Co-funded by the European Union

Project Nr. 2022-1-FR01-KA220-HED-000086863

lightcode

Strengthening the Digital Transformation of Higher Education Through Low-Code

7. Web and Mobile Apps with LowCode

Paris-Dauphine University — PSL

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Erasmus+ Project
lightcode

Co-funded by the European Union

PROJECT'S COORDINATOR

Paris Dauphine University
France

PROJECT'S PARTNERS

University of Macedonia Greece

University of Niš
Serbia

Karmic Software Research
France

REACH Innovation
Austria

University of Zagreb
Faculty of Economics & Business

University of Zagreb
Croatia

Symplexis
Greece

symplexis.eu

TABLE OF CONTENTS

OVERVIEW

Introduction

Whether you're checking your emails, ordering a pizza, or finding your way with GPS, you're using an app. But did you know that not all apps are the same? Some are installed directly on your phone. Others run in a browser. Some load instantly, while others stop working if your internet connection fails.

This chapter is your gateway to understanding how different kinds of apps behave—and what this means for you as a builder. You won't start with code or complex theories. Instead, we'll explore familiar apps from your everyday life and uncover how they work behind the scenes.

By the end of this module, you'll know how to recognize different types of apps, identify the visual building blocks they use (called widgets), and understand how low-code tools like LightCode make it possible to build useful applications without needing to master every underlying technology.

Why is this Module Important?

Before you can build apps with confidence, you need to understand what you're actually building—and what your users will expect.

This module gives you a mental map of:

- What kind of apps exist (native, web, mobile),
- What makes them work—or break,
- And how basic elements like buttons, lists, and forms interact with data.

You'll see that while real-world app development involves a deep tech stack (like HTML, CSS, JavaScript, APIs, and databases), low-code platforms hide much of that complexity. That's a good thing—it means you can focus on logic, design, and user interaction. But it also means making some built-in assumptions, which you'll start learning to spot.

This chapter is also the **bridge** to Chapter 8, where you'll begin working not just with simple widgets, but with higher-level interactive objects that behave more like "things" in the real world—such as orders, surveys, or documents.

Learning Outcomes

By the end of this module, you will be able to:

- Distinguish between native apps, web apps, and mobile-optimized apps using real-world examples.
- Identify and describe common UI widgets (e.g., checkbox, text input, picklist, button).
- Match basic widgets to their natural data types (e.g., checkbox → boolean).
- Explain the role of low-code platforms in simplifying app development.
- Begin forming mental models of how app components interact: browser vs server, temporary vs persistent memory.
- Prepare to use higher-level interactive objects introduced in Chapter 8.

Prerequisites

This module is designed for beginners. There is no technical experience required.

However, to fully enjoy the journey, you should be:

- Comfortable using everyday digital tools (e.g., email, web browser, messaging apps).
- Curious about how things work behind the screen.
- Ready to explore how digital elements connect and interact, without needing to write complex code.

Think of it this way: if you know how to order a pizza online, you already have everything you need to start building your own apps.

STRUCTURE OF THE MODULE

This chapter unfolds progressively, starting from everyday digital experiences and guiding you toward the deeper logic that powers modern apps. You'll move from observation to understanding—without needing prior technical expertise.

The journey is divided into three parts:

1. From Everyday Apps to Digital Thinking

You'll begin by looking at the kinds of apps you already use—native apps, web apps, and mobile-optimized apps. Through real-world examples like email, food delivery, and GPS, you'll learn to recognize how these apps behave differently and why that matters.

Then, you'll explore the basic visual building blocks shared across all apps: **widgets**. From buttons to picklists and text inputs, you'll discover how these elements work, what they're meant to capture, and how low-code platforms like LightCode simplify their setup.

You'll finish this section by linking widgets to their **natural data types**, setting the stage for more advanced logic to come.

2. What Makes Apps Work (or Not)

Next, you'll shift your perspective from screens to systems.

Through simple riddles, live exercises, and thought experiments, you'll investigate:

- What happens when you lose connection?
- Why some changes disappear after refresh?
- Where your app actually "lives"—in your browser, on a server, or in the cloud?

You'll explore the **invisible architecture** of apps: memory vs storage, local vs server, temporary vs permanent data. This gives you the tools to understand why things sometimes work… and sometimes don't.

3. Widgets, Objects, and Preparing for Interactions

In the final section, you'll look beyond individual widgets and start thinking in **objects**—like Order, Task, or Survey.

