4 Things You Need to Know to Become an Expert in SAP Build Apps

Mark Wright, Director Technical Product Marketing, SAP

Las Vegas

2024

SAPinsider

Agenda

1. Learn the principles of good app developme
2. Best practices for designing your app
3. Best practices for developing your app
4. Collaboration and teamwork

Is SAP Build Apps right for your project?

Define your needs

- Check that SAP Build Apps meets your **security** requirements.
- Verify that you can access the **data** you need with SAP Build Apps or if you will need additional services.
- Determine which **features** are critical and what is available in SAP Build Apps.

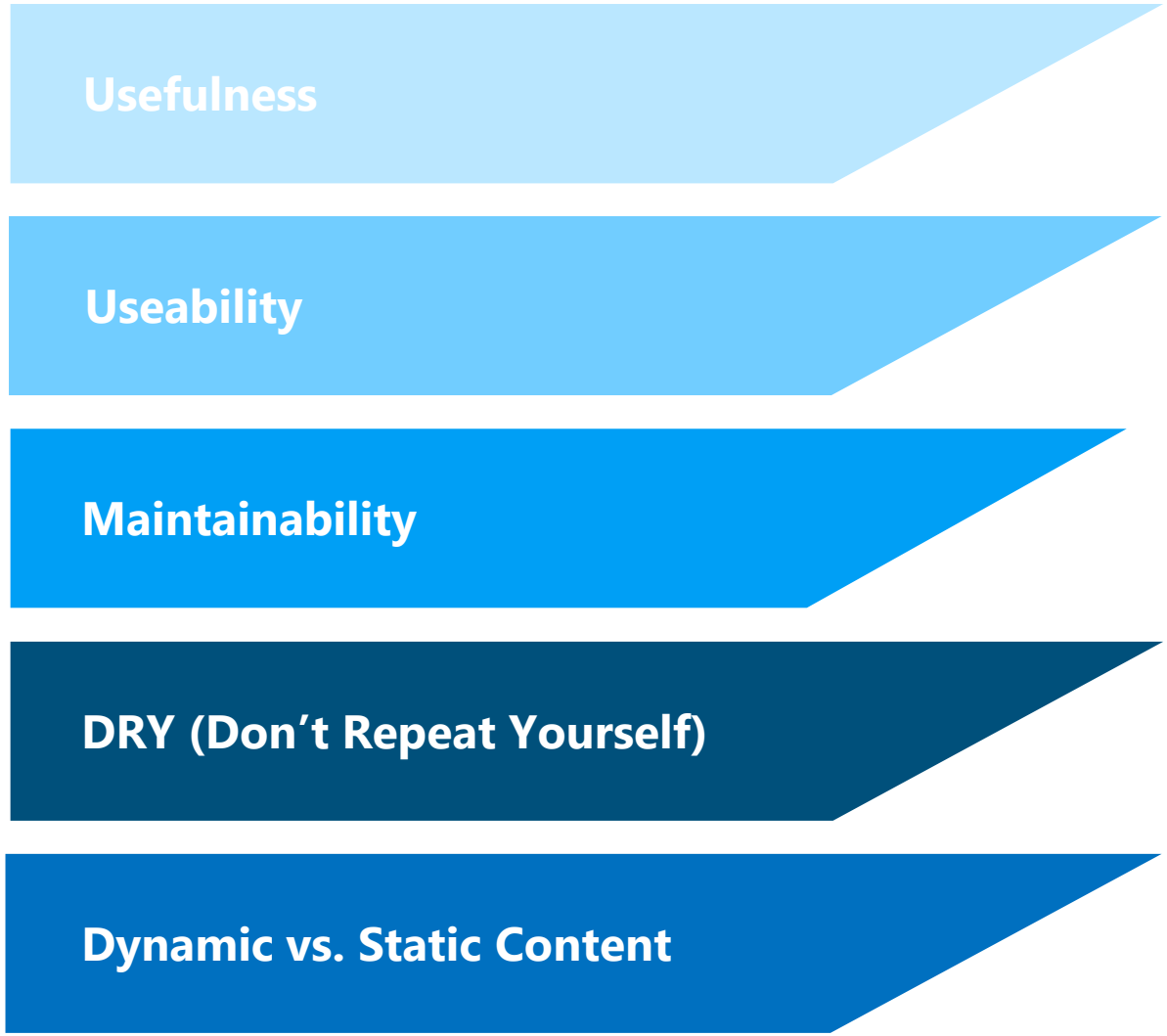
Define your requirements

- Who are the **users** of your app?
- What **tasks** are they trying to accomplish?
  - What are they currently using to do this?
  - What steps are needed to accomplish it?

