic deployment settings

Setting	Example	What to verify
Repository	Your portfolio repository	Correct owner and access.
Branch	main	The branch intended for release.
Build command	npm run build	Works from a clean dependency install.
Output directory	dist	Matches the build output.
Environment	Public API URL	Correct target for this environment.

Different providers name these settings differently, but the underlying questions are the same. Save the settings in the README without exposing secrets. A deployment should identify the source commit it used.

Key point: Use the provider's current instructions for the chosen project type.

Slide 28: Frontend variables are visible to visitors

Safe public configuration

- A public API base URL.

- A non-secret analytics identifier.

- Feature settings intended for the browser.

Private configuration

- Database passwords.

- Private API signing keys.

- Server-only authentication secrets.

A variable named SECRET is not protected merely because it came from a .env file. The frontend build can replace variables with literal values. Keep secret-dependent operations in an authenticated backend.

Key point: Anything bundled into browser JavaScript can be inspected by visitors.

Slide 29: Deploy a backend as a running service

Configure the process

- Choose a supported runtime.

- Provide the documented start command.

- Use the port assigned by the host.

Configure its dependencies

- Set server-side secrets securely.

- Connect the right database.

- Use persistent storage for durable uploads.

Do not assume a file written beside server.js will remain forever on hosted infrastructure. Use the host's documented persistent volume or object storage design. Keep backups and migration steps separate from ordinary process startup.

Key point: An ephemeral application filesystem may be replaced during deployment or restart.

Slide 30: Development, staging and production

Environment	Purpose	Data rule
Development	Build and debug on your machine	Use disposable practice data.
Staging	Rehearse a release	Use safe data and test integrations.
Production	Serve real users	Protect real data and access.

Use distinct settings and clear naming so students know where requests go. A test payment or email service should stay in its documented test mode. Deployment permissions should be limited to the people who need them.

Key point: Do not point a practice script at a production database by accident.

Slide 31: A release needs a repeatable sequence

Before publishing

- Commit the intended source.

- Run the build and relevant checks.

- Prepare migrations and configuration.

After publishing

- Check health and important user paths.

- Inspect logs for new failures.

- Keep the previous release available for rollback.

Record the commit, deployment time and environment. If database changes are involved, consider compatibility with both the old and new app versions. A rollback of files alone may not reverse a destructive data change.

Key point: A successful upload is not the same as a working user experience.

Slide 32: Use a health endpoint after release

Context: Express backend

```
app.get("/health", (req, res) => {
  res.json({ status: "ok" });
});
```

Part	What it means
Purpose	Confirms that this route and process respond.
200 response	A basic successful health signal.
Scope	Does not prove every dependency is healthy.
Privacy	Keep internal configuration out of the response.

The simple route is a liveness check. A deeper readiness check may inspect essential dependencies with careful timeouts. Do not expose database credentials, private hostnames or detailed error traces in a public health response.

Key point: A health check and a real user-path test provide different evidence.

Slide 33: Test the published site

Area	Action	Expected result
Main page	Open the final public URL	Correct title and content.
Assets	Load CSS, images and scripts	No missing-file errors.
Routes	Refresh a nested route directly	The correct screen opens.
Forms and API	Run one successful and failed request	Useful feedback and correct target.
Mobile view	Use a small real viewport	Readable layout without sideways scrolling.
HTTPS	Visit both preferred and alternate names	Valid certificate and correct redirect.

A static host for a client-side router may need a fallback rule to index.html. A blanket fallback must not accidentally turn missing API routes into HTML responses. Test both page routes and API endpoints separately.

Key point: Test the deployed URL, not only the local preview.

Slide 34: Rollback and backups solve different problems

Rollback

- Restores an earlier application release.
- Helps recover from a bad code deployment.
- May depend on schema compatibility.

Backup and restore

- Recovers stored data or configuration.
- Needs a tested restore procedure.
- Must match a useful recovery point.

A backup that has never been restored gives limited confidence. Document who can initiate recovery and where the protected backups live. A rollback can fail if the old app cannot read the new schema, so plan migrations carefully.

Key point: Keep a known working release and a tested data-recovery plan.

Slide 35: Troubleshoot deployment by layer

Symptom	Likely layer	First check
Domain does not resolve	DNS	Nameservers and records.
Certificate warning	TLS or domain binding	Hostname and certificate status.
Homepage works, assets fail	Build paths	Network responses for CSS and JS.
Nested route refresh gives 404	Static route fallback	Host routing configuration.
API fails only online	Runtime or environment	Service logs and public API URL.

