# Material Design UI Styling and Dummy Data Best Practices for a Task Management App

## UI/UX Styling with Material Design Principles

Designing a polished task management interface requires following **Material Design** guidelines for consistency and a modern feel. This includes styling all interactive elements (links, buttons, cards, menus, etc.) with appropriate hover states, focus indicators, smooth transitions, and subtle animations. It’s important to create a responsive, desktop-first UI that feels lively but professional. Below are best practices and examples inspired by Material Design and leading productivity apps:

### Interactive Elements & States (Links, Buttons, etc.)

* **Buttons:** Implement all standard states – **default**, **hover**, **pressed/active**, **focused**, **disabled**, etc.[[1]](https://www.uxpin.com/studio/blog/button-states/#:~:text=Buttons%20typically%20have%20four%20to,found%20in%20modern%20product%20development)[[2]](https://www.uxpin.com/studio/blog/button-states/#:~:text=Hover%20state). In Material Design, buttons often have a flat default style with a **ripple** effect on press and slight elevation on hover. For example, Material guidelines suggest that a raised button’s hover state can be achieved by overlaying a 12% opacity black shade (for subtle shading)[[3]](https://stackoverflow.com/questions/47988813/wpf-material-design-raised-button-hovered-state#:~:text=I%27m%20trying%20to%20recreate%20a,could%20be%20achieved%20in%20WPF). In practice, *hovering* should slightly change the button’s color or elevation to indicate interactivity, without a drastic change[[4]](https://www.uxpin.com/studio/blog/button-states/#:~:text=Design%20principles%20and%20best%20practices%3A). The *pressed/clicked* state typically triggers a **ripple overlay** – a quick, radial animation signifying the press[[5]](https://becquerel.brandcompete.com/directives/ripple#:~:text=Ripples%20are%20visual%20representations%20used,that%20need%20visual%20feedback%20upon). Ensure the pressed visual (ripple or shade) briefly remains and then disappears, confirming the action was registered[[6]](https://www.uxpin.com/studio/blog/button-states/#:~:text=Apply%20a%20visual%20effect%20like,when%20the%20click%20is%20released). The *focused* state (e.g. when tabbing via keyboard) should be clearly indicated, often by an outline or highlight around the button[[7]](https://www.uxpin.com/studio/blog/button-states/#:~:text=Design%20principles%20and%20best%20practices%3A), which is crucial for accessibility. Disabled buttons should be visibly “off” (e.g. grayed out with reduced opacity) yet still recognizable in shape[[8]](https://www.uxpin.com/studio/blog/button-states/#:~:text=Design%20principles%20and%20best%20practices%3A).
* **Links:** Style text links so they are clearly distinguishable as clickable. Material-UI’s default is to use the theme’s **primary color** for link text so it stands out, and to underline it by default[[9]](https://mui.com/material-ui/react-link/#:~:text=However%2C%20the%20Link%20component%20has,props%20than%20the%20Typography%20component)[[10]](https://mui.com/material-ui/react-link/#:~:text=,button). This aligns with usability guidelines that recommend colored, underlined links for maximum clarity[[10]](https://mui.com/material-ui/react-link/#:~:text=,button). In a polished design, you might use an underline only on hover if you prefer a cleaner look, but be sure that the link’s color has sufficient contrast and is used consistently so users recognize it. Avoid using generic “click here” text – links should have meaningful labels (e.g. “View Project Details” instead of “click here”) for clarity and accessibility[[10]](https://mui.com/material-ui/react-link/#:~:text=,button). On hover, a link can slightly darken or show an underline if not always underlined, providing a subtle cue that it’s interactable.
* **Other UI Components:** Apply consistent state styling to *all interactive UI components*. For instance, **menu items, list rows, or cards** that are clickable should highlight on hover (e.g. a light background fill or elevation increase) to show they are active targets. Material Design uses state layers for this – a semi-transparent overlay of a color (often the component’s primary or “on-surface” color at ~8% opacity for hover) to indicate hover, and a higher opacity for pressed state[[3]](https://stackoverflow.com/questions/47988813/wpf-material-design-raised-button-hovered-state#:~:text=I%27m%20trying%20to%20recreate%20a,could%20be%20achieved%20in%20WPF). Ensure these state indications are *“subtle yet noticeable”*[[4]](https://www.uxpin.com/studio/blog/button-states/#:~:text=Design%20principles%20and%20best%20practices%3A) – for example, a card might lift or its shadow deepen slightly when hovered, indicating it’s clickable, but without shifting layout. All interactive icons (like edit/delete icons visible on hover) should also follow a similar pattern (e.g. a slight color highlight on hover).
* **Consistency and Visual Hierarchy:** Maintain a **consistent style** across components to reinforce familiarity[[11]](https://www.uxpin.com/studio/blog/button-states/#:~:text=Visual%20consistency). Use a coherent color scheme (e.g. your app’s primary color for all primary actions, a secondary color for secondary actions) and apply Material Design’s **color & contrast** guidelines. For instance, using a slightly darker shade of the primary color on hover states helps ensure the element remains visible when hovered[[12]](https://www.uxpin.com/studio/blog/button-states/#:~:text=Leverage%20color%20and%20contrast%20to,ensure%20the%20button%20remains%20noticeable). Primary buttons might be a filled style with bold color, whereas secondary buttons might be outlined – but both should still show hover feedback (color change or ripple) in line with the same principles. Align with Material Design’s typography and spacing as well, to give a clean, professional baseline.