Pro tip: Use the resources in your organization and find out who the stakeholders of your application are. Is there an IT administrator who can provide you with the required access rights and perhaps a new BTP destination you would need? Are there designers who could collaborate to make a good user experience?

Learn the Principles of Good App Development

The principles of good app development



Usefulness: Is this app effective at solving a problem the users have?

Effectiveness

A useful app provides an **effective** way for the users to solve a problem or concern they have.

Keep it simple

Don't try to solve too many problems in one application, as it will grow huge and more likely to become difficult to use and maintain.

Achieving **usefulness:** When first developing your application, aim for a minimum viable product (MVP) instead of adding functionalities the users may not need or use. More functionalities can be added later if necessary.

Usability: Is this app easier to use than what they're currently doing?

Usable

A usable app is **easy and intuitive** for the user to use to complete the task they have.

Consult

If usability and user experience are unfamiliar topics to you, it's recommended to get a designer or consultant involved in designing the experience for the application.

Achieving **usability:** Implement error prevention and handling as you go. Prevent users from taking actions that would cause them to end up in erroneous states in the app, but if there's an error, communicate about it clearly.

Maintainability: Is the app maintainable by other people in your organization?

Future-Proof

If not, it will grow outdated quickly or never be taken into use.

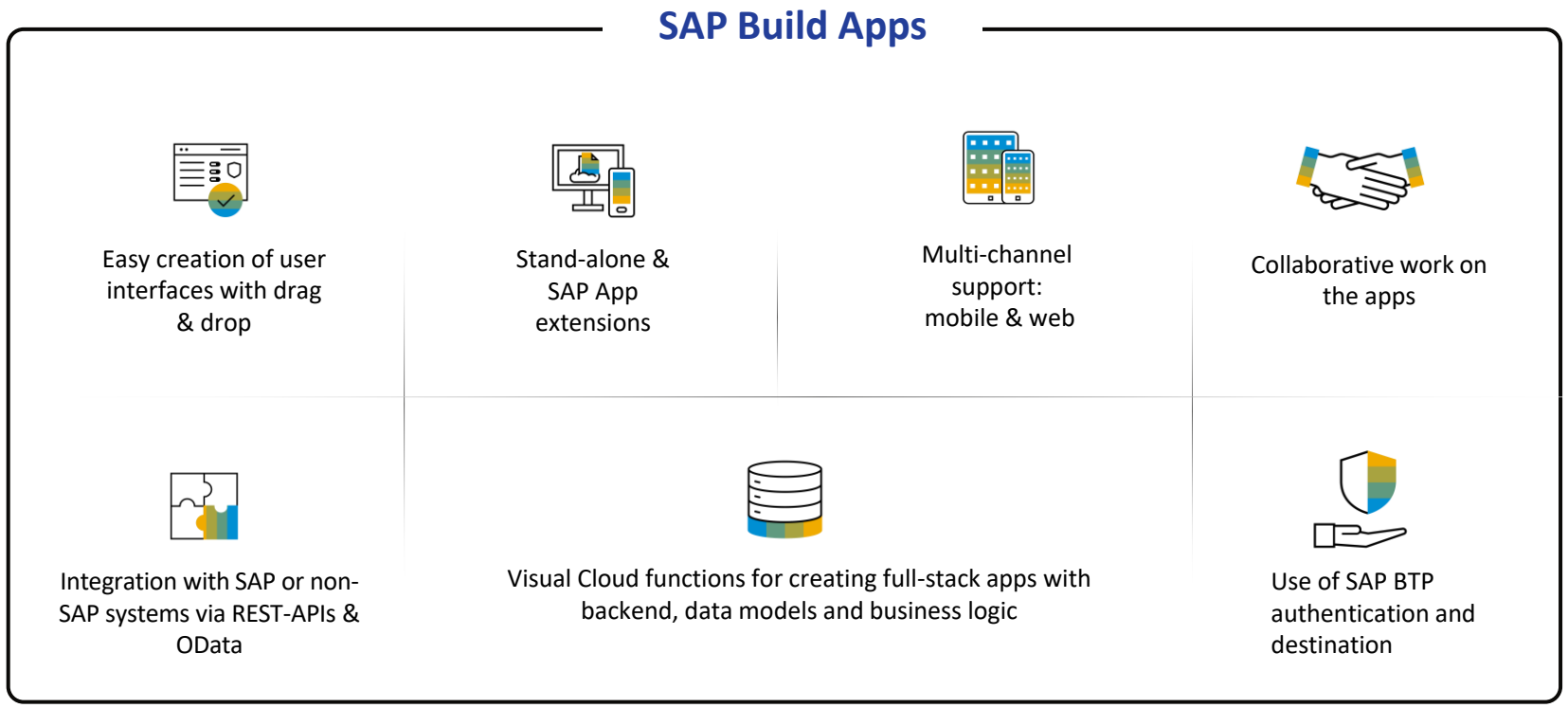
Structured

A maintainable app is **structured and named in an understandable way**, and fellow developers can easily know where to look when they need to change something. It also follows the DRY (Don't Repeat Yourself) principle.

Achieving **maintainability:** Consistency is key in maintainability, and consistency requires a plan of how the app functionality is structured.

Choose a Solution That Delivers

Enterprise-grade apps with an intuitive visual drag-and-drop experience, regardless of skill level



DRY (Don’t Repeat Yourself)

