Configuration Pattern: Before & After Comparison

Side-by-side comparison of all configuration sections showing current vs. proposed patterns.

Legend: - 🔴 **Current** - Existing format - 🟢 **Proposed** - New pattern variant - 📊 **Variant** - Which pattern variant used (A/B/C)

Session 0: Getting Ready for Camp

Section 1: Introducing Yourself

📊 **Variant:** B (Hybrid) - 3 steps

🔴 **Current**

👋 Introducing Yourself

Goal: Connect with your cohort and share your coding journey.

Step 1: Log in to the portal

Go to the Wizcamp Portal and log in with your account.

Step 2: Create your post

Click "New Introduction" and share:

- Your name and coding experience
- What excites you about learning to code
- One fun fact about yourself

Step 3: Engage with others

Read and respond to other student introductions.

💡 **Why this matters:** Building connections with your cohort creates a supportive learning community where you can share challenges, celebrate wins, and learn from

each other throughout the course.

🟢 **Proposed**

👋 Introducing Yourself

🎯 **Goal:** Connect with your cohort and share your coding journey.

1. **Log in** to the Wizcamp Portal
2. **Click** "New Introduction" and share:
   - Your name and coding experience
   - What excites you about learning to code
   - One fun fact about yourself
3. **Read** and respond to other student introductions

Verify: Your introduction post is visible in the portal.

💡 **Why this matters:** Building connections with your cohort creates a supportive learning community where you can share challenges, celebrate wins, and learn from each other throughout the course.

Changes: - ✅ Kept goal statement - ✅ Removed step subheadings - ✅ Used numbered list with bold verbs - ✅ Added brief verification - ✅ Kept callout

---

Section 2: Planning Your Schedule

📊 **Variant:** A (Minimal) - 1 step

🔴 **Current**

📅 Planning Your Schedule

Go to the class calendar within the Wizcamp Portal and RSVP for each session you plan on attending. You'll receive confirmation and reminders.

🟢 **Proposed**

📅 Planning Your Schedule

Go to the class calendar within the Wizcamp Portal and RSVP for each session you plan on attending. You'll receive confirmation and reminders.

💡 **Stay organized:** RSVPing helps you plan your schedule and ensures you receive session reminders.

Changes: - ✅ Kept minimal format (1 step) - ✅ Added callout for context

---

Section 3: Setting Up Your GitHub Account

📊 **Variant:** A (Minimal) - 2 steps

🔴 **Current**

🐙 Setting Up Your GitHub Account

Go to github.com and create a new account (signup guide). If you already have an account, verify you can log in.

💡 **Why GitHub matters:** GitHub is where developers worldwide store and share code. You'll use it throughout the course to access your coding environment and AI-powered coding assistant.

🟢 **Proposed**

🐙 Setting Up Your GitHub Account

1. Go to github.com and create a new account (signup guide)
2. If you already have an account, verify you can log in

💡 **Why GitHub matters:** GitHub is where developers worldwide store and share code. You'll use it throughout the course to access your coding environment and AI-powered coding assistant.

Changes: - ✅ Numbered the 2 steps for clarity - ✅ Kept callout - ✅ No goal statement (obvious from title)

Section 4: Connecting with Your Instructor

📊 **Variant:** A (Minimal) - 1 step

🔴 **Current**

📨 Connecting with Your Instructor

DM Mr. Colestock in the Wizcamp Portal with your full name, email, and GitHub username. You'll receive confirmation that you've been added to the Wizcamp GitHub Organization.

💡 **Why this matters:** Being added to the organization gives you access to Codespaces and GitHub Copilot Pro — the tools you'll use to build your trivia game.

🟢 **Proposed**

📨 Connecting with Your Instructor

DM Mr. Colestock in the Wizcamp Portal with your full name, email, and GitHub username. You'll receive confirmation that you've been added to the Wizcamp GitHub Organization.

💡 **Why this matters:** Being added to the organization gives you access to Codespaces and GitHub Copilot Pro — the tools you'll use to build your trivia game.

Changes: - ✅ No changes needed (already optimal for 1-step task)

Session 1: Setting Up Your Trivia Game

Section 5: Creating Your Codespace

📊 **Variant:** B (Hybrid) - 4 steps

🔴 **Current**

☁️ Creating Your Codespace

🎯 **Goal:** Set up your cloud development environment so you can start coding without any local installations.

Step 1: Sign in to GitHub

Go to github.com and log in with your account.

Step 2: Launch the template

Go to github.com/wizcamp/wizcamp-realms-demo and click **Use this template** → **Open in a codespace**.

Expected result: Your Codespace begins building (this takes 1-2 minutes).

Step 3: Wait for the environment to load

Once the build completes, VS Code will open in your browser.

Expected result: Project files appear in the file explorer on the left side.

Step 4: Customize your theme (optional)

Click the ⚙️ gear icon in the bottom left → **Themes** → **Color Theme** → pick your favorite (Dark+ is popular for coding).

