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

9. Collaborate with your Team

Paris-Dauphine University – PSL

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
Croatia

Symplexis
Greece

symplexis.eu

Table of Contents

Co-funded by the European Union

OVERVIEW

Introduction

You've learned how to design logic, connect data, and test your app's behavior. But real-world apps are rarely built alone. Whether you're working in a pair or across a whole classroom, the challenge isn't just to build—it's to build together.

This chapter is about how teams create things that work.

Before anyone touches a screen or edits a module, collaboration starts with something deeper: a shared understanding. Why are we building this? Who is it for? What kind of experience do we want to create? These questions shape everything that follows.

As your team begins designing and building together, you'll face practical challenges:

- How do we divide the work?
- How do we avoid breaking each other's changes?
- How do we make sure everything still fits?

LightCode helps with these challenges, but the real key is how your team communicates. You'll learn how to plan collaboratively, assign responsibilities, review progress, and test your work to keep everything consistent—no matter who's contributing.

This chapter will guide you step-by-step, from early planning all the way to shared delivery. Along the way, you'll also discover how LightCode quietly supports teamwork under the hood—while keeping your focus on the people and the process.

LightCode Erasmus+ Project Nr. 2022-1-FR01-KA220-HED-000086863 Co-funded by the European Union

Why is this Module Important?

When you're building an app on your own, you can move fast. You make the decisions. You test. You fix. Done.

But when you build with others, everything changes. It's no longer just about what you know—it's about how well you **share**, **listen**, and **coordinate**. The bigger the team, the more invisible things can go wrong: duplicated effort, lost edits, unclear expectations, or changes that don't quite fit together.

That's why this module matters.

It teaches you how to approach app building as a **team process**, not just an individual task. You'll explore what needs to be clear before you start designing—like the purpose of your app, who it's for, and how your work connects with others. You'll also learn how to structure your team's efforts so that everyone can contribute without stepping on each other's toes.

Whether you're using LightCode or another low-code tool, collaboration introduces risks—but also great rewards. A shared sense of ownership. A stronger, more polished product. And the chance to learn from each other along the way.

LightCode makes this easier by quietly tracking your contributions and letting teams work in parallel. But tools don't make collaboration work—**people do**. This chapter will give you the mindset and practical techniques to build better apps—together.

Learning Outcomes

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

Co-funded by the European Union

- Describe what effective collaboration looks like when designing and building an app as a team.
- Identify the key shared resources that should be clarified before implementation: vision, mission, user personae, specifications, and design.
- Apply general team practices such as role distribution, shared ownership, and incremental delivery to a low-code context.
- Recognize common challenges in low-code team collaboration, including version confusion, conflicting changes, and limited traceability.
- Explain how LightCode enables team collaboration through modular file structures and a tracked editing system.
- Plan a collaborative workflow using LightCode, including dividing work across YAML modules while maintaining collective responsibility.
- Use testing to validate team contributions and ensure consistent app behavior throughout the development process.

Prerequisites

This module is designed for learners who have already explored the fundamentals of app-building in LightCode—especially the concepts of widgets, interactive objects, and testing.

You do **not** need any prior experience with programming, Git, or GitHub.

However, to fully benefit from this chapter, you should be:

- Familiar with the basic components of a LightCode app (objects, attributes, actions).

- Comfortable editing or reviewing YAML-based modules inside LightCode.
- Aware of how testing works from Chapter 8 (e.g., .test.yaml files and expected behavior validation).
- Curious about how people work together—not just how tools work.

If you've ever worked on a group project and wondered, "How do we avoid stepping on each other's toes?", this chapter is for you.

STRUCTURE OF THE MODULE

This chapter is designed to take you from individual app builder to confident team collaborator, step by step.

It begins by showing that collaboration starts before building, with shared goals, expectations, and understanding of your users. From there, it introduces essential teamwork practices—like dividing responsibilities, planning roles, and working iteratively—rooted in the agile mindset.

You'll then explore why collaboration can be difficult in many low-code tools, and how LightCode addresses these challenges by giving you full ownership of your files through GitHub integration.

The middle of the module walks you through:

- How to recognize and manage your own modules and menu
- How to partition work safely across a team
- How to maintain consistency using testing habits

A realistic team scenario brings all of this together, illustrating how four builders divide work and test their progress using LightCode's structure.

