VIEW POINT

# SOFTWARE WHITE LABELING: PATTERNS & BEST PRACTICES

Navigate your next

## Preface

With multiple corporations working in multiple countries, multiple geographical locations, it has become an immediate need to provide experience which suits the local consumers' preferences, tastes etc. There is also a need to work with local resellers to increase the market share quickly with a low cost of ownership. These forces are the motivation to provide white label applications.

## What is White Labeling?

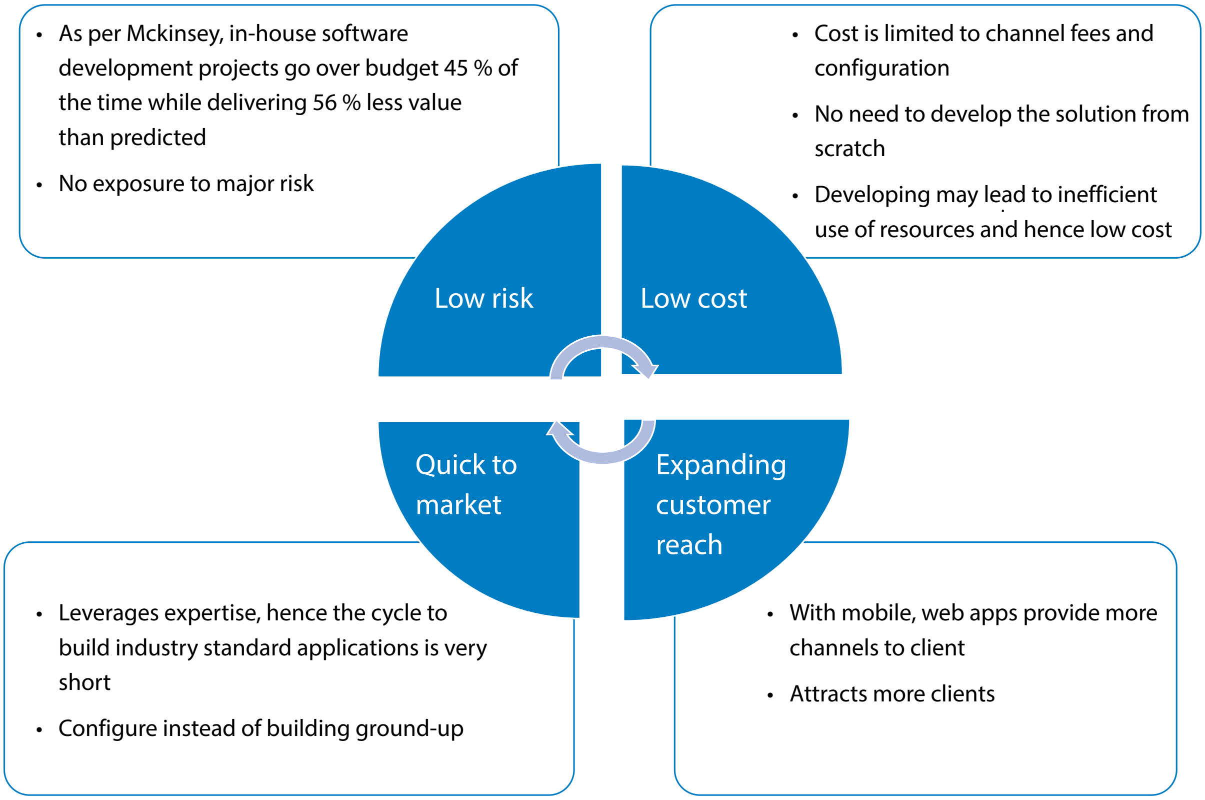
A white-label product is a product or service produced by one company (the producer) that other companies (the marketers, the resellers, the retailers) rebrand to make it appear as if they had made it. The name derives from the image of a white label on the packaging that can be filled in with the marketer's trade dress. White label products are sold by retailers with their own trademark but the products themselves are manufactured by a third party.

At times, the producer lets its reseller or partners put their brand on the product, thereby helping resellers to reach out to its consumer more effectively.

This in turn helps producers to improve their share in the market. This will benefit the parties i.e. producer as well as it's reseller.