💡 **Why Codespaces rocks:** Everyone gets the same setup — no more "it works on my machine" drama. Mess something up? Just delete and start fresh. It's like having a reset button for your entire dev environment. You'll use this Codespace for all 12 sessions.

🏆 **Bonus Challenge:** Go to github.com/codespaces to explore more about managing your Codespaces.

🟢 **Proposed**

☁️ Creating Your Codespace

🎯 **Goal:** Set up your cloud development environment so you can start coding without any local installations.

1. **Sign in** to github.com
2. **Navigate** to github.com/wizcamp/wizcamp-realms-demo
3. **Click** Use this template → Open in a codespace
4. **Wait** for the build to complete (1-2 minutes)

Verify: VS Code opens in your browser with project files visible in the left sidebar.

💡 **Why Codespaces rocks:** Everyone gets the same setup — no more "it works on my machine" drama. Mess something up? Just delete and start fresh. You'll use this Codespace for all 12 sessions.

🏆 **Bonus Challenge:** Customize your theme by clicking the ⚙️ gear icon → Themes → Color Theme, or explore github.com/codespaces to learn about managing Codespaces.

Changes: - ✅ Kept goal statement - ✅ Removed step subheadings - ✅ Condensed to numbered list with bold verbs - ✅ Single verification statement - ✅ Moved optional customization to bonus - ✅ Streamlined callout

---

Section 6: Starting Your Development Server

📊 **Variant:** C (Full) - 3 steps with commands

🔴 **Current**

🚀 Starting Your Development Server

🎯 **Goal:** Practice starting and stopping the local server you'll use to preview real-time changes as you build your app.

Step 1: Run the dev server

From the terminal at the bottom of your Codespace, run:

```
npm run dev
```

Step 2: Open the app in your browser

After running the command, you'll see output like:

```
VITE v7.1.7  ready in 2473 ms

➜  Local:   http://localhost:5173/
➜  Network: use --host to expose
➜  press h + enter to show help
```

Follow the link (ctrl + click), copy-paste it into a new browser tab, or click “Open in Browser” if a dialog appears.

Expected result: A web page displaying the starter app with placeholder content.

Step 3: Stop the server

Go back to your terminal and press `Ctrl + C`.

Expected result:

- Terminal returns to the command prompt
- Refreshing the browser shows a connection error (app no longer running)

💡 **Your dev server cheat sheet:** Run `npm run dev` to fire up your server and see your app live. Hit `Ctrl + C` to shut it down. You’ll use these commands constantly — they’re about to become muscle memory.

🟢 **Proposed**

🚀 Starting Your Development Server

🎯 **Goal:** Practice starting and stopping the local server you’ll use to preview real-time changes as you build your app.

Step 1: Run the dev server

From the terminal at the bottom of your Codespace, run:

```
npm run dev
```

Step 2: Open the app in your browser

After running the command, you'll see output like:

```
VITE v7.1.7  ready in 2473 ms

➜  Local:   http://localhost:5173/
➜  Network: use --host to expose
➜  press h + enter to show help
```

Follow the link (ctrl + click), copy-paste it into a new browser tab, or click "Open in Browser" if a dialog appears.

Expected result: A web page displaying the starter app with placeholder content.

Step 3: Stop the server

Go back to your terminal and press `Ctrl + C`.

Expected result:

- Terminal returns to the command prompt
- Refreshing the browser shows a connection error (app no longer running)

💡 **Your dev server cheat sheet:** Run `npm run dev` to fire up your server and see your app live. Hit `Ctrl + C` to shut it down. You'll use these commands constantly — they're about to become muscle memory.

Changes: - ✅ **NO CHANGES** - Already optimal for complex config with commands and output - ✅ Variant C is appropriate here

---

Session 2: Building Game Components

Section 7: Accessing Your Codespace

📊 **Variant:** A (Minimal) - 1 step

🔴 **Current**

☁️ Accessing Your Codespace

Visit github.com/codespaces to relaunch your Codespace from Session 1.

🟢 **Proposed**

☁️ Accessing Your Codespace

Visit github.com/codespaces to relaunch your Codespace from Session 1.

💡 **Quick tip:** Your Codespace saves your work automatically. You can close and reopen it anytime without losing progress.

Changes: - ✅ Added callout for reassurance

Section 8: Installing React DevTools

📊 **Variant:** B (Hybrid) - 4 steps

🔴 **Current**

🔍 Installing React DevTools

React DevTools is like X-ray vision for your React app — see component structure, props, and state in real-time.

Browser Installation

Browser	Installation Link	Notes
Chrome	Chrome Web Store	Most popular choice
Firefox	Firefox Add-ons	Great alternative
Edge	Edge Add-ons	Windows default

Safari	Manual installation required	Advanced users only

Using DevTools

1. **Open** DevTools by pressing F12 or right-clicking → Inspect
2. **Find** Components tab by looking for "Components" next to Console, Network, etc.
3. **Explore** your app by clicking on components in the tree to see their props
4. **Inspect** GameButton by finding your GameButton component and see the text, onClick, and variant props!