### Transitions and Micro-Animations

Polish the experience with **smooth transitions** and small animations that guide the user without overwhelming them. Material Design emphasizes using motion purposefully to convey state changes and navigational transitions. Here are some recommendations:

* **Hover & Click Transitions:** Rather than instant state changes, use brief CSS transitions (e.g. 150–200ms ease-in) for hover effects on buttons and links. For example, fading a button’s background or elevation over 0.2s on hover makes it feel smoother[[13]](https://www.uxpin.com/studio/blog/button-states/#:~:text=Use%20subtle%20transitions%20for%20state,from%20hover%20to%20active%20state). When a button is pressed, the state change (e.g. ripple and color change) should also be animated quickly (~100–300ms) so it’s noticeable but doesn’t delay the interaction. *Avoid overly long or elaborate animations* for basic states, as those can feel sluggish; the goal is a **subtle feedback** that feels “snappy” and responsive[[13]](https://www.uxpin.com/studio/blog/button-states/#:~:text=Use%20subtle%20transitions%20for%20state,from%20hover%20to%20active%20state)[[14]](https://www.uxpin.com/studio/blog/button-states/#:~:text=,and%20causing%20potential%20%2021). A recommended practice is a short fade-in for hover and a slightly longer fade-out, which creates a natural feeling transition[[13]](https://www.uxpin.com/studio/blog/button-states/#:~:text=Use%20subtle%20transitions%20for%20state,from%20hover%20to%20active%20state).
* **Page Transitions & Dialogs:** If your task app has multiple views (e.g. switching between a task list and a calendar, or opening a task detail), implement Material Design’s motion patterns. Common approaches include **fade or slide transitions** when modals or side-panels open/close, and **content animations** when navigating between pages. For example, clicking a task could slide in a details panel from the right, or open a dialog with a quick scale/fade animation. Material Design suggests using consistent easing curves and durations (often in the 200–300ms range) for such transitions so they feel natural. These touches make the app feel modern and polished, akin to how a Google Workspace app might slide open a panel for document details or how Asana opens task details with a smooth animation.
* **Microinteractions:** Little animations on specific user actions provide feedback and delight. **Asana** is famous for its playful microinteraction when completing a task – occasionally a unicorn animation flies across the screen to celebrate[[15]](https://www.nngroup.com/articles/microinteractions/#:~:text=To%20keep%20track%20of%20day,an%20example%20of%20a%20microinteraction). While you may not add confetti and unicorns for an in-house tool, consider *subtle microinteractions*: for instance, a brief highlight on a task row that was just updated, or a slight bounce on a dragged-and-dropped task card to confirm its new position. These micro-animations should also serve usability: e.g. when a task is marked complete, show a checkmark icon with a quick fade-in/out, or gray out the task and move it to “Done” with a smooth motion. Microinteractions can also convey system status or undo opportunities. As **Nielsen Norman Group** notes, when Asana users mark a task complete, a confirmation dialog appears (with an “undo” option) and then the celebratory animation plays – providing both functional feedback and positive reinforcement[[16]](https://www.nngroup.com/articles/microinteractions/#:~:text=When%20a%20user%20marks%20a,inspired%20unicorn%20animation). Aim for similar patterns: on critical actions, give feedback (e.g. a snackbar or small toast “Task completed – Undo” and perhaps a subtle animation to draw attention) so users feel the system responded.
* **Inspiration from Google Workspace & Notion:** Polished apps like Google’s and Notion achieve a feeling of fluidity by keeping animations **quick and context-appropriate**. For example, in Google apps (Docs, Calendar), interactive buttons use the Material ripple on click, menus smoothly fade in/out, and focus states are clearly indicated with outlines – all of which you can emulate. **Notion** offers a clean, minimal interface but uses microinteractions like showing a plus button when you hover near a new line, or gently animating the expansion of toggled lists. The common thread is **restraint with purpose**: use animations to **clarify interactions** (not to show off). A *well-timed 200ms animation* can guide the user’s eye to a change (like a newly inserted task or a menu opening) without feeling flashy[[13]](https://www.uxpin.com/studio/blog/button-states/#:~:text=Use%20subtle%20transitions%20for%20state,from%20hover%20to%20active%20state). Keep the overall UX snappy: ensure that any animated feedback doesn’t impede rapid interaction.

