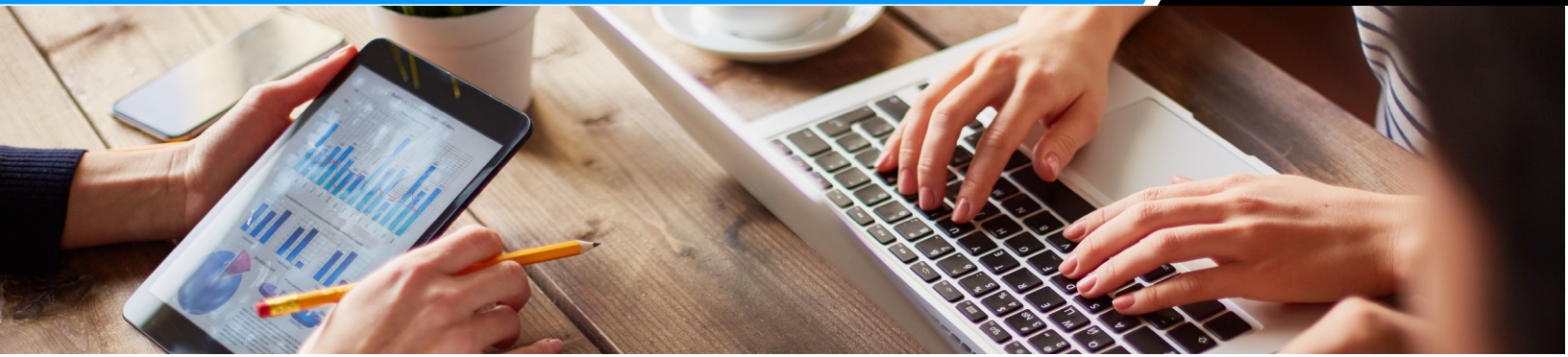




# Desktop to Web Application Migration

Framework to guide enterprise development teams.

Introduction	1
Vision Phase	3
Analysis Phase	4
Design Phase	10
Implementation Phase	14
Summary	16



# Introduction

The high cost of desktop application support and the sudden mandate to support a wide variety of devices are some of the compelling business drivers forcing enterprises to migrate applications from the desktop to web.

The concerns are not new and have been gaining in significance over the past decade. Web technologies did not exist to support the migration of complex enterprise applications from traditional applications that run on a Windows PC to applications that can run on virtually any computer or device. Fortunately, leaders of the technical community identified the need to rethink the web years ago. With various companies and organizations within the technical community working vigorously to address such concerns, we are finally at a state of technological advancement where we can confidently migrate complex applications from the desktop to the web.

One major opportunity of desktop-to-web migrations is the chance to modernize and optimize the user experience (UX). The typical highly complex business application running on a desktop in an enterprise has been in existence for many years as these types of products have longer lifecycles than products in the consumer market. Consequently, the need for UI improvements that drive gains in productivity, usability and user satisfaction has accumulated – now as code has to be touched in the course of a migration project, it's a perfect opportunity to modernize the user experience and leverage the power of modern web platforms.

The purpose of this framework is to provide concrete guidance to IT management and software development leads in migrating applications from desktop to web technologies. The phases of this framework should be followed in order.

The purpose of this framework is to provide concrete guidance to IT management and software development leads in migrating applications from desktop to web technologies. The phases of this framework should be followed in order.

## Audience

This document is intended to be read by IT executives who are responsible for driving transition from desktop to modern web experiences.