Don’t Repeat Yourself (DRY) is a core principle of all software development. Its aim in traditional pro-code development is to reduce duplicated code or functionality. When there's duplicated code or functionality, there may be bugs or inconsistencies in the application if one implementation of the functionality slightly differs from another.

- Whenever changes are made to the application, these need to be applied in all the spots where the duplicated code exists.
- The same principle applies to no-code development: If functionality is duplicated in different parts of the application, it becomes more error-prone and difficult to maintain.
- In SAP Build Apps, DRY is especially relevant for the logic of the application (See ‘Developing your app’)

Dynamic vs. Static Content

To understand this concept, consider this example:

Imagine an app that has a catalogue of products. You want the app to show a list of products, and when the user clicks or taps on a specific product name, they're taken to a page where more information about the product is given. Let’s say that you have 100 products in your catalogue.

With static content: You would create a separate page for each of the products and manually type or copy-paste content for each product on their page. You would have 100 product details pages in your app, and one list page. This app would very quickly become unmaintainable, as to change the information of some of the products or the layout for all the product details pages, you would have to do a lot of manual work on separate pages to do this.

With dynamic content: You would store the information of the products in a database and show that in the application dynamically. You would have one list page and one product details page. When the user would click or tap on a product name on the list, you would pass the identifier of that product onto the product details page, and then fetch the information for that product onto the product details page and show it dynamically.

This way you would be able to maintain the information about your products in one place (your database) and any layout changes would only need to be done once.

Designing your app

Why does good design matter?

Take Your Time

Taking the time to properly design your application before jumping into the implementation is crucial for long-term success.

Plan

Without a plan and a design for your application, there's a risk that you end up with something unusable and unmaintainable.