### Accessibility and Feedback

A truly polished UI also accounts for accessibility and consistent feedback:

* **Focus Indicators:** As mentioned, always provide a visible focus indicator on interactive elements for keyboard users[[7]](https://www.uxpin.com/studio/blog/button-states/#:~:text=Design%20principles%20and%20best%20practices%3A). This can be a blue outline or shadow (Material UI by default adds a focus ring). This not only helps users navigating via keyboard, but also contributes to a professional, compliant design.
* **Error and Status Feedback:** Material Design principles say to use color and small animations for feedback. For example, if a form fails to save a task, you might briefly shake the dialog or highlight the field in red, accompanied by an error message – similar to how Google might animate a password field with a subtle shake on error. For success feedback (like saving changes), a short lived message or icon change (e.g. a check icon appearing) can reassure the user. These are beyond basic styling but part of a polished UX.
* **Consistent Iconography:** Consider using Material Icons or a consistent icon set for actions (edit, delete, comment, etc.) to match the Material aesthetic. Ensure icons also have hover states (e.g. change opacity or color on hover to indicate they are clickable). Google Workspace apps, for example, often turn icon buttons a darker shade on hover and show a tooltip – small touches you can incorporate for clarity.

By following Material Design’s system for color, typography, spacing, and the above interactive behaviors, your app will feel cohesive and modern. The key is to **apply these styles uniformly** (every button, link, and card should behave in a familiar way) and to incorporate just enough motion to guide and reward the user (but not so much as to distract). This approach mirrors what we see in polished products like Google’s apps, Asana, and Notion – clean layouts, smooth feedback on every interaction, and thoughtful details that enhance usability.

## Populating Realistic Sample Task & Project Data

When testing your task management tool in-house, **realistic sample data** is essential. Instead of obvious filler text or dummy content, populate the app with projects and tasks that **feel authentic** to users. This helps testers immerse in the scenario and better evaluate the UI and workflow. Below are guidelines and examples for structuring believable sample projects, tasks, and team data:

### Avoid Obvious Dummy Content (Keep It Real)

* **No Lorem Ipsum or “Task 1/Task 2”:** Avoid generic placeholders that break the illusion of a real project. Seeing names like *“John Doe”* or tasks labeled *“Task 123”* will instantly feel fake and lazy to users[[17]](https://supademo.com/blog/dummy-data/#:~:text=1). Instead, use plausible names and titles that one might actually see in a project. *For example:* use a person’s name like **“Alex Chen”** or **“Priya Menon”** instead of “Test User”, and a task title like **“Draft Project Proposal”** instead of “Sample Task”. As one guide notes, if you’re showing a project management tool, populate it with tasks that have owners, deadlines, and statuses *“in motion”* — not just static placeholders[[18]](https://supademo.com/blog/dummy-data/#:~:text=1). The dummy data should tell a story about real work, not force the tester to imagine what “Task 1” might represent.
* **Mirror Real Workflows:** Think of a realistic scenario that your team might undertake, and build the sample data around that. For instance, if your organization does software development, create a sample project for a **“New Feature Release”** or **“Website Redesign Q4 2025”**. If it’s a general productivity tool, you could also include a non-software project like **“Marketing Campaign Launch”** or **“Event Planning for Annual Conference”**. Choose project types that resonate with typical team activities. Then fill those projects with tasks that logically fit the workflow (design tasks, development tasks, review tasks, etc.), mimicking how a real team would break down work. This *mirroring of actual workflows* makes the demo data immediately relatable[[18]](https://supademo.com/blog/dummy-data/#:~:text=1).
* **Diversity and Edge Cases:** Populate a range of data, not one-note. Include tasks of varying sizes and priorities, and sprinkle in a few edge cases or anomalies to simulate real-life. For example, have some tasks marked with **urgent/high priority**, and maybe one task that is **overdue or past its deadline**, to see how the UI highlights it. Showing an edge case like an overdue task or a blocked task can make the scenario more convincing and test how the app handles it[[19]](https://supademo.com/blog/dummy-data/#:~:text=4). Also vary the types of work – e.g. some tasks can be bugs, others features, others chores – to reflect a realistic mix. A common mistake is using the same kind of dummy item repeatedly (all tasks of the same type or all deadlines set to the same date); real projects have variety, so ensure your sample data does too (mixing different categories, dates, etc.)[[20]](https://supademo.com/blog/dummy-data/#:~:text=One%20type%20of%20record%20won%E2%80%99t,your%20product%20feel%20more%20robust).

### Sample Project Scenarios and Tasks

Below are a couple of **example project scenarios** with realistic task breakdowns. These illustrate how to craft project data that looks authentic and professionally relevant:

* **Project: Website Rebrand Launch** – *An example software/design project.*
* **Description:** A project to redesign and launch a new company website in Q4. Involves design, content, development, and marketing coordination.
* **Sample Tasks:**
  + *Design New Homepage Layout* – **Assignee:** Dana (UI/UX Designer) – **Status:** In Progress.  
    *Description:* “Create high-fidelity homepage mockups aligning with new brand guidelines. Include mobile-responsive designs.”  
    *Comments:* – *Manager:* *“Looks great so far. Please ensure the hero section matches the new branding colors.”* – *Designer:* *“Will do; adding the updated logo and palette now.”*
  + *Implement Homepage Frontend* – **Assignee:** Alex (Frontend Developer) – **Status:** Not Started (To Do).  
    *Description:* “Develop the homepage in React using the new design. Set up responsive breakpoints and integrate with CMS for content.”  
    *Comments:* – (No comments yet; linked to design task for reference).
  + *Migrate Old Content* – **Assignee:** Maria (Content Strategist) – **Status:** Completed.  
    *Description:* “Audit existing site content and migrate or rewrite as needed for the new pages (Home, About, Features).”  
    *Comments:* – *Maria:* *“All main pages content is updated in the Google Doc. Ready for review.”*
  + *SEO and Analytics Setup* – **Assignee:** Jamal (Marketing Specialist) – **Status:** In Progress.  
    *Description:* “Set up SEO meta tags for new pages and integrate Google Analytics & Search Console.”  
    *Comments:* – *Jamal:* *“Analytics code added; will monitor metrics post-launch.”*
  + *Launch Coordination* – **Assignee:** (multiple – **Team Task** for PM + IT) – **Status:** Blocked (Dependency).  
    *Description:* “Coordinate go-live on Friday: backup old site, deploy new site, QA post-deployment.”  
    *Comments:* – *Project Manager:* *“Waiting on final OK from design and dev before scheduling deploy.”* – *IT:* *“Deployment script is ready, just give the word.”*
* **Status Distribution:** Among these tasks, you can see a realistic spread: a couple *In Progress*, one *Completed*, one *To Do*, and one *Blocked* awaiting dependencies. In a real project, not everything is “Done” or “To Do”; having various statuses makes the project feel active and authentic. The “Blocked” task here adds realism – teams often have a task paused pending something else.
* **Assignees:** Note how different roles are represented (Designer, Developer, Content, Marketing, PM, IT). Each task is assigned to a specific person with a plausible job role. Using distinct, real-sounding names (Dana, Alex, Maria, Jamal, etc.) avoids the dummy data pitfall[[17]](https://supademo.com/blog/dummy-data/#:~:text=1) and helps testers envision real colleagues in those roles.
* **Project: Q4 Marketing Campaign – “Product X Launch”** – *A marketing/sales project example.*
* **Description:** A project for planning and executing the Q4 launch campaign for *Product X*, involving content creation, outreach, and event planning.
* **Sample Tasks:**
  + *Campaign Plan & Timeline* – **Assignee:** Priya (Marketing Manager) – **Status:** Completed.  
    *Description:* “Outline the campaign strategy, key messages, channels (email, social, ads), and timeline of deliverables.”  
    *Comments:* – *Priya:* *“Plan finalized. Review the timeline in the attached doc and provide feedback by EOD.”* – *CMO:* *“Reviewed, looks solid. Let’s proceed.”*
  + *Design Ad Creatives* – **Assignee:** Leo (Graphic Designer) – **Status:** In Progress.  
    *Description:* “Design a set of banner ads and social media graphics. Formats needed for Twitter, LinkedIn, newsletter, and print.”  
    *Comments:* – *Leo:* *“Draft versions for banners uploaded. Waiting for feedback on style.”*
  + *Landing Page Development* – **Assignee:** Nina (Web Developer) – **Status:** In Progress.  
    *Description:* “Develop a new landing page for the campaign, including lead capture form and integration with CRM.”  
    *Comments:* – *Nina:* *“Page is live at staging URL for QA. Need final copy to replace placeholders.”*
  + *Content Writing: Blog & Email* – **Assignee:** Rahul (Content Writer) – **Status:** To Do.  
    *Description:* “Write a blog post announcing Product X and draft the email newsletter content. Emphasize new features and value prop.”  
    *Comments:* – (No comments yet; task not started)
  + *Launch Webinar Event* – **Assignee:** Team (Marketing & Sales) – **Status:** To Do.  
    *Description:* “Plan and schedule a live webinar for the launch. Include coordinating speakers, creating slide deck, and promotions.”  
    *Comments:* – *Priya:* *“Tentatively looking at Nov 15th for webinar date. Will firm up once speakers confirm.”*
* **Additional Realism:** This project involves cross-team collaboration (marketing, design, web dev, sales), reflected in the assignee list. Some tasks are individual, others could have multiple collaborators or a team label. We included a task waiting on copy (“Landing Page” waiting on final text), which is a believable dependency in a marketing project. Statuses range from *Completed* planning to *In Progress* design/dev tasks and *To Do* tasks that are upcoming. This staggered progress mirrors how real projects have some parts already done while others are yet to start. The task names are descriptive and specific (no “Misc Task” or “Dummy”), and each has context in its description so a tester can understand what it is at a glance.