You'll see how these objects:

- Combine multiple fields
- Evolve over time with states
- React to user input through actions and methods

DIGITAL APPS AND LOWCODE, STEP BY STEP

1. From Everyday Apps to Digital Thinking

Imagine it's a typical day. You check your email on your laptop, send a message to a friend using WhatsApp on your phone, then order a pizza from a food delivery app before heading out. Later, you open Google Maps to navigate across town.

You just used at least four different apps—but they didn't all work the same way.

Let's look at them more closely:

- **Gmail (in browser):** You didn't install it. It ran in a web browser.
- **WhatsApp (mobile app):** You downloaded and installed it from the App Store.
- **Pizza Delivery Site:** It might be a web app or a mobile-friendly website.
- **Google Maps:** Could be an installed app—or opened via browser on your phone.

💡 *Some apps live in your browser, some live on your phone, and some do both. For instance, in LightCode, you will be building web apps—apps that run in browsers, and adapt to both desktop and mobile screens.*

🧭 Types of Apps You Already Use

Type	Example	How You Use It	Needs Installation?
Native App	WhatsApp, Instagram	Downloaded and installed on your device	✅ Yes
Web App	Gmail (in browser)	Runs in Chrome, Safari, Firefox…	❌ No
Mobile Web	Uber Eats (mobile site)	Optimized for phone screens in browser	❌ No

These apps may look similar, but they work differently under the hood. Some store information on your phone. Others rely on live internet connections. Some have advanced access to your device (like camera or GPS), while others don't.

As an app builder, **you don't need to master the deep technologies behind all this—but you do need to know how your app behaves** depending on where and how it runs.

🧱 All Apps Are Built from Widgets

Despite their differences, all apps—from Gmail to games to delivery apps—are made of similar **building blocks**, also called **widgets**.

Let's look at some:

Widget	Purpose	Where You've Seen It
Text Input	Type your name, message, or search query	Search bars, login screens
Checkbox	Yes/No decisions	"I agree to terms", "Subscribe to newsletter"
Picklist	Choose one from many options	Country selector, pizza toppings
Button	Trigger an action	"Submit", "Buy now", "Next"
Slider	Adjust a value	Brightness, sound, star ratings
Image/Video	Display media	Logos, profile pictures, maps
Map	Show locations	Google Maps, delivery tracking
Grid/List	Show many items in a layout	Inbox, product listings, playlist views

These are the same building blocks you'll use in LightCode. But instead of writing code for each widget, you'll drag and configure them visually—**like using digital LEGO**.

🎨 Why Low-Code Makes It Easier (and a Bit Different)

Under the surface, every widget is powered by **real code**—HTML, CSS, JavaScript, React, databases, APIs, and more.

But in LightCode, these technologies are **hidden away**.

- You don’t write the code.
- You **choose** the widget, **configure** it, and **connect** it to behavior.

This makes your app faster to build—but also means that LightCode makes **design decisions for you**. For example:

- The layout is already mobile-responsive.
- The picklist may behave a certain way (like showing default values).
- You can’t always modify every little thing (unless you use advanced settings).

💡 *It’s easier—but opinionated. And that’s a good thing, especially while learning.*

🧠 From Widgets to Meaning: Prepare for Logic

Now comes the smart part. You’ll soon learn that **widgets don’t just look nice—they carry information**. Each widget is linked to a **data type**, which describes what kind of value it holds or action it triggers.

Here's how they connect:

Widget	Data Type (meaning)	Example
Checkbox	Boolean (True/False)	"Remember me?" → checked or not
Picklist	List (options)	"Select a language"
Text Input	String (text)	"Name", "Email"
Slider	Number	"Rate from 1–10"
Button	Behavior (Method)	"Submit" → triggers an action
Table/Grid	List of objects	A list of participants, tasks, etc.

🧩 You'll soon be matching widgets to data types *naturally*—because that's how LightCode works. If your object has a boolean field, it will most likely show up as a checkbox. If you define a list of options, it might appear as a dropdown or button group.

And here's the exciting part:

*In **Chapter 8**, we move beyond these widgets. You'll begin using **interactive objects**—like Order, Beverage, Survey, and more—that include multiple attributes, states, and behaviors. You won't just build forms. You'll build **smart processes**.*

But first, let's understand where all this information goes—into your browser? The internet? The cloud?