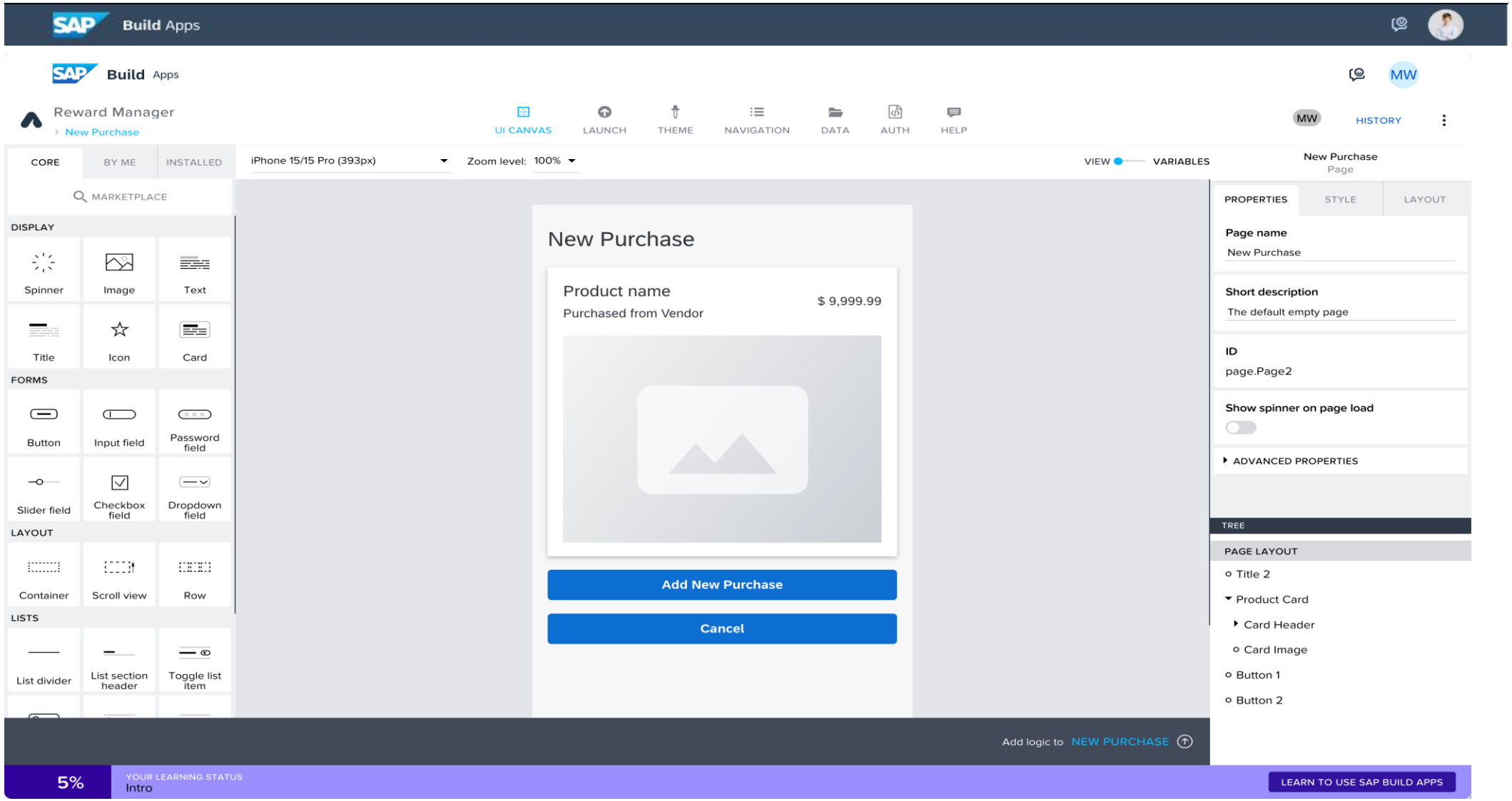
Best practices for designing your app

Gather requirements from potential users and plan a rough structure for your application.

- What pages are needed, what happens on each page, how does the user move through the app?
- You might study existing applications that do similar things and note their design and flow for ideas for your own application

Once you have pages planned, make sure your app is not growing too large.

- Too many pages will likely cause usability and performance issues.
- Good rule of thumb: If you have much **more than 20 pages** planned, it is likely that you are trying to do too many things in one app.
- Use **dynamic** rather than **static** content. If you still have more than 20 pages, consider reducing features or splitting functionality into several applications.