### Tips for Believable Data Setup

* **Use Realistic Names and Roles:** Create a roster of fictional team members with diverse, professional names (e.g. include different genders and cultural backgrounds in names for realism) and assign tasks to them. If possible, have profiles with their role or department. This way, when someone sees tasks in the app, they recognize a *pattern*: tasks are owned by **different people** who logically would do them (design tasks to designers, etc.), which is what they’d expect in a real scenario. It also avoids the impression that one test user owns everything.
* **Detailed Descriptions & Comments:** Write a short description for each task that explains its purpose or what’s needed. Aim for a single sentence or two that sounds like a requirement or note from a real project (as shown in examples). For comments, add a few where it makes sense – e.g. a manager giving feedback, or a team member asking a question. These should resemble actual team communication: *“@Alex can you update the client about this delay?”* or *“I’ve attached the draft design, let me know your thoughts.”* Such comments give life to the data and let testers see how collaboration would appear. **Avoid empty tasks** with no details; even a one-liner description like “Implement the authentication API endpoint” is better than leaving a field blank or with filler text.
* **Status and Progress Distribution:** Configure your sample project so that not everything is 100% done or 0% done. A believable project has some tasks finished, some in progress, some not started, maybe a couple running late. For instance, in a set of 10 sample tasks you might mark 2 as **Completed**, 5 as **In Progress**, 2 as **To Do**, and 1 as **Blocked** or **Delayed**. This distribution helps test all UI states (you can see how completed tasks are shown vs. active tasks) and also feels authentic. Users will recognize the scenario of juggling multiple ongoing tasks. *If your app tracks deadlines*, set a mix of past and future dates (e.g. one task due 2 days ago – which should show as overdue, a few due in the next week, etc.). This again mirrors reality and checks that the UI highlights overdue items properly.
* **Avoid Repetition:** Don’t use the same description or same date for every task. It’s a common dummy data mistake to, say, have five tasks all due on “Dec 31” or all named “Sample Task X”. Instead, stagger the due dates and vary the language. Even small tweaks help – e.g. use different verbs (“Design”, “Implement”, “Write”, “Review”) in task titles to reflect variety. Ensure each task has a unique purpose in the project, to avoid the appearance of clones.
* **Privacy and Professionalism:** Since this is in-house testing, you likely don’t have to worry about real personal data. But still, treat the dummy content professionally – no joke names or inappropriate filler that wouldn’t appear in a real work context. The data should be safe to show to stakeholders and feel like it could be a real (if fictitious) project at your company. By using fictitious yet realistic data, you also avoid any privacy issues that would come from using actual client or employee data in a test[[21]](https://supademo.com/blog/dummy-data/#:~:text=Never%20use%20customer%20records%2C%20even,you%20get%20realism%20without%20risk).

By carefully planning your sample data, you create a **richer testing environment**. Testers can engage with the app as if they were managing an actual project, which yields more genuine feedback on the UI/UX. Moreover, when demonstrating the tool to others (e.g. internal demos or training), the realistic data *“tells a story”* about how the app supports real workflows[[22]](https://supademo.com/blog/dummy-data/#:~:text=But%20the%20difference%20between%20an,features%20and%20telling%20a%20story). This makes your product easier to connect with, as people can immediately see realistic scenarios instead of blank screens or obviously dummy text[[23]](https://supademo.com/blog/dummy-data/#:~:text=But%20the%20difference%20between%20an,features%20and%20telling%20a%20story).

Following the above guidance, you’ll achieve two things: a **fully Material-themed interface** that feels modern and consistent, and a **set of sample projects** that convincingly simulate real team usage. Together, these contribute to a highly professional, polished in-house trial of your task management app – one where users focus on coordinating work (not on glaring UI issues or laughable placeholder data). By paying attention to the small details (both in UI styling and in content realism), you set your tool up for a smoother testing phase and better adoption within the team.

**Sources:**

* Material Design & UX guidelines for interactive states and transitions[[4]](https://www.uxpin.com/studio/blog/button-states/#:~:text=Design%20principles%20and%20best%20practices%3A)[[13]](https://www.uxpin.com/studio/blog/button-states/#:~:text=Use%20subtle%20transitions%20for%20state,from%20hover%20to%20active%20state)[[5]](https://becquerel.brandcompete.com/directives/ripple#:~:text=Ripples%20are%20visual%20representations%20used,that%20need%20visual%20feedback%20upon)
* UX design best practices for button states (hover, active, focus)[[4][7]](https://www.uxpin.com/studio/blog/button-states/#:~:text=Design%20principles%20and%20best%20practices%3A) and smooth animations[[13]](https://www.uxpin.com/studio/blog/button-states/#:~:text=Use%20subtle%20transitions%20for%20state,from%20hover%20to%20active%20state)
* Examples of microinteractions improving UX (Asana’s completion animation)[[15]](https://www.nngroup.com/articles/microinteractions/#:~:text=To%20keep%20track%20of%20day,an%20example%20of%20a%20microinteraction)[[16]](https://www.nngroup.com/articles/microinteractions/#:~:text=When%20a%20user%20marks%20a,inspired%20unicorn%20animation)
* Material-UI component styling defaults (links and their underlines)[[9]](https://mui.com/material-ui/react-link/#:~:text=However%2C%20the%20Link%20component%20has,props%20than%20the%20Typography%20component)[[10]](https://mui.com/material-ui/react-link/#:~:text=,button)
* Guidance on creating authentic dummy data and avoiding placeholders[[17][18]](https://supademo.com/blog/dummy-data/#:~:text=1)[[19]](https://supademo.com/blog/dummy-data/#:~:text=4)