💡 **React DevTools** gives you X-ray vision into your app. You can inspect components, props, and state in real time — essential for debugging and understanding how your app works under the hood.

🟢 **Proposed**

🔍 Installing React DevTools

🎯 **Goal:** Install and learn to use React DevTools for inspecting your app's components.

Browser Installation

Browser	Installation Link	Notes
Chrome	Chrome Web Store	Most popular choice
Firefox	Firefox Add-ons	Great alternative
Edge	Edge Add-ons	Windows default
Safari	Manual installation required	Advanced users only

Using DevTools

1. **Open** DevTools by pressing `F12` or right-clicking → Inspect
2. **Find** the Components tab next to Console, Network, etc.

3. **Click** on GameButton in the component tree
4. **Inspect** the props: `text`, `onClick`, and `variant`

Verify: You can see GameButton's props in the right panel.

💡 **Why this matters:** React DevTools gives you X-ray vision into your app — essential for debugging and understanding how components work.

Changes: - ✅ Added goal statement - ✅ Kept platform table - ✅ Simplified usage steps (removed redundancy) - ✅ Added brief verification - ✅ Streamlined callout

---

Session 3: Managing Game Flow

Section 9: Using React DevTools for Exploring State

📊 **Variant:** C (Full) - 6 exploratory steps

🔴 **Current**

🔍 Using React DevTools for Exploring State

Let's use React DevTools to see how **shared state** works behind the scenes and experiment with changing it manually.

1. **Open** DevTools by pressing F12 or right-clicking → Inspect
2. **Find** Components tab by looking for "Components" next to Console, Network, etc.
3. **Locate** GameProvider by clicking on GameProvider in the component tree
4. **Examine** the hooks by looking for the screen state value (if you don't see hook names clearly, click the gear icon and enable "Parse hook names")
5. **Experiment** with state by changing the screen value from "splash" to "playing" and watch the UI update!
6. **Change** it back by setting it back to "splash" to see the SplashScreen return

💡 React DevTools gives you X-ray vision into your app's **state**. You can see exactly what data each component has and even modify it in real-time. This is invaluable for debugging and understanding how **shared state** affects your entire app. Notice how changing one value in `GameProvider` instantly changes what component renders!

🏆 Bonus Challenge

Try changing the screen state to different values and see what happens. What occurs when you set it to a value that doesn't match any of your conditions?

🟢 **Proposed**

🔍 Using React DevTools for Exploring State

🎯 **Goal:** Explore how shared state controls your app by manually changing values in DevTools.

Step 1: Open DevTools

Press `F12` or right-click → Inspect to open your browser's developer tools.

Step 2: Navigate to Components tab

Look for "Components" next to Console, Network, etc. and click it.

Expected result: You see a tree of React components.

Step 3: Locate GameProvider

Click on GameProvider in the component tree.

Expected result: The right panel shows GameProvider's hooks and state.

Step 4: Find the screen state

Look for the `screen` state value in the hooks section. If hook names aren't clear, click the gear icon and enable "Parse hook names."

Step 5: Experiment with state

Change the `screen` value from `"splash"` to `"playing"` and watch the UI update instantly!

Step 6: Change it back

Set `screen` back to `"splash"` to see the SplashScreen return.

Verify: The app switches between screens as you change the state value.

💡 **Why this matters:** React DevTools gives you X-ray vision into your app's state. You can see exactly what data each component has and modify it in real-time — invaluable for debugging and understanding how shared state affects your entire app.

🏆 **Bonus Challenge:** Try changing the screen state to different values and see what happens. What occurs when you set it to a value that doesn't match any of your conditions?

Changes: - ✅ Added goal statement - ✅ Added step subheadings (6 steps = complex) - ✅ Added expected results after key steps - ✅ Streamlined instructions - ✅ Kept bonus challenge - ✅ Improved callout

---

Summary of Changes

By Variant

Variant A (Minimal) - 4 sections: - Planning Schedule - GitHub Account - Connecting with Instructor - Relaunch Codespace

Variant B (Hybrid) - 3 sections: - Introducing Yourself - Creating Codespace - Installing DevTools

Variant C (Full) - 2 sections: - Starting Dev Server (no changes) - Exploring State (enhanced)

Key Improvements

1. **Reduced cognitive load** - Removed step subheadings for 3-4 step tasks
2. **Better scannability** - Bold action verbs in numbered lists
3. **Appropriate structure** - Complexity matches content
4. **Maintained student support** - Kept verification and callouts
5. **Streamlined content** - Removed redundancy while keeping clarity

What Stayed the Same

- Goal statements (where appropriate)
- Callouts explaining value
- Verification points

- Bonus challenges
- Code blocks and expected output
- Platform-specific tables

Recommendation

Adopt these changes because they: - ✅ Reduce unnecessary structure for simple tasks - ✅ Maintain clarity and student support - ✅ Align with industry best practices - ✅ Scale appropriately with complexity - ✅ Keep unique educational value (goals, callouts, verification)