Co-funded by the European Union

The final sections highlight the importance of soft skills—like listening, explaining, and being reliable—and how they contribute to better collaboration just as much as technical structure.

Throughout, you'll see that LightCode isn't just a tool for building apps—it's a platform for learning how to work well with others, one clear, testable module at a time.

By the end of this module, you'll be ready to:

- Contribute to a shared project
- Align with a team's vision
- Plan your work modularly
- Test what you build
- And grow as a collaborator

Co-funded by the European Union

COLLABORATION, STEP BY STEP

1. Building Together Starts with Shared Intent

Imagine three learners open LightCode at the same time and start dragging widgets into different modules. They each have ideas, energy, and great intentions—but if they haven't talked first, the result might be a mess. Duplicate buttons. Conflicting data. No flow. Or worse: they might build three completely different apps by accident.

That's why real collaboration doesn't start in the editor.

It starts **before anyone clicks anything**.

🧭 What Good Teams Align On (Before Building)

Before code, before layout, before even naming your app—great teams start with shared clarity:

What to Align On	Why It Matters
Vision	What impact do we want this app to have?
Mission	What problem are we solving, and for whom?
User Personae	Who will use this app? What do they expect or struggle with?
Specifications	What features do we need now? What comes later?
Design & Flow	What should the experience feel like, step by step?

These are not just planning documents.

Co-funded by the European Union

They are **living resources** your team should revisit and refine as the app evolves.

📚 Where to Keep Shared Intent

Your team doesn't need special tools for this. Use what's easiest:

- A shared Google Doc or Notion page
- A hand-drawn sketch uploaded to the chat
- A sticky-note wall in a classroom
- A voice memo with a user scenario

💬 *"Who are we helping, and what should it feel like to use our app?"*

If your team can answer that together—you're ready to build.

🧠 Collaboration = Design of Responsibility

In LightCode, your "code" is visual and modular.

But **collaboration is still about design**—not just of screens, but of responsibility.

Ask early:

- Who's guiding the experience?
- Who owns which part of the logic?

Co-funded by the European Union

- Who’s making sure it all connects?

Once this intent is shared, the technical collaboration (which we’ll explore next) becomes much smoother—and much more meaningful.

2. Principles of Agile and Healthy Teamwork

Once your team shares a common purpose, the next challenge is building together—without losing sync or momentum. This is where agile thinking offers guidance.

The agile movement, born in the world of software development, promotes adaptability, collaboration, and continuous improvement.

Since low-code platforms make it easy to try ideas, gather feedback, and make changes quickly, the agile way of working is especially well suited to this environment.

Agile isn’t a fixed recipe. It’s a set of values that help teams stay flexible, focused, and human. These values are especially useful when you’re moving quickly in a visual tool like LightCode, where ideas evolve fast and decisions are made collaboratively.

Here are the four core values of the Agile Manifesto, and what they mean for you:

> 🧑‍🤝‍🧑 **Individuals and interactions over processes and tools**
>
> Even the best tool can’t replace a quick conversation or shared understanding.

In LightCode, it’s easy to work visually—but that doesn’t mean you should skip discussing who’s doing what and why. Strong teams talk first.

✅ Working software over comprehensive documentation

Instead of waiting to get everything “perfect” on paper, agile teams prefer to build small pieces and try them. In LightCode, this means previewing, testing, and improving as you go—rather than spending days planning before anything runs.

🤝 Customer collaboration over contract negotiation

Even if your users are just classmates or teachers, agile teams keep them close. Show early versions of your app. Ask for feedback. Adapt. Collaboration with your users leads to better design decisions than guessing from afar.

🔁 Responding to change over following a plan

Plans are useful—but they’re not the finish line. In every team project, surprises happen. A user may get stuck. A teammate may find a better idea. Agile thinking encourages teams to treat change as part of progress—not a failure of planning.

Together, these values remind us that great teamwork isn’t just efficient—it’s responsive, thoughtful, and centered on real users.

LightCode gives you the tools to act on these values. But it’s your habits, communication, and openness that bring them to life.