[[1]](https://www.uxpin.com/studio/blog/button-states/#:~:text=Buttons%20typically%20have%20four%20to,found%20in%20modern%20product%20development) [[2]](https://www.uxpin.com/studio/blog/button-states/#:~:text=Hover%20state) [[4]](https://www.uxpin.com/studio/blog/button-states/#:~:text=Design%20principles%20and%20best%20practices%3A) [[6]](https://www.uxpin.com/studio/blog/button-states/#:~:text=Apply%20a%20visual%20effect%20like,when%20the%20click%20is%20released) [[7]](https://www.uxpin.com/studio/blog/button-states/#:~:text=Design%20principles%20and%20best%20practices%3A) [[8]](https://www.uxpin.com/studio/blog/button-states/#:~:text=Design%20principles%20and%20best%20practices%3A) [[11]](https://www.uxpin.com/studio/blog/button-states/#:~:text=Visual%20consistency) [[12]](https://www.uxpin.com/studio/blog/button-states/#:~:text=Leverage%20color%20and%20contrast%20to,ensure%20the%20button%20remains%20noticeable) [[13]](https://www.uxpin.com/studio/blog/button-states/#:~:text=Use%20subtle%20transitions%20for%20state,from%20hover%20to%20active%20state) [[14]](https://www.uxpin.com/studio/blog/button-states/#:~:text=,and%20causing%20potential%20%2021) Button States Explained – How to Design them | UXPin

<https://www.uxpin.com/studio/blog/button-states/>

[[3]](https://stackoverflow.com/questions/47988813/wpf-material-design-raised-button-hovered-state#:~:text=I%27m%20trying%20to%20recreate%20a,could%20be%20achieved%20in%20WPF) c# - WPF Material Design Raised Button Hovered state - Stack Overflow

<https://stackoverflow.com/questions/47988813/wpf-material-design-raised-button-hovered-state>

[[5]](https://becquerel.brandcompete.com/directives/ripple#:~:text=Ripples%20are%20visual%20representations%20used,that%20need%20visual%20feedback%20upon) Ripple | Becquerel UI

<https://becquerel.brandcompete.com/directives/ripple>

[[9]](https://mui.com/material-ui/react-link/#:~:text=However%2C%20the%20Link%20component%20has,props%20than%20the%20Typography%20component) [[10]](https://mui.com/material-ui/react-link/#:~:text=,button) Links - Material UI

<https://mui.com/material-ui/react-link/>

[[15]](https://www.nngroup.com/articles/microinteractions/#:~:text=To%20keep%20track%20of%20day,an%20example%20of%20a%20microinteraction) [[16]](https://www.nngroup.com/articles/microinteractions/#:~:text=When%20a%20user%20marks%20a,inspired%20unicorn%20animation) Microinteractions in User Experience - NN/G

<https://www.nngroup.com/articles/microinteractions/>

[[17]](https://supademo.com/blog/dummy-data/#:~:text=1) [[18]](https://supademo.com/blog/dummy-data/#:~:text=1) [[19]](https://supademo.com/blog/dummy-data/#:~:text=4) [[20]](https://supademo.com/blog/dummy-data/#:~:text=One%20type%20of%20record%20won%E2%80%99t,your%20product%20feel%20more%20robust) [[21]](https://supademo.com/blog/dummy-data/#:~:text=Never%20use%20customer%20records%2C%20even,you%20get%20realism%20without%20risk) [[22]](https://supademo.com/blog/dummy-data/#:~:text=But%20the%20difference%20between%20an,features%20and%20telling%20a%20story) [[23]](https://supademo.com/blog/dummy-data/#:~:text=But%20the%20difference%20between%20an,features%20and%20telling%20a%20story) A Guide to Dummy Data: Examples, Generators & Best Practices

<https://supademo.com/blog/dummy-data/>