It is important to note that while white labeling offers numerous advantages, it's essential to choose a reputable software provider that aligns with your business requirements and values. Conducting thorough due diligence, considering factors such as reliability, scalability, support, and security, will help ensure a successful white labeling partnership.

**Benefits:**



Low risk

- As per Mckinsey, in-house software development projects go over budget 45 % of the time while delivering 56 % less value than predicted
- No exposure to major risk

Low cost

- Cost is limited to channel fees and configuration
- No need to develop the solution from scratch
- Developing may lead to inefficient use of resources and hence low cost

Quick to market

- Leverages expertise, hence the cycle to build industry standard applications is very short
- Configure instead of building ground-up

Expanding customer reach

- With mobile, web apps provide more channels to client
- Attracts more clients

## Branding

Let's try and understand what the few major enablers are to brand the application and reflect the company's intended brand.
Here are a few, with details about them. If we try to narrow it down only to the digital presence, then the visual identity would consist of the following two blocks.

**Elements of basic design:** Basic design includes typefaces / typography, colors, layouts, grids, etc. These are the basic elements to create an impactful digital presence. Colors, fonts help to tell brand's persona to end users. Font impacts the experience, builds perception about the brand etc. It should work on every platform, be unique, memorable and legible. Selecting the right fonts plays a critical role in branding the company. It impacts the experience, builds perception about the brand and more.

**Visual Components:** Common forms of visual content include pictures, diagrams, charts, infographics, online videos, screenshots, memes, and slide decks used mainly for marketing purpose.

**Logo:** It is the visual representation of the brand. The logo is the element that has the power to stick in people's minds.

**Content:** Mission Statements, about US, who we are etc. Company may want to change the contents such as mission statement, about us etc. to reflect its culture, who they are, what they do etc. So there will be a need to provide the facility to change certain content as per the company's preferences.

## Accessibility

Changing the visual elements may impact the accessibility compliance of the product. There are many types of accessibility standards.

Accessibility guidelines are a reflection of various factors such as

1. Industry in which it is being used,
2. Product type,
3. Laws, compliance, policies,
4. Overall accessibility goals etc.

If no specific standard is mentioned, then Web Content Accessibility guidelines can be followed. This is a universally accepted standard.
**Tech Enablers:** So, the point is, when any company tries to use the white label application, they will be interested in changing the above mentioned elements to match to their current brand personality. And hence when we develop the solution, we will need to be mindful of enabling the companies to alter these elements considering the product and approaches you choose to build and provide the white label product.

**Few main building blocks:** Here is a list of a few main building blocks to build a white label solution.

Depending upon your business case, you may need many more and may have many other

use cases to use these. So while choosing these blocks, please make sure it is time and cost efficient.
My intent is to highlight a few possible components. Depending on your use cases, you may want to find

alternate solutions e.g., for CMS, you can have them stored into blocks as well.

| | |
|---|---|
| Content Management System | This will be useful to store all the content, assets such as fonts, icons, logos, marketing flyers, product information etc. These will vary from one customer to another. |
| Database | To store the mapping of the configurations and associated themes. |
| Build System | Since there are multiple products to be built, you will need to develop efficient configuration based on a complex build system, which should bundle and package asset files, sources depending on the application being built. |
| Accessibility Testing Suite | Assuming there is a need to support it, you will need reliable tools to make sure the product is as per compliance. |