SAP Build Apps
Build Apps
Reward Manager
New Purchase
UI CANVAS
LAUNCH
THEME
NAVIGATION
DATA
AUTH
HELP
MW
HISTORY
CORE
BY ME
INSTALLED
MARKETPLACE
iPhone 15/15 Pro (393px)
Zoom level: 100%
VIEW
VARIABLES
New Purchase
Page
DISPLAY
Spinner
Image
Text
Title
Icon
Card
FORMS
Button
Input field
Password field
Slider field
Checkbox field
Dropdown field
LAYOUT
Container
Scroll view
Row
LISTS
List divider
List section header
Toggle list item
New Purchase
Product name
Purchased from Vendor
$ 9,999.99
Add New Purchase
Cancel
PROPERTIES
STYLE
LAYOUT
Page name
New Purchase
Short description
The default empty page
ID
page.Page2
Show spinner on page load
ADVANCED PROPERTIES
TREE
PAGE LAYOUT
Title 2
Product Card
Card Header
Card Image
Button 1
Button 2
Add logic to NEW PURCHASE
5%
YOUR LEARNING STATUS
Intro
LEARN TO USE SAP BUILD APPS

Keep in mind: App development isn't a linear process. Users may discover new requirements that you weren't previously aware of. It's a good idea to keep validating your designs with your users so that you can make small adjustments early in the process instead of major changes later.

Whenever there are some new requirements, evaluate their importance: which are crucial and which would be nice to have, but the app would still be functional without. Once this is done, check that SAP Build Apps still meets the crucial requirements of your app.

Developing your app

Best practices for development with SAP Build Apps

In the previous sections, we discussed app development from a high level. Now we'll dive into how to put those topics into practice using:

- Variables, application state and scope
- Navigation in mobile apps
- Logic
- Error handling and debugging
- User interface

Variables: Crucial for dynamic content

Variables are values that change based on logic or user input. Understanding how to use variables will improve the development experience and the app's overall maintainability.

Variety

SAP Build Apps offer various types of variables: data, theme, translation, system, page parameters, and component's internal variables

Pre-defined Pre-Populated

Some types of variables come pre-defined and pre-populated, such as system variables with information about the user's device and theme variables with information of how the current theme affects all the shown components.

Purpose

Some variable types have very specific purposes – i.e. a **translation variable** contains text content in various languages and a **page parameter** is used to pass a value, usually an identifier for data, between two pages.

Pro-tip: Naming variables descriptively enhances app maintainability. For example, if you have a variable that determines if a spinner is visible, naming it **showSpinner** will make it easier to understand what it's used for.

Variables: Key for managing app state & scope

The purpose of variables is to store the **state of the application**, which contains all information the app has available on the end-user's device and what happens in the app while it is used.

Distinguish

Distinguish between **app and page variables**, using app variables across all pages and restricting page variables to individual pages.

Scope

This defines the **scope** that the information is needed in, either on the scope of a page or of the whole app.

Determination

Over-reliance on app variables can lead to confusion and inefficiency. Determine which information isn't needed on every page.

Pro-tip: Instead of creating separate app variables, consider storing information in an Object or List type app variable, which can hold multiple properties. Their settings are more complex than a simple variable such as Text, Number or True/False.

Navigation in Mobile Apps

Effective **navigation** is crucial for user experience in applications with multiple pages.

Effect

Consider the effect of navigation on **page variables**, especially in scenarios involving back-and-forth navigation or opening the same page multiple times

Familiarity

Familiarity with the **navigation stack concept** is necessary to comprehend variable behavior in different navigation scenarios

- A navigation stack can best be described as a pile of pages. Example: The user enters the app on page A, then taps a button to open page B, e.g. a list of expenses.
- Pages in the navigation stack retain their existing state, affecting variable values.

Pro-tip: Pay attention to **events triggered** when opening or returning to pages for app logic. Opening a page in the stack multiple times creates separate instances of page and data variables for each instance.

Logic

Your application's **logic** defines how your app works and reacts to being interacted with. Here are some best practices for implementing logic that will make the app more maintainable and less prone to errors

Easy to find

Placing logic in relevant and easy-to-find places

DRY

Using events to follow the DRY principle

Error Handling

Implementing error handling

Pro-tip: The type of variables needed and where to place logic events depends on whether you need to use the same logic on one page or several pages. For example, you should always place reusable logic on a page's main logic canvas.