3. What Low-Code Platforms Often Struggle With

Low-code platforms open amazing doors. You can go from idea to working prototype in a single afternoon. You don't need to know HTML, CSS, or APIs. And instead of writing dozens of lines of code, you just drag, drop, connect, and configure.

But like any tool, low-code has its blind spots—especially when multiple people are building at once.

Here are some common struggles:

⚠️ Changes Are Instant—Sometimes Too Instant

In many low-code tools, updates are live as soon as you hit save. That's great for speed, but risky for collaboration.

If two people edit the same app at once, one can accidentally erase the other's work—without realizing it.

🔍 No Real Version History

Traditional coding tools keep a history of every change. Low-code platforms often don't.

That means:

- No way to see what changed (and who changed it).
- No way to go back to a previous version safely.
- No easy comparison between "before" and "after."

This makes debugging and reviewing more difficult—especially in larger teams.

🧪 Testing Is Rarely Built-In

While low-code is great for building interfaces and flows, it often lacks serious testing tools.

Without consistent testing, teams may:

- Break things without realizing.
- Lose trust in each other's changes.
- Ship apps that behave unpredictably.

🔄 Collaboration Becomes Manual

Since low-code platforms don't always support branching, merging, or review workflows, collaboration becomes a manual process:

- "You work on this today, I'll do that tomorrow."
- "Don't save anything yet—I'm still editing."
- "Let's meet before we try to publish anything."

This slows down teams and increases the risk of mistakes.

💡 Why This Matters for You

Even if LightCode makes many things easier, these challenges still exist beneath the surface. Knowing about them will help you:

- Design your team process more carefully.
- Divide responsibilities more clearly.
- Use LightCode's built-in tools more wisely.

Co-funded by the European Union

And most importantly, it prepares you to become a **conscious collaborator**, not just a fast builder.

4. How LightCode Supports Teams

After seeing the challenges that many low-code platforms face, you might wonder: *How can teams avoid overwriting each other's work or losing progress when changes happen fast?*

This is where **LightCode takes a different approach**.

Behind the scenes, LightCode stores every app as a set of **structured files**—each written in a format called **YAML**. These files are not locked inside the platform. Instead, they live in **your own GitHub repository**.

That means:

- You and your team own the project.
- Every change is tracked and recoverable.
- You can view, compare, and manage versions anytime.

🧠 But what if I don't know Git or GitHub?

Don't worry. You don't need to.

LightCode takes care of the technical side for you. You can edit and build visually—just like before. The platform quietly saves and organizes those edits inside your team's GitHub space.

Still, if you want to:

- You can browse your files on **GitHub's web interface**.
- You can see who changed what, and when.

Co-funded by the European Union

- You can even make edits directly (for advanced users).

🧩 Why This Matters

By combining visual editing with open, professional version control, LightCode gives your team:

- The speed of low-code,
- The safety of tracked history,
- And full ownership of your work.

This makes collaboration safer and smoother:

- You work on your part.
- Your teammate works on theirs.
- Everyone can contribute without risk of overwriting or guessing.

And when something breaks? You’re not stuck. You can review, compare, and roll back with confidence.

Your app belongs to you—and your whole team can help shape it.

Co-funded by the European Union

5. 🖼️ What It Looks Like to Own Your Work in LightCode

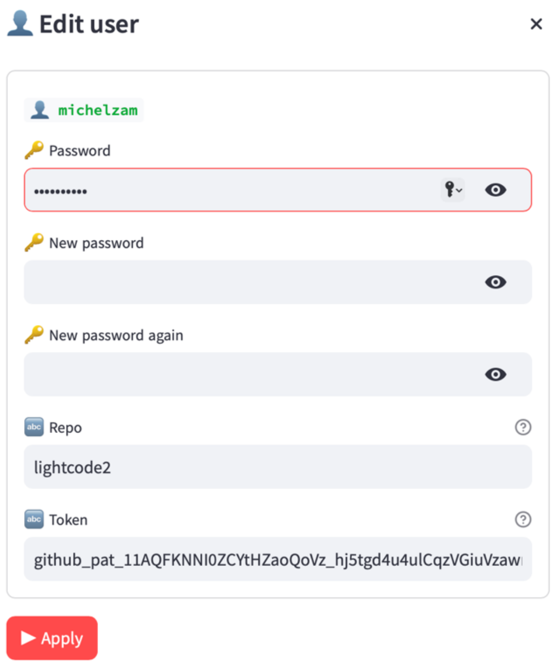
In LightCode, you're not just building inside a visual tool—you're building on top of your own GitHub repository. That means every object, page, test, or menu you create becomes a real file you control, visible both inside LightCode and in your repo.