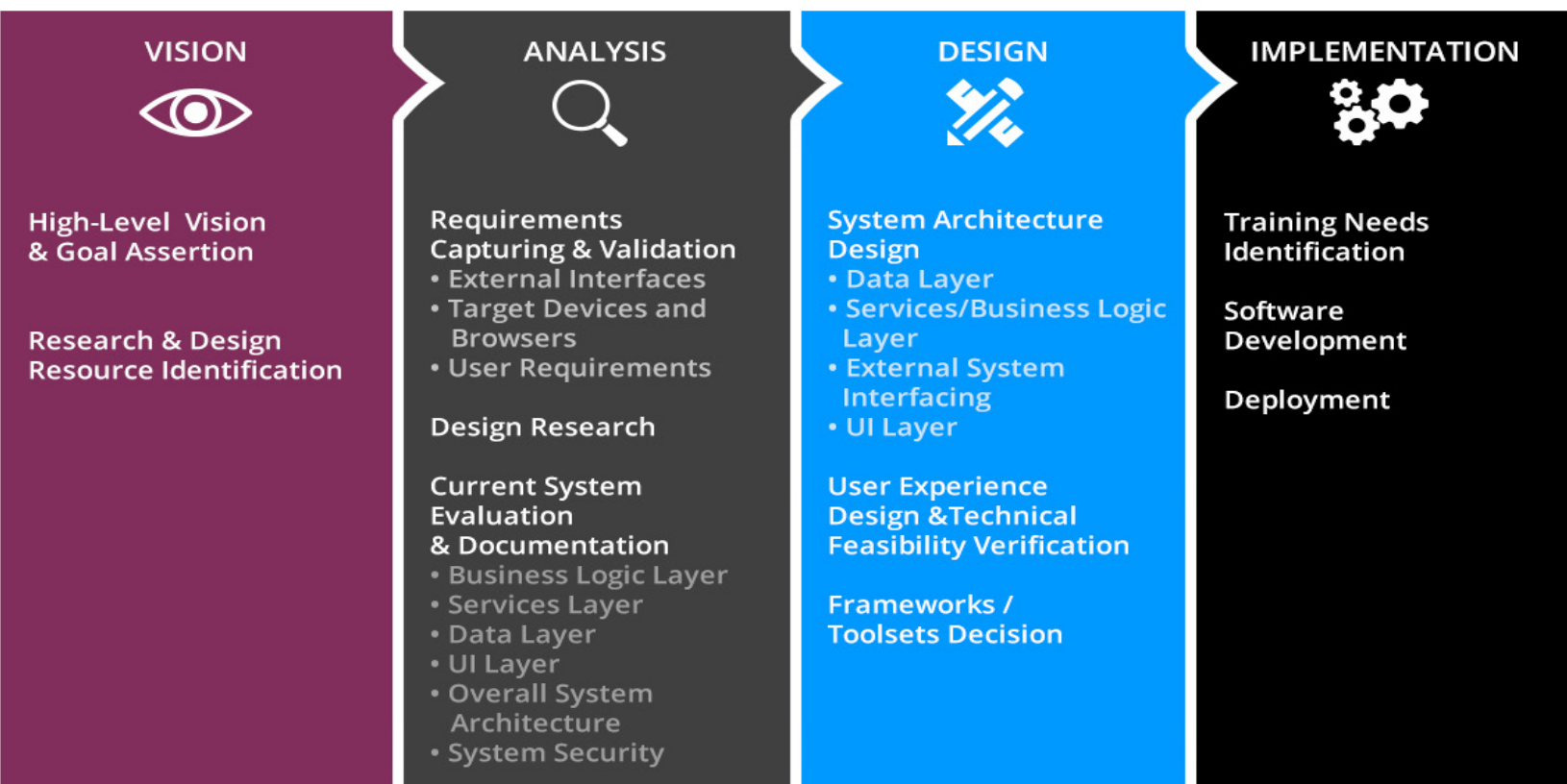
## Scope

This framework focuses on Line of Business (LOB) applications. Applications in other categories such as gaming, consumer, and entertainment are out of scope. This framework can be applied to any software development methodology such as Agile or Waterfall. So actual implementation techniques will not be discussed.

## Phases

This framework breaks up the migration process into 4 distinct phases.

The steps and output of each phase will be described in detail.







# Vision Phase

The purpose of the Vision Phase is to develop high level goals and identify resources that will be required for the project.

## High-Level Vision/Goal Assertion

The Vision Phase begins with goal identification. It is critical to understand, enumerate and articulate the vision & goals of the migration to the entire team and stakeholders. This step ensures that each member of the team clearly understands the objectives of the migration and aligns their decisions around them. These high-level goals and vision statements will guide both the UX and development of the entire project, so they should be open-ended enough to allow for innovation based on design research as well as scoped enough to give the UX professionals a good place to start.

It is important that the goals be focused on the business drivers rather than technology.

### The following are sample goals:

- Enable underwriters with the ability to process loans while working on their tablets out of the office.
- Support the firm's BYOD policy by providing Board Members with the ability to approve X process while traveling.
- Enable sales representatives to enter customer contact data faster than before.

## Research & Design Resource Identification

Typically, at this stage of the process, you need to identify the resources you believe should be available for the project. As with the high-level vision, this may be an incomplete picture at this stage, but it is an important step. A successful migration will require the time and knowledge of these external resources, and resources outside the development team are especially needed during the research and design phases, so it is important to identify them up front.

## Engaging a Third Party Firm

During this resource identification step, it is important to ensure that there are resources on the team that are experienced with desktop to web migrations as well as both UX/Design and modern Web technology.

**If you do not have this expertise, Infragistics is ready to partner with you and provide:**

- Leading experts in User Experience who can guide you through research and design.
- Deeply experienced architects who can provide insights for desktop to modern Web migrations.
- World-class, modern Web design and development tools and components.
- Training/mentoring in-house developers on modern Web technology.



# Analysis Phase

During the analysis phase business requirements will be enumerated and the current system will be dissected.

## Capture/Validate Requirements

A key part of creating any new software system is identifying business requirements and ensuring they are met. Moving to a new technology does not negate the need to do this, although it can greatly help speed the process up. If you have gone through the above steps of inspecting and documenting the current system, that provides a great jumpstart towards this effort when moving to the modern Web. As noted above, it is highly recommended that in addition to this kind of as-is documentation and rationale discovery that you do design research to discover in an unbiased way the current needs and even uncover previously latent or hidden needs that were not being met with the current system.