**Reference Architectures:**

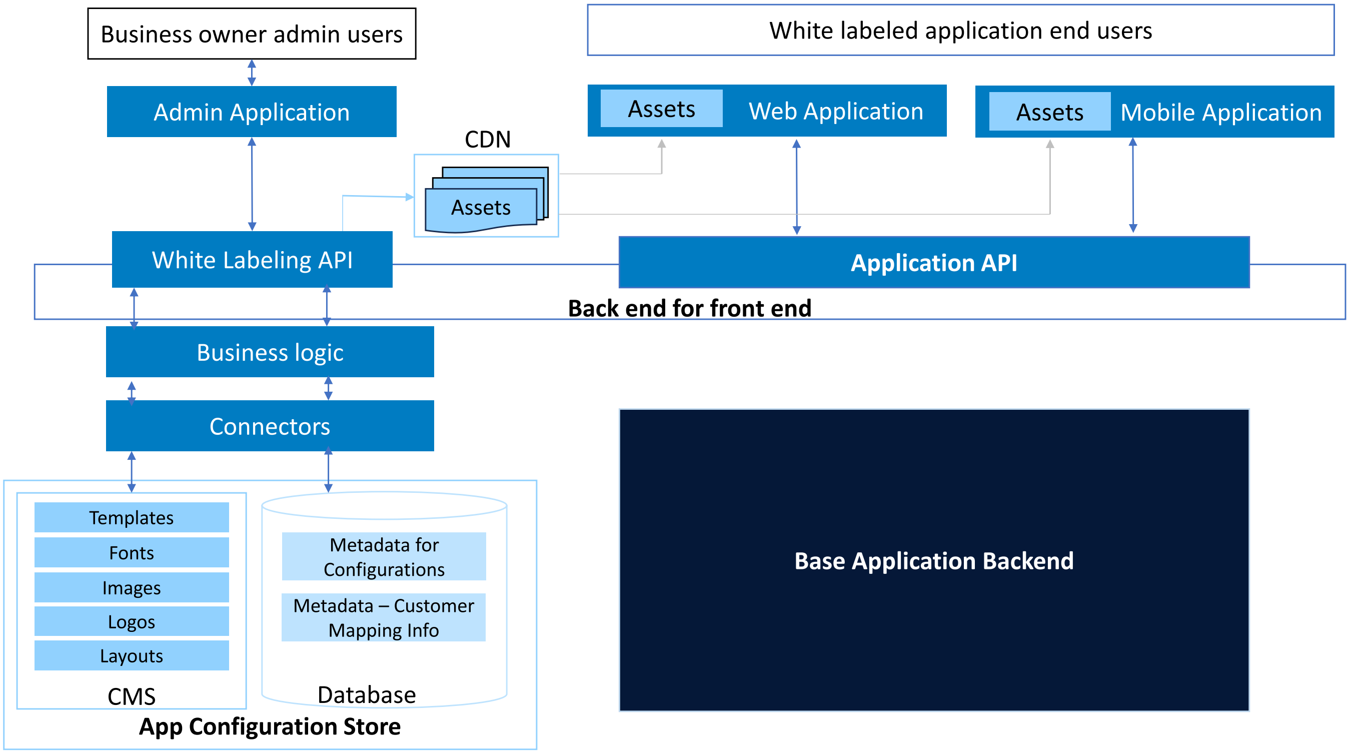
There are two ways to develop and distribute a white labelled application. One is to develop a dedicated built and deployed application – Single Tenant. Another is to build it once and deploy one to the set of servers (multi-tenant).

Here is reference architecture for the same.