Error Handling & Debugging

When implementing logic in your app, the best practice is to take any possible error states into account as you go. Proper **error handling** is essential for user understanding and data consistency in app development.

Debugging is the process of troubleshooting app logic when it's not working as expected. Here are some best practices to follow for error handling debugging:

Tools

Use your browser's **developer tools** to diagnose problems and error messages.

Debug Logs

The pre-built Debug logic function in SAP Build Apps also creates custom **debug logs** that can help with debugging complex logic and increasing maintainability.

Logic

The debugging logic function can produce different types of logs (info, warn, error, etc.)

User Interfaces

A basic understanding about **UX/UI design or design thinking** concepts can be valuable in creating user interfaces in SAP Build Apps. Here are some best practices to follow

Maintainability

Whenever you make changes, consider the **long-term maintainability and consistency** of your application.

Theme & Style

To keep your app maintainable and consistent, utilize **theme variables** and **style classes** appropriately in SAP Build Apps.

Components

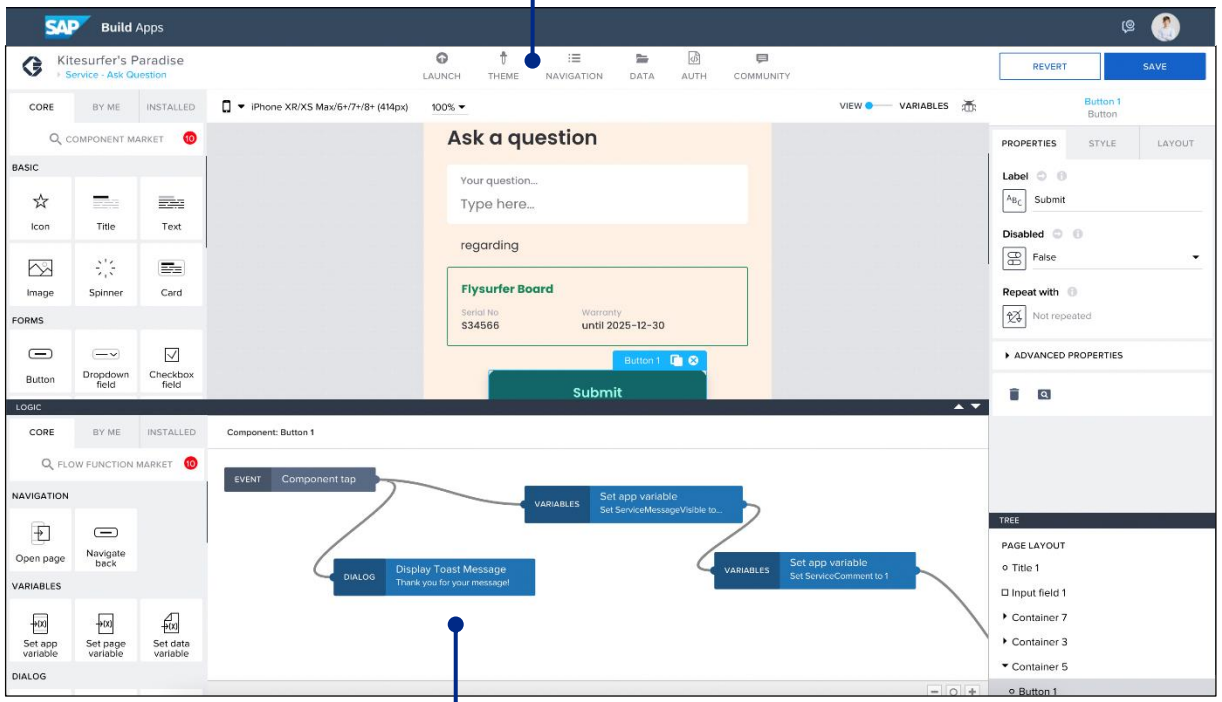
Primitive components and **composite components** are the two types used to construct UIs.

Pro-tip: UI building is quicker by using existing components than building everything from scratch with primitive components. Creating your own composite components is also possible, though it's recommended for more advanced users.