Next up: we uncover how apps run behind the scenes—what works offline, what doesn't, and how to debug a disappearing form.

Excellent. We now move to the next section of Chapter 7 — the previously outlined **"big components behind the screen"**. This is **Section 2: What Makes Apps Work (or Not)**, following your outside-in philosophy and using experience-first examples, riddles, and observations.

2. What Makes Apps Work (or Not)

You've seen what apps are made of: screens, buttons, picklists, and more. But now comes a different question:

- "When I click, where does it go?"
- "Why does it sometimes remember things—and sometimes forget?"
- "What breaks when I'm offline?"

Let's investigate.

🔍 When Things Disappear

Imagine Emma is building a quiz in LightCode. She types in questions, previews the app, and is happy with the result. She closes the browser. The next day, it's gone.

What happened?

🧩 Riddle:

"You wrote it, you saw it, but you didn't press save. I'm fast but forgetful. I vanish when you leave. What am I?"

→ **Browser RAM / Temporary Memory**

💡 *Your browser stores things in its short-term memory—like writing on a whiteboard. If you don't hit "save", it disappears.*

📡 Offline Mode: What Still Works?

Now Emma's phone loses internet. She opens her LightCode app anyway. It shows her layout but not the survey questions.

Why?

Try this yourself:

1. Open an app while online.
2. Disconnect Wi-Fi.
3. Reload or use the app.

You'll find:

- The layout may still load (cached).
- Dynamic data (like a live list of answers) will not.
- Submit buttons might do nothing.

📦 Teaching Moment:

Apps rely on:

- **Local files** (cached layout & content)

- **Live data** (comes from the internet/server)
- **User actions** (stored *somewhere*)

Without internet, only some of these survive.

🧠 Where Stuff Lives

To keep it simple, let's split the app into **three zones**:

Zone	Lives in…	What it's good for
Browser (Local)	RAM, sessionStorage, localStorage	Fast, temporary display of your app
Server (Remote)	Memory, sessions, databases	Shared data, persistence, multiple users
Third-Party (Cloud)	Google Sheets, Dropbox, Firebase	Special-purpose storage (needs permissions & APIs)

🔁 *Think of it like this:*

- RAM = your working table (fast, but cleared when you leave).
- Disk = your notebook (keeps info even when you shut down).
- Server = your library in another building (slower but shared).
- Third-party = someone else's library you borrow from (with rules).

🧪 Mini-Investigation: Where Did It Go?

Let's try this game: Emma updates a form title. Her friend doesn't see the change. What could have gone wrong?

1. Was it **saved in the browser** only?
2. Did it **sync to the server**?
3. Was the **session expired** before saving?
4. Did the app lose **connection to the cloud file**?

Each of these layers matters.

🧠 Three Invisible Factors to Watch

🧱 1. Memory vs Storage

- **RAM** is fast but forgetful (e.g., edits before saving).
- **Storage** is permanent (e.g., saved YAML or published blocks).

🌍 2. Local vs Server

- **Local data** stays on your device (e.g., draft mode).
- **Server data** is shared across users (e.g., published quiz).

🔐 3. Sessions & Tokens

- Servers often use **sessions** (temporary identity).
- If your session expires, your actions may be lost.

💡 *Ever been logged out unexpectedly? That's a session timeout.*

🧰 Debug Tip: Use the "Where Am I?" Rule

When something breaks, ask:

- Did I save?
- Was I online?
- Is this shared with others—or just visible to me?
- Am I editing a real version or a preview?

These questions turn you from a clicker into a builder.

🎓 Wrap-Up

Before you move on to building advanced logic, it's essential to know:

- **Where your app runs**
- **Where your data lives**
- **Why things sometimes vanish**

In Chapter 8, you'll work with smarter interactive objects—like Orders, Payments, and Surveys. These objects need memory, structure, and behaviors. You're now ready to understand what powers them.

Great — let's continue with the final segment of Chapter 7.

3. Widgets, Objects, and Preparing for Interactions

So far, you've seen the everyday building blocks of apps—text fields, checkboxes, buttons—and you've discovered where your data goes, how it moves, and why things sometimes don't work as expected.

But LightCode doesn't stop at widgets. In fact, widgets are only the beginning.

🧩 From Widgets to Objects

Let's look again at these examples:

- A **text input** might capture someone's *name*.
- A **picklist** lets you choose a *beverage*.
- A **button** might submit an *order*.