Let's walk through a simple example: customizing your app's menu.

5.1 Connect to Your GitHub Repository

When setting up your profile, you link LightCode to your GitHub account using a secure token and specify which repository to use. From this point on, LightCode saves everything directly to your chosen GitHub space.

Every app you create becomes your own version-controlled project.



👤 Edit user ×

👤 michelzam

🔑 Password

••••••••••

🔑 New password

🔑 New password again

Repo

lightcode2

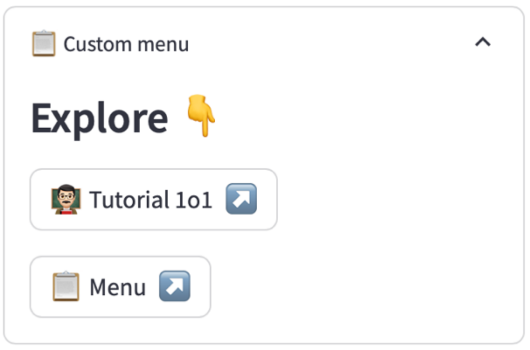
Token

github_pat_11AQFKNNI0ZCYtHZaoQoVz_hj5tgd4u4ulCqzVGiuVzaw

▶ Apply

5.2 Your App Menu Comes from Your Repo (Yes, Really!)

After configuring your GitHub token and repo, you'll land on your app's main page. You might see a **custom menu** already displayed.



Co-funded by the European Union

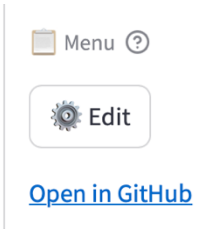
That menu? It's not just a UI—it's **rendered from your own YAML file in GitHub**.

Congratulations: your repo is now live and driving the interface.

In this example, the custom menu shows links to two modules:

- A first module (e.g., a "Tutorial 1o1")
- A second one that points **to the menu module itself**

That second link is editable: when you open it in **design mode**, you can update the icon, title, or internal description—just like any other module.

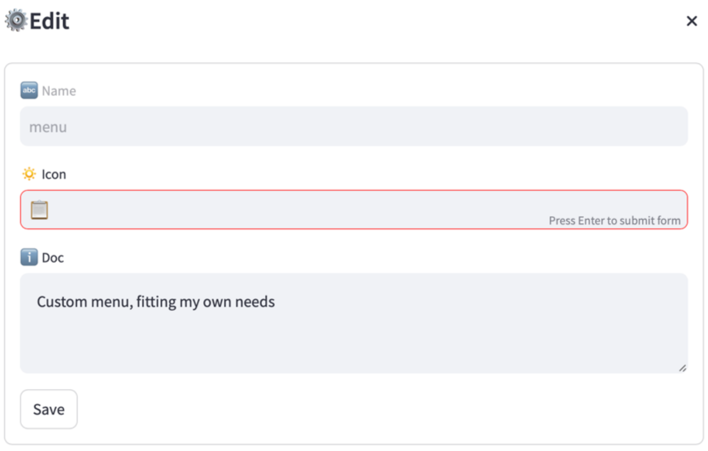


5.3 Customize a Module, Your Way

LightCode makes it easy to personalize your menu structure and documentation. You're free to:

- Rename documentation
- Add icons and help text
- Organize your app into meaningful sections

Every change you make here updates your GitHub repo in real time when you save it.



Name
menu

Icon
Press Enter to submit form

Doc
Custom menu, fitting my own needs

Save

5.4 Quick Access: "Open in GitHub"

From within LightCode, you can click **"Open in GitHub"** to inspect or manage your files directly. You'll see structured YAML, just like what powers your app logic.

It's not a simulation—it's real.