**Multi-Tenancy Application**

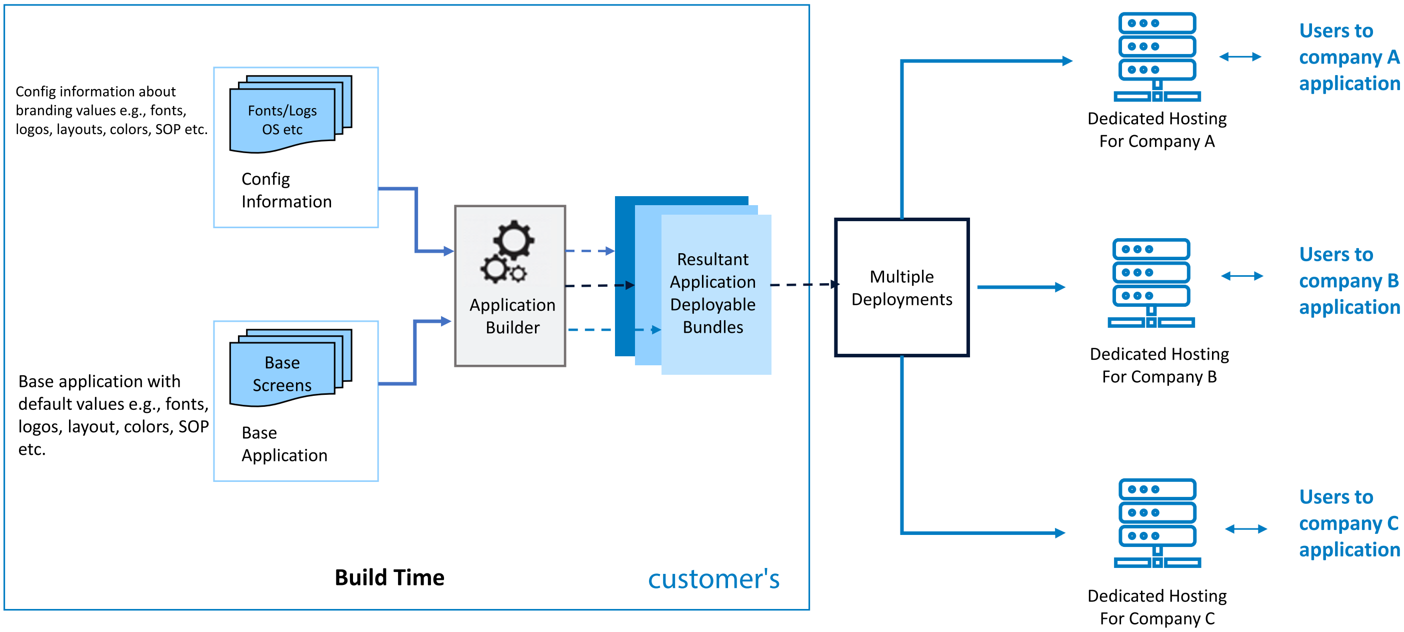
- Admin application to let white label using companies configure application to suit their branding guidelines
- White Labeling APIs to let Admin application interact with configuration stores
- An application API interacts with its business application backends for channel data, logic etc.
- An application API interacts with White Labeling APIs to fetch meta data for configurations store per users intended customer configured application. These APIs configure the end user experience as per the customers branding.
- Application APIs should be developed as query-based APIs such as Graph QL. This will help to develop the APIs which will map to most of the needs for various customers considering the fact that, end user experience may vary from one company to another.
- CMS is the best option to store all the configurable assets such as logos, templates, fonts etc. This will provide out-of-the-box support for version, scaling, authorizing etc.
- Database to keep a mapping of these configurable mappings to the companies using the white label application.

- It is important to keep configurable assets closure to the front-end application. CDN, multi-level caching strategy should be implemented. This will make sure there is very minimal delay in configuring front end application



**Single Tenant Application:**

- White Label Application customer organization communicates the desired changes to white label solution provider.
- Solution provider will create one dedicated build system to build the separate application and deploys/publishes to server.
- Post deployment, the solution provider communicated to the partner to start using the application.
- Solution provider will need to work out the deployment details such as server capacity, location etc. depending on the customer's preferences.
- This approach gives a lot of flexibility to the customer as well as the solution provider.
- It will have higher cost of ownership.
- Since the build, deployment is dedicated, time to market may be slightly longer than multi-tenant approach.



Config information about branding values e.g., fonts, logos, layouts, colors, SOP etc.
Fonts/Logs OS etc
Config Information
Base application with default values e.g., fonts, logos, layout, colors, SOP etc.
Base Screens
Base Application
Application Builder
Resultant Application Deployable Bundles
Build Time
customer's
Multiple Deployments
Dedicated Hosting For Company A
Users to company A application
Dedicated Hosting For Company B
Users to company B application
Dedicated Hosting For Company C
Users to company C application

**Build Systems:**

As noted earlier, build systems play a vital role in building the product flavor application to bundle the applications as per customer's customized requirement. Here are references to three prominent use cases.

**Web Application:**

All the neatly designed web applications will have dedicated folders to store the assets. Projects can have multiple subfolders to store application specific assets into these subfolders. Once that is achieved, one needs to configure the bundlers to pick up the assets and files to bundle those and create multiple versions of the applications. Most of the web applications are transpiled, bundled using module bundling. On a high level, this is a process of integrating a group of modules in a single file so that multiple modules can be sent to the browser in a single bundle. There are a lot of module bundlers in the market today like webpack, browserify, rollup, parcel etc. Each of these has ways to configure which assets, which source files etc. to pick and build the final application to be deployed on the server.