But soon you'll stop thinking in isolated widgets. You'll think in **objects**—like "Customer", "Order", "Survey", or "Task".

Each object:

- Has **attributes** (e.g. name, size, type)
- Has a **state** (e.g. pending, paid, completed)
- Can **do things** (e.g. confirm itself, trigger a reminder)

🧠 How LightCode Helps You Think Smarter

Widgets are the surface. But objects give your app meaning.

In **Chapter 8**, you'll meet interactive objects that:

- Carry multiple fields
- Transition between states
- React to user actions or automated triggers

Example: The "Order" object you'll build has:

- A beverage (text or picklist)
- A payment status (boolean or state)
- Actions like .confirm() and .serve()

This is where LightCode gets powerful—without losing simplicity.

🧠 Final Reflection: Can You Match These?

Let's end with a challenge. Match each **widget** to its most natural **data type** or **behavior**:

Widget	Match With
Checkbox	A. Boolean
Picklist	B. List of options
Text Input	C. Character string
Button	D. Action/Method
Slider	E. Number
Submit Form	F. Save object & trigger behavior

Bonus Thought: In Chapter 8, many of these will be part of **one single object**. Instead of building 10 widgets manually, you'll configure a smart, reusable object that handles it all for you.

✅ You're Ready for Chapter 8

You now understand:

- The differences between app types
- What widgets do, and how they relate to data
- How data flows between browser, server, and storage
- Why low-code platforms simplify—but also shape—your decisions

- That apps aren't just screens—they're **systems with memory, logic, and purpose**

In the next chapter, you'll begin working with **interactive, stateful objects** that **bring your app to life**. You'll see how data changes over time, how actions trigger new states, and how LightCode makes this accessible—even for beginners.

CONCLUSION: FROM SCREENS TO SYSTEMS

You started this chapter thinking about the apps you use every day. Now, you've taken your first real step behind the screen.

You've explored the difference between app types—native, web, mobile—and learned how they run on different devices, with different needs. You've seen how visual widgets like buttons and checkboxes connect to deeper data types, and how low-code platforms simplify the technical maze of building, connecting, and saving.

But most importantly, you've learned that:

- **Not everything happens in one place**—apps are split between browser, memory, server, and cloud.
- **Not everything is permanent**—until you save and sync, your changes may vanish.
- **And not everything is manual**—buttons are just the beginning. Logic can live in objects.

In Chapter 8, you'll stop thinking in individual parts. Instead, you'll start thinking in **interactive objects**—things that behave, evolve, and react. From simple orders to smart assistants, your apps are about to become living systems.

You're not just building screens. You're designing experiences.

And now, you know how to make them work.

Remember before moving on:

- Checkboxes store true or false
- Text inputs hold words and sentences
- Picklists offer choices from a list
- Buttons trigger actions and events

Apps don't just live on your screen—they live in your browser, your memory, the internet, and sometimes in someone else's cloud.

If something disappears, slows down, or acts weird…

💡 *Ask yourself: "Where does this live? And who remembers it?"*

REFERENCES

- Paris Dauphine University. (2023). *WP3.2: Let's Explore the LightCode Platform.* Erasmus+ Project KA220-HED-E1CD9927.

  Provides a foundational overview of the LightCode environment, introducing learners to basic widgets, visual editing, and the logic behind Lego-style application building.

- **Nielsen, J.** (1995). *10 Usability Heuristics for User Interface Design.* Nielsen Norman Group. Retrieved from https://media.nngroup.com/media/articles/attachments/Heuristic_Summary1-compressed.pdf

  Outlines ten fundamental principles for designing intuitive and user-friendly interfaces, aiding learners in understanding the rationale behind common UI elements.

- **Wroblewski, L.** (2011). *Mobile First.* A Book Apart. Retrieved from https://mobile-first.abookapart.com/

  Emphasizes the importance of designing for mobile platforms first, offering strategies that align with the simplified, user-centric approach discussed in this chapter.

- **Sharp, H., Rogers, Y., & Preece, J.** (2019). *Interaction Design: Beyond Human-Computer Interaction* (5th ed.). Wiley.

  Explores the principles of interaction design, providing insights into how users engage with digital systems, which complements the chapter's focus on user interface components and behaviors.

These resources are selected to reinforce the concepts introduced in Chapter 7, offering learners also alternative avenues to explore the topics of app types, user interface elements, and the foundational principles of interaction design further.