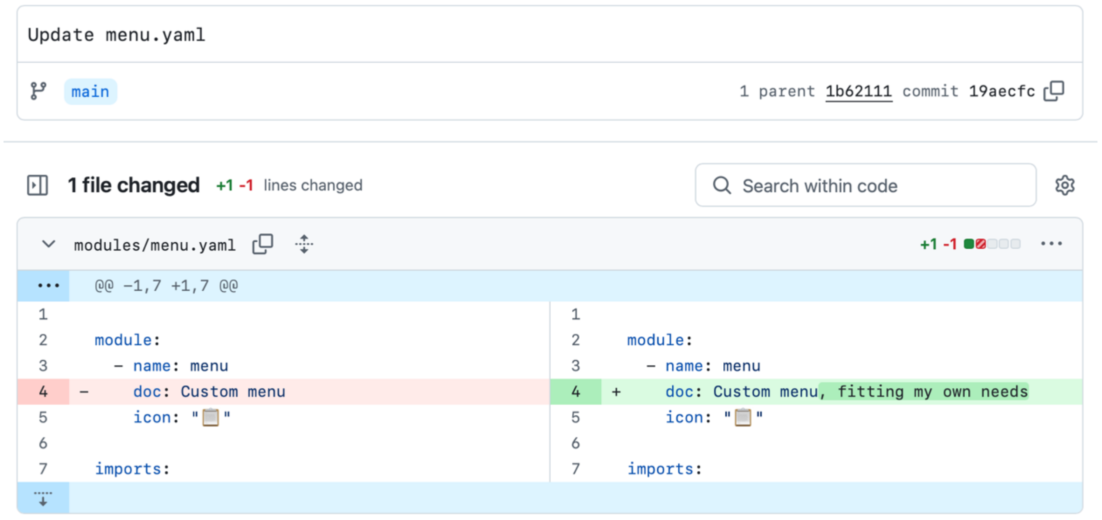
5.5 Track What Changed (and When)

Your GitHub repo gives you full access to:

- File history
- Side-by-side comparisons
- Author tracking for each edit

Even if you're not a developer, this level of clarity helps you:

- Understand collaboration dynamics
- Review your own work
- Safely experiment without fear of losing progress



Update menu.yaml

main   1 parent 1b62111 commit 19aecfc

1 file changed +1 -1 lines changed   Search within code

modules/menu.yaml   +1 -1

```
@@ -1,7 +1,7 @@

module:
  - name: menu
-   doc: Custom menu
    icon: "📋"

imports:
```

```

module:
  - name: menu
+   doc: Custom menu, fitting my own needs
    icon: "📋"

imports:
```

✅ You’re Not Just a Builder—You’re a Maintainer

LightCode teaches you that every screen, menu, and module is more than just something you click through. It’s a **file**, it lives in **your repo**, and it can evolve with your project.

Ownership, persistence, and transparency—these aren’t just features.

They’re skills.

Perfect — here’s the next section, **Section 6: Dividing Work Across Modules**, continuing seamlessly from what learners just experienced in the previous walkthrough. This section introduces a **collaboration-friendly structure**, while reinforcing the LightCode principle of **collective ownership**.

6. Dividing Work Across Modules

Now that you know every module in LightCode is a file in your GitHub repo, you can start thinking like a team:

How do we divide our work without stepping on each other’s changes?

LightCode makes this easy by encouraging a **modular structure**: each part of the app lives in its own file. You can edit one module while your teammate edits another—no conflict, no confusion.

Co-funded by the European Union

🧩 A Simple Collaboration Strategy

When working in teams, a helpful pattern is:

Module Type	Managed By	Example
Shared entry module	The whole team	main.yaml, menu.yaml for layout or routing
Specialized modules	Individuals or small groups	order.yaml, survey.yaml, quiz.yaml, etc.
Test modules*	Shared or per object	test_order.yaml, test_quiz.yaml

This lets each person—or pair—focus on a specific object or process, while keeping integration clean.

(*) In LightCode, unit tests follow Martin Fowler's SelfTestingCode vision: https://martinfowler.com/bliki/SelfTestingCode.html

🛠️ LightCode Makes It Natural

- Each module is visual and self-contained.
- Team members can open different modules in parallel.
- GitHub tracks edits module by module—so you can review work independently.

No special setup is required. Partitioning work in LightCode is just a matter of **clear agreement** among teammates.