**Android Mobile Application:**

The Android build system compiles app resources and source code and packages them into APKs or Android App Bundles that one can test, deploy, sign, and distribute.  Android Studio uses Gradle, to automate and manage the build process while letting you define flexible, custom build configurations. Gradle and the Android Gradle plugin help to configure product flavors and other aspects. Product flavors represent different versions of your app that you can release to users, such as free and paid versions. You can customize product flavors to use different code and resources while sharing and reusing the parts that are common to all versions of the application. This can be leveraged to build various white labelled applications.

**iPhone iOS Mobile Application:**

iOS systems use targets to build applications. A target specifies a product to build and contains the instructions for building the product from a set of files in a project or workspace Use Targets to build multiple white labelled applications.  A target specifies a product to build and contains the instructions for building the product from a set of files in a project or workspace. For each application, create dedicated assets folder to store assets such as Fonts, Images, App Icon, Color definitions/declaration. One can add target specific Key and Value into Info.plist file. Add build configurations under Info to make sure, there are application specific build instructions.

**Single Tenant Vs Multi-Tenant Application Approach:**

The main difference between them is that these software applications can serve either one or more customers at the same time. As the name suggests, Single Tenant application serves only one customer unlike multi-tenant. Both have their advantages and disadvatanges. Here is quick snapshot of the same.

| Parameters | Single Tenant | Multi Tenant |
|---|---|---|
| Cost efficient | ★ ★ | ★ ★ ★ ★ |
| Time to market | ★ ★ | ★ ★ ★ ★ |
| Reliability | ★ ★ ★ ★ | ★ ★ ★ |
| Performance | ★ ★ ★ ★ | ★ ★ ★ |
| Security | ★ ★ ★ ★ | ★ ★ ★ |
| Setup complexity | ★ ★ | ★ ★ ★ ★ |
| Flexibility | ★ ★ ★ ★ | ★ ★ |
| Efficient resource usage | ★ ★ | ★ ★ ★ ★ |

## Use cases

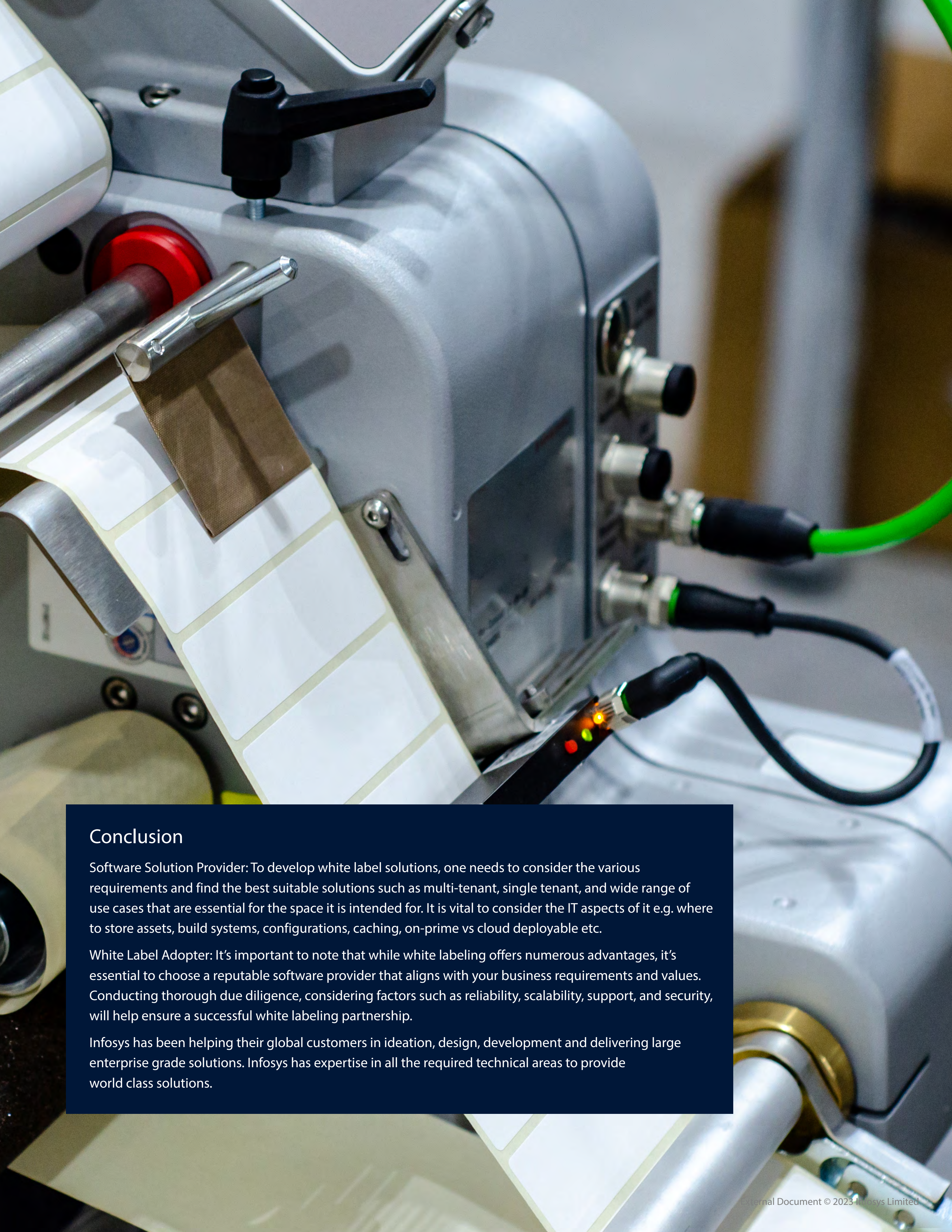
There are multiple motivations to create white label applications. Some of the prominent ones are,