A screenshot of a blank page is not enough diagnosis. Inspect the URL, status, Console, Network tab and provider logs to identify the responsible layer.

Key point: Make one justified change at a time and record its result.

Slide 36: Final practice: publish and verify your portfolio

Try it first

- Deploy a committed working version.
- Check HTTPS and any chosen domain.
- Test assets, nested routes and a key interaction.

Check your work

- A classmate can open the public URL.
- Your README records the release settings.
- You can explain how to restore the previous version.

Submit the repository commit, public URL and a short test checklist. If a custom domain is unavailable, the provider's public URL is suitable for demonstrating the deployment workflow.

Key point: Use placeholder DNS examples only as learning illustrations.

Guided practice and assessment

Build, test, explain and submit your work

Slide 38: Guided lab: first working version

1. Step 1: Deploy the portfolio and API to a staging environment.

2. Step 2: Record website DNS changes while retaining existing mail records.

3. Step 3: Configure server-only secrets and the public API base URL separately.

Key point: Pause after each step. Repeat a failed step before adding the next feature.

Slide 39: Guided lab: complete and verify

1. Step 4: Test HTTPS, a direct client route and an authenticated operation.

2. Step 5: Write a release and rollback checklist, including database compatibility.

Key point: Repeat the task from a clean start using your README.

Slide 40: Test the successful and unsuccessful paths

Expected behaviour

- The direct nested URL loads after a browser refresh.

- API logs show no secrets and the service restarts after a controlled restart.

Failure and recovery

- The browser makes no mixed HTTP requests from an HTTPS page.

- A previous compatible release can be restored through the documented process.

A feature is ready when it works for ordinary input and handles the relevant failure cases. Explain what you tested, what happened and how another person can repeat the same test.

Key point: Keep screenshots, request results or a short demonstration as evidence.

Slide 41: Assignment: demonstrate what you can build

1. Deploy your React portfolio on Netlify or Vercel with a custom subdomain

2. Deploy your Back End API on a platform such as Render, with environment variables

3. Add the DNS records and confirm HTTPS works on both

4. Write a deployment checklist in your README

Rubric: functionality 40; robustness 25; code and setup 20; evidence and README 15.

Submit the implementation and the evidence requested in the class materials. A README must state prerequisites, setup steps, how to run the project and how to repeat the acceptance checks. The instructor evaluates the practical submission separately from the Odoo quiz.

Slide 42: Review: explain the ideas in your own words

Question	What a strong answer explains
Why can static hosting serve React but not this Express API?	The built React assets are files. Express requires a running server process.
Why might a DNS edit appear ineffective?	The wrong zone may have been edited or recursive resolvers may still hold cached records until expiry.
What does a 502 suggest?	A gateway could not obtain a valid upstream response. Inspect the API process, port and proxy configuration.

Use your own project as evidence. Point to the specific code, configuration or interface that demonstrates your answer. If you cannot yet explain a term, return to the glossary and its worked example.

Key point: Try answering aloud before reading the explanation.

Slide 43: How to complete the Odoo quiz

Before attempting it

- Complete the guided lab.
- Review the definitions and worked examples.
- Answer every question with one selected choice.

When checking your result

- Read the feedback for incorrect answers.
- Return to the relevant lesson section.
- Retry after correcting your understanding.

A submitted wrong answer should provide feedback and leave the quiz incomplete. A completely correct submission should mark the quiz complete. This lesson contains single-answer questions; selecting more than one choice is not required.

Key point: The class quiz is marked by Odoo; the practical assignment is reviewed separately.

Slide 44: Official documentation and further practice

Resource	Use it to study
MDN: What is a domain name?	https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/What_is_a_domain_name
Cloudflare Learning: DNS	https://www.cloudflare.com/learning/dns/what-is-dns/
Let's Encrypt: How it works	https://letsencrypt.org/how-it-works/
cPanel documentation	https://docs.cpanel.net
DigitalOcean tutorials	https://www.digitalocean.com/community/tutorials

Read the relevant page while repeating the example, then change one small input and predict the result. Documentation links are also listed in the study guide and the speaker notes.

Key point: Use documentation for the version installed in your project.

Class quiz

Select one answer for each question. Try all 12 before checking the answer key. The same class quiz is available in Odoo with submission feedback.