⚠️ Ownership Is Temporary, Not Territorial

Even if one person creates or edits a module, they don't "own" it forever.

This kind of partitioning is just a **working convention**—useful for clarity and speed, but always flexible.

LightCode encourages **collective ownership**. Everyone should be able to:

- Read each other's modules
- Understand how parts connect
- Jump in and help when needed

This ensures the team isn't slowed down by waiting, bottlenecks, or missing knowledge.

✅ Think Like a Team

When you plan your modules:

- Discuss early: who's working on what?
- Share your progress often: test, review, align
- Treat your repo as a **shared space**—not a silo

LightCode gives you the structure.

You give it the teamwork.

7. Keeping It Together with Tests

When you work in a team, it's easy for small changes to break things—especially when multiple people are building in parallel. A form might stop showing. A method might stop behaving as expected. And no one notices until it’s too late.

That’s why the best teams make testing part of their daily rhythm.

🧪 Testing Is Not Just for Debugging

Testing isn’t something you save for the end. It’s a way to:

- Check your own work as you go.
- Confirm that recent changes didn’t break something else.
- Help your teammates understand what’s supposed to happen.

When everyone builds, **everyone should test**.

🔁 Make Testing a Team Habit

Keep it simple:

- If you change how something behaves—test it.
- If you add something new—test it.
- If you review someone else’s work—test it.

Even a small set of consistent checks helps the whole team move faster and with more trust.

Co-funded by the European Union

✅ Testing Helps Teams Stay in Sync

In collaborative projects, testing is how you:

- Catch mistakes early
- Prevent surprises during sharing or publishing
- Keep quality from being one person’s responsibility

You don’t need to write complicated rules.

Just keep asking: **“Does this still do what we expect?”**

If the answer is yes—great.

If not—it’s a perfect time to talk, align, and improve together.

🧠 Pro Tip: Want to Go Further? Try TDD

Some teams follow a professional testing mindset called **Test-Driven Development** (TDD). The idea is simple but powerful:

Write the test **before** you build the feature.

This way, your app is always designed to pass a known expectation.

The TDD cycle looks like this:

1. 🔴 **Red** – Write a test that fails (because the feature doesn’t exist yet).
2. 🟢 **Green** – Build just enough to make the test pass.
3. 🏗️ **Refactor** – Clean up the code or logic without breaking the test.

It’s a habit used in serious software teams—but it’s just as useful in low-code.

Co-funded by the European Union

Learning to think this way is a **great skill to add to your resume** as a builder.

Great! Here's the draft for **Section 8: A Team Scenario – Collaboration in Practice**, bringing all the previous principles together into a practical, relatable team story. It's short, clear, and shows how collaboration, modular structure, and testing come together naturally in LightCode.

8. A Team Scenario – Collaboration in Practice

Let's bring everything together with a real-world team example.

👩🏽‍💻👨🏼‍💻👩🏿‍💻 Meet the Team

Four students are building a learning app together in LightCode.

They've already agreed on a shared purpose and user flow:

"We're making a quiz app for language learners. It should let users pick a topic, take a quiz, and get feedback."

They hold a short kickoff call to clarify:

- Who's handling which parts
- What their menu and module names will be
- When they'll check in with each other

🧩 How They Divide the Work

Team Member	Task	Module Name
Alex	Quiz structure & scoring	quiz.yaml
Jamie	User preferences & language	settings.yaml
Riley	Menu & navigation	main.yaml, menu.yaml
Sam	Builds initial test cases	test_quiz.yaml, test_settings.yaml

Each person works in their own file—while all files live in the same shared GitHub repository.

🛠 How They Collaborate

- Each student opens only the modules they're editing.
- Changes are saved automatically and tracked in the repo.
- When a teammate finishes something, they share a preview link in the group chat.
- If something breaks, the team checks the test files to identify what changed—and why.

They're not using any complicated systems. Just:

- Shared understanding
- Clear division of work
- Visual editing

- Light but useful testing habits

✅ The Outcome

After two days, they've built a working prototype:

- The menu works.
- The quiz can be played.
- Test files confirm that everything flows as expected.

No one had to touch Git.

No one lost someone else's work.

No one waited for permission.

They collaborated—not just on screens, but on purpose, logic, and learning.