a. Corporations which have a franchise business model, will let the franchise owners/resellers have the industry standard software infrastructre at low / no cost. This will help the corporations retain loyalty from franchise owners and they will get industry standard, cutting edge applications to reach out to their own customers and serve them better. This benefits both the parties.

b. Corporations which have entered into multiple geographical markets would like to sell their multiple brands in accordance with local preferences. White labeling helps use the same application source code but with a flavor of local preferences such as language, logos, marketing material etc.

**SaaS and White Labeling:** Software as a Service (SaaS) and white labeling are closely related concepts in the software industry. SaaS is a term used to describe a software delivery paradigm in which users can access hosted software applications online. On the other side, white labelling is rebranding a software product or service created by one firm and making it available under the brand of another company.

White labeling is a beneficial strategy for SaaS providers, as it allows other businesses to offer their software under their own branding and take advantage of their infrastructure and technology stack. It also enables customization and branding, reseller partnerships, value-added services, and scalability and efficiency. White labeling enables SaaS providers to differentiate their offering and create additional revenue streams, while also allowing them to scale their business efficiently.



## Conclusion

Software Solution Provider: To develop white label solutions, one needs to consider the various requirements and find the best suitable solutions such as multi-tenant, single tenant, and wide range of use cases that are essential for the space it is intended for. It is vital to consider the IT aspects of it e.g. where to store assets, build systems, configurations, caching, on-prime vs cloud deployable etc.

White Label Adopter: It's important to note that while white labeling offers numerous advantages, it's essential to choose a reputable software provider that aligns with your business requirements and values. Conducting thorough due diligence, considering factors such as reliability, scalability, support, and security, will help ensure a successful white labeling partnership.

Infosys has been helping their global customers in ideation, design, development and delivering large enterprise grade solutions. Infosys has expertise in all the required technical areas to provide world class solutions.

## Author

### Rahul Sale

Principle Technology Architect

Rahul is an accomplished Principal Technology Architect, with focus area in Digital Transformation. He is with the Digital Experience Unit at Infosys. He brings over two decades of experience spearheading digital transformative initiatives for leading organizations. Proficient in aligning technology strategies with business objectives, driving innovation, and optimizing digital ecosystems to achieve sustainable growth. Demonstrated leadership in designing scalable architectures, leveraging emerging technologies, and empowering businesses to thrive in the digital age.

For more information, contact askus@infosys.com

Infosys®
Navigate your next