1. Which DNS record points a domain to a server's IPv4 address?

A. A record

B. CNAME to an email server

C. TXT record

D. MX record

2. What does an SSL certificate give your site?

A. More storage

B. A free domain

C. Faster DNS

D. HTTPS: encrypted traffic and the padlock in the browser

3. Which hosting suits a WordPress site for a small business on a budget?

A. Shared hosting with cPanel

B. A laptop at home

C. GitHub Pages

D. A Kubernetes cluster

4. You changed a DNS record but some visitors still see the old site. Why?

A. Browsers block new records

B. The SSL certificate expired

C. The site is broken

D. DNS propagation: caches keep the old record until the TTL expires

5. Which command follows a server log file live?

A. ls app.log

B. rm app.log

C. tail -f app.log

D. cat -live app.log

6. Where should production secrets for a deployed app be set?

A. In the HTML

B. In the host's environment variable settings

C. In the Git repository

D. In the React bundle

7. A nested React URL works through a link but fails after refresh. What is likely missing?

A. A new database password

B. A second Git branch

C. An MX record

D. A host fallback to the application entry point

8. The server deploys correctly but disappears after reboot. What is needed?

A. A longer domain name

B. A CSS reset

C. A sitemap

D. A configured service supervisor with startup enabled

9. Which deployment target is suitable for a built React client that uses a separately hosted API?

A. A DNS TXT record containing all JavaScript

B. Static hosting for the built frontend files

C. A local file path sent to visitors

D. A database password inside every page

10. What is the difference between a domain and hosting?

A. A domain contains the entire database

B. The domain is a name; hosting serves files or runs the app

C. Hosting only changes font sizes

D. They are exactly the same service in all cases

11. A nested React route works after clicking a link but returns 404 after refresh. What should you inspect?

A. The database column type

B. The static host's client-route fallback configuration

C. The image alt text only

D. The user's password length

12. Which value is safe to treat as public browser configuration?

A. A private email-service API key

B. A public API base URL

C. A JWT signing key

D. A private database password

Quiz answers and explanations

1. Which DNS record points a domain to a server's IPv4 address?

A record

A maps a name to an IPv4 address; CNAME points one name to another name; MX is for email.

2. What does an SSL certificate give your site?

HTTPS: encrypted traffic and the padlock in the browser

Let's Encrypt and cPanel AutoSSL provide free certificates.

3. Which hosting suits a WordPress site for a small business on a budget?

Shared hosting with cPanel

Shared hosting with cPanel runs PHP and MySQL cheaply and simply.

4. You changed a DNS record but some visitors still see the old site. Why?

DNS propagation: caches keep the old record until the TTL expires

Lower the TTL before planned changes.

5. Which command follows a server log file live?

tail -f app.log

tail -f keeps printing new lines as they are written.

6. Where should production secrets for a deployed app be set?

In the host's environment variable settings

Every platform and server has a place for environment variables.

7. A nested React URL works through a link but fails after refresh. What is likely missing?

A host fallback to the application entry point

The host must route client-side paths to the SPA entry point.

8. The server deploys correctly but disappears after reboot. What is needed?

A configured service supervisor with startup enabled

The process must start and restart independently of an interactive shell.

9. Which deployment target is suitable for a built React client that uses a separately hosted API?

Static hosting for the built frontend files

The built client is a set of static assets. Its API is a separate running service, even if both are managed by one provider.

10. What is the difference between a domain and hosting?

The domain is a name; hosting serves files or runs the app

A domain and hosting can be purchased from the same company, but they perform different roles. DNS and host configuration connect the intended name to the service.

11. A nested React route works after clicking a link but returns 404 after refresh. What should you inspect?

The static host's client-route fallback configuration

A direct URL request reaches the host before the client router runs. Configure the documented fallback for page routes and keep API routes distinct.

12. Which value is safe to treat as public browser configuration?

A public API base URL

The browser bundle can be inspected. Public endpoint configuration can be included, but private credentials must remain in an appropriate server environment.

Official references

MDN: What is a domain name?

https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/What_is_a_domain_name

Cloudflare Learning: DNS

https://www.cloudflare.com/learning/dns/what-is-dns/

Let's Encrypt: How it works

https://letsencrypt.org/how-it-works/

cPanel documentation

https://docs.cpanel.net

DigitalOcean tutorials

https://www.digitalocean.com/community/tutorials

cPanel documentation

https://docs.cpanel.net/cpanel/

Curriculum alignment

This lesson develops the supplied Software Development Curriculum within its existing five-chapter, three-class structure. The examples use JavaScript/React, Express/PostgreSQL and Flutter for consistent practice. Alternative frameworks are identified where relevant.