Powerful features for full-stack no-code development

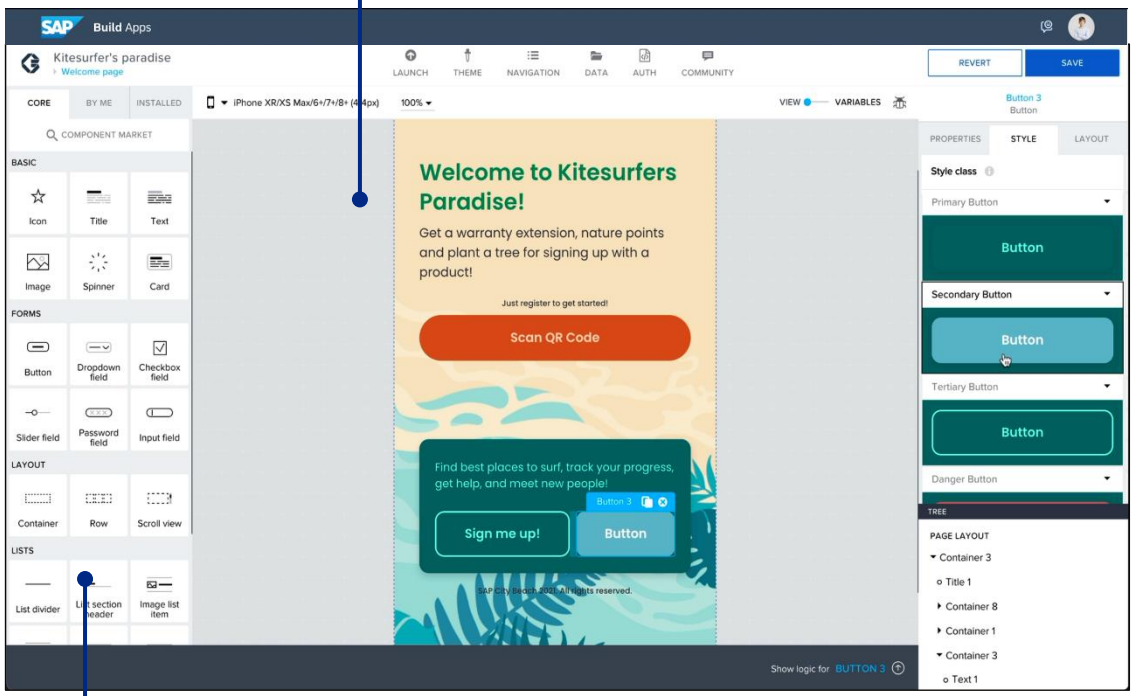
Key frontend capabilities

Customizable design themes and navigation



Visual application logic with flow functions

Simplified UI creation with drag-and-drop functionality



Rich library of components and 400+ built-in formula functions

Collaboration & teamwork

Why does teamwork matter in SAP Build Apps?

Maintainability is Key

When developing an application that is intended to be used for a long time, its maintainability is key. Even if you're working on an app alone, in the long run other members of your team will maintain the app as well.

Collaborate

In addition to getting your technical requirements right, it’s essential to design an app that you can collaborate on with your team.

Best Practices for Teamwork & Collaboration

Here are some tips for enabling multiple developers to work on one application efficiently

Naming

Decide on the **naming conventions** that you'll use in your app for variables and components. Use descriptive names, such as “Submit button” or “Header container” or “showSpinner” (i.e. using camel case for variables).