Excellent — here's the final **additional section** to place just before the conclusion, focusing on soft skills as **core enablers of collaboration**. It fits naturally after testing, module structure, and team practices.

💙 Soft Skills That Make You a Great Collaborator

You've learned how to divide work, test what you build, and track your progress across shared files. But even with great tools and structure, collaboration only works if people feel safe, heard, and aligned.

That's where **soft skills** come in.

In LightCode—and in most modern digital teams—your ability to **communicate clearly, stay curious, and support others** is just as valuable as your ability to build features.

Co-funded by the European Union

Here are the soft skills that make the biggest difference:

Soft Skill	**What It Looks Like in Practice**
Listening	Understanding what your teammate wants to build before suggesting your version.
Explaining	Leaving clear comments or descriptions when you change a module.
Asking	Reaching out early when something is unclear or unexpected.
Giving feedback	Sharing ideas respectfully and focusing on improvement, not blame.
Receiving feedback	Staying open—even when your first version isn't perfect.
Reliability	Communicating when you'll finish your part, and following through.

These aren't side notes. They're **professional assets**.

In team projects—especially with remote tools like LightCode—**how you work matters as much as what you build**. Developing these habits now means:

- Your team runs more smoothly.
- You build more trust.
- You're better prepared for real-world collaboration in any tech, design, or product setting.

And yes, these are skills worth putting on your résumé.

Co-funded by the European Union

CONCLUSION: COLLABORATION IS MORE

🤝 Collaboration Is More Than Just Dividing Tasks

In this chapter, you didn't just learn how to share a project—you learned how to share responsibility, understanding, and improvement.

You saw that real collaboration:

- Begins before building—with shared vision and clear intentions
- Relies on structure—like modular files and version tracking
- Grows through habits—like testing and regular feedback
- Depends on trust—made stronger through clear communication and care

You discovered that LightCode encourages these practices:

- Each module is a file in your GitHub repo
- Every change is saved and visible
- Your work can be previewed, shared, and improved with others

But even the best tools don't guarantee good teamwork.

You do.

Your willingness to listen, explain, adapt, and align is what turns a group of contributors into a team. These are skills that make collaboration joyful—and real.

✅ Skills You Practiced in This Chapter

Technical & Organizational

- Structuring modular apps across multiple collaborators
- Using GitHub for persistent, trackable app versions
- Editing, reviewing, and previewing modules in parallel
- Maintaining consistency through shared testing habits

Agile Mindset

- Working iteratively with early feedback
- Adjusting plans when learning something new
- Balancing clarity with flexibility

Soft Skills

- Listening to teammates and aligning before acting
- Asking thoughtful questions when things aren’t clear
- Giving and receiving feedback constructively
- Following through on shared plans and responsibilities

💡 Remember Before You Build Together Again

- Apps are easier to build when everyone shares the same goal
- Modular structure avoids overlap—but only if everyone stays in sync
- Testing protects your app—and your team’s time
- GitHub helps track your work—but communication keeps it smooth
- Team habits matter more than team tools

Co-funded by the European Union

You’re not just building features. You’re building alignment, rhythm, and shared understanding. That’s collaboration. And it’s a real skill—worth keeping, using, and even adding to your résumé.

Co-funded by the European Union

REFERENCES

- **Beck, K., et al.** (2001). *Manifesto for Agile Software Development*. Agile Alliance. Retrieved from https://agilemanifesto.org

  The foundation of agile thinking, focused on collaboration, adaptability, and user-centered development.

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

  Discusses team communication, shared understanding, and design thinking as essential components of building user-focused digital systems.

- **Hoda, R., Noble, J., & Marshall, S.** (2012). *Developing a Grounded Theory to Explain the Practices of Self-Organizing Agile Teams*. Empirical Software Engineering, 17(6), 609–639.

  Explores how agile teams self-organize, share ownership, and coordinate complex work—relevant to collaborative low-code teams.

- **GitHub Docs.** (2024). *Introduction to GitHub*. Retrieved from https://docs.github.com/en/get-started

  Clear documentation on GitHub's basic features, useful for understanding the platform LightCode uses for file tracking and team version control.

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

  Introduces the core structure of LightCode apps, including how visual modules map to YAML files stored in GitHub.

Co-funded by the European Union