Shared Place

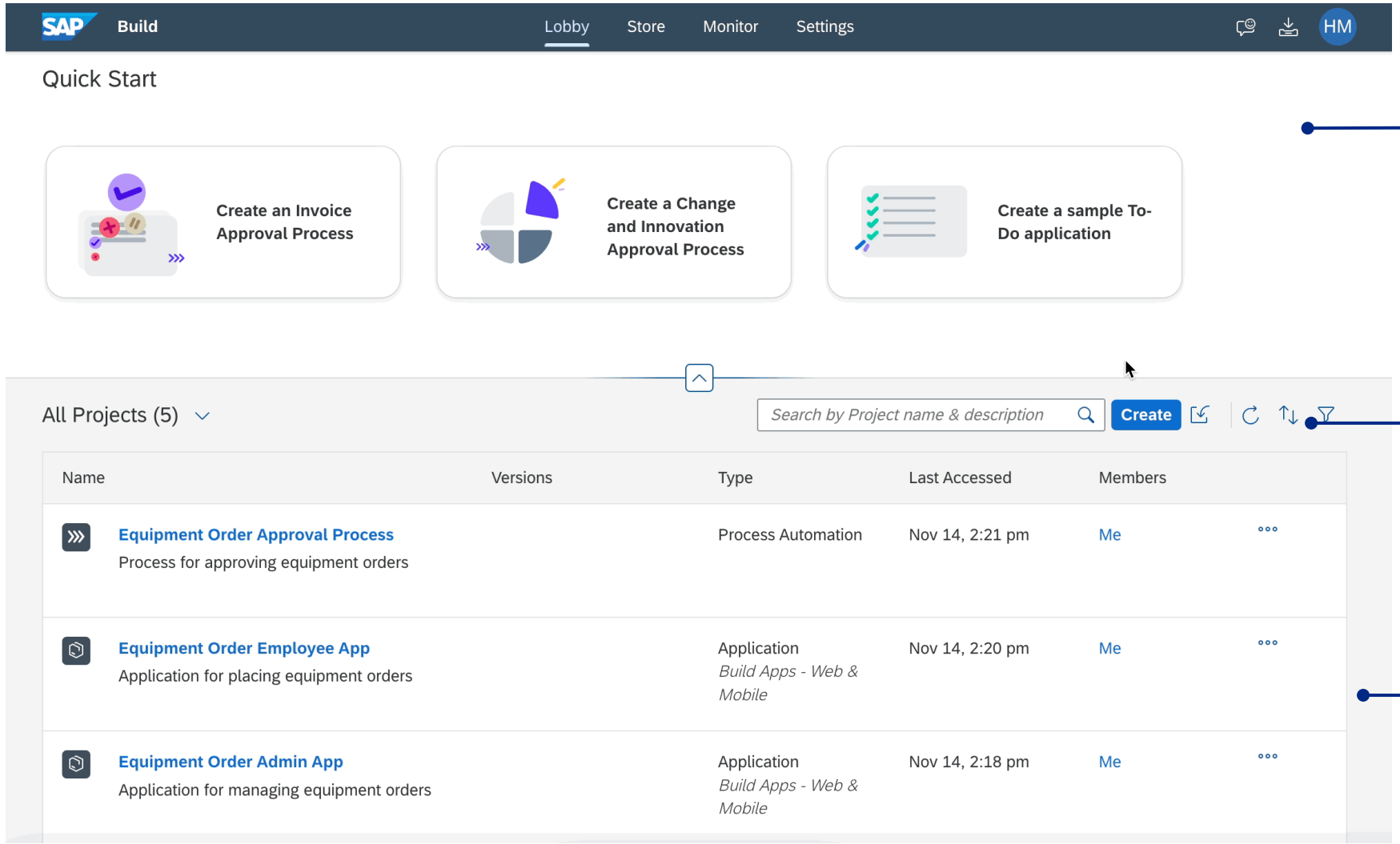
Establish a **shared place** where you make notes and share information about the application’s on-going development, any known issues or blockers, and any work-arounds implemented. Use this place to share original designs and plans for the app, decisions made on naming conventions, etc

Principles

Make sure that everyone is familiar and agrees with the **principles of good application development**. Check-in regularly with team members and keep the place where you share information up to date.

Powerful features for full-stack no-code development

Visual user experience from the start



Easy access to all SAP Build projects

Create and collaborate on projects

Centralized lifecycle management, monitoring, and governance

Q&A

SAP.com/Build-Apps

SAP Build Apps

Thank you.

Mark Wright Director Product Marketing – App Dev, Business Technology Platform
mark.wright@sap.com

SAPinsider

SAPinsider.org

PO Box 982Hampstead, NH 03841

SAPinsider comprises the largest and fastest growing SAP membership group with more than 800,000 members worldwide.