Taking the basis of the current system and the results of the design research, you can validate, update, and improve the design of the system to ensure it is most effectively meeting business requirements—and what better time to do that than when investing in a new technology? Why move forward old, broken, no-longer-valid business requirements?

### Identify External Interface Requirements



In addition to current business requirements for the application itself, it is almost certain that there are and will continue to be interfaces with other systems in your environment. Every system that the application must interface with must be listed. For example, a loan origination system must interface with credit bureaus and third party underwriting systems. External interfaces may also include internal applications that the application has dependencies on (General Ledger, CRM, etc.).

### Target Device/Browser Requirements



Any hard requirements pertaining to devices or browsers must be noted. Client devices/browsers have varying functionality. Of course, with modern Web, one of the key goals is to be “future proof” such that the application doesn’t break just because there is a new Android device on the market. This is usually done through a combination of feature detection and the use of polyfills when newer browser features are not available. You can’t

be 100% certain, but if you follow current browser standards, use feature detection, and use polyfills, you can get a much higher degree of confidence that the application will work on both existing and future devices.

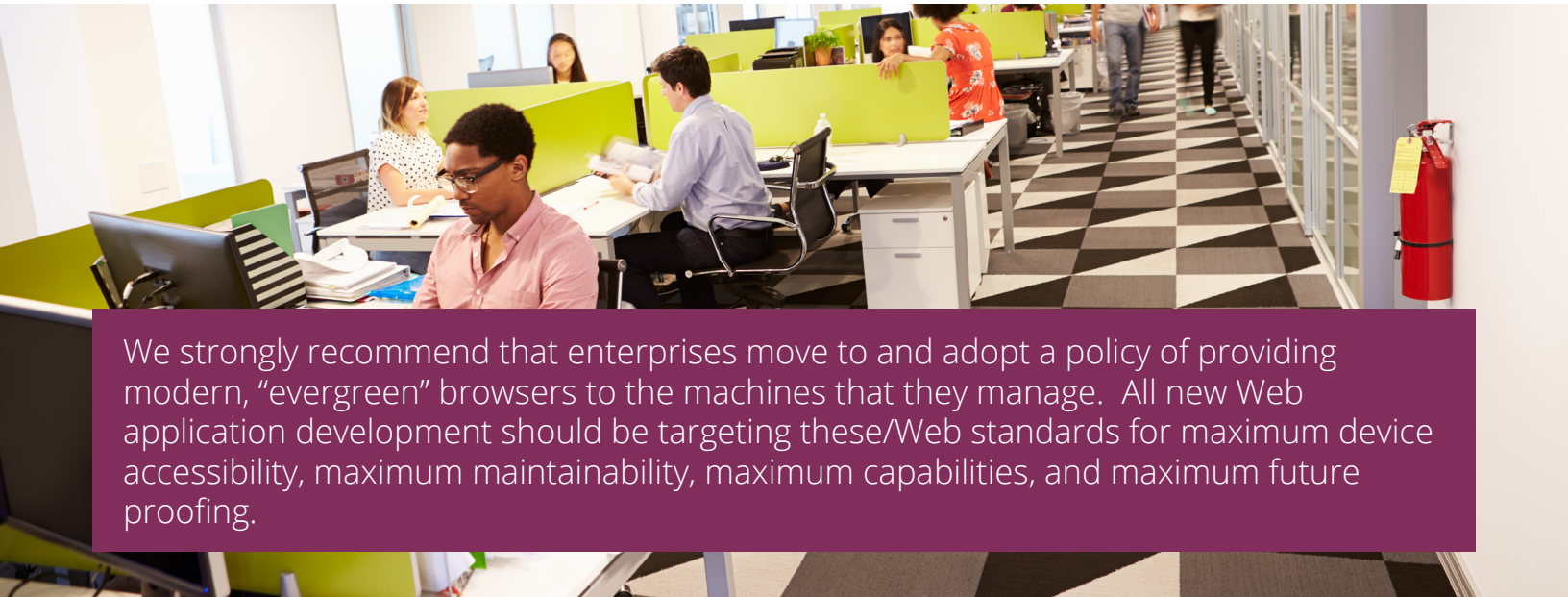
Where identifying known target devices comes in most handy is in the ability to potentially optimize the design for those devices as well as providing concrete parameters around testing—you typically can't test on every possible device, so knowing which ones you definitely want to ensure it works well on makes the quality assurance aspect of modern Web applications much more manageable.



### No More “Only Works In IE 6/8/9/Whatever”!

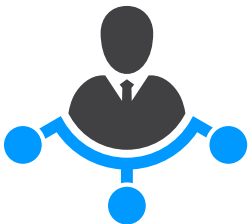
It is imperative to break away from the old notion of locking down browsers that was and often still is very prevalent in the enterprise space. There is nothing that has so handicapped the forward evolution of the Web platform than being required to support old versions of Internet Explorer. The evolution (that is, improvement) of the Web platform is just as important to enterprises as it is to consumer-oriented solutions—perhaps even more so. Many of the solutions that are in current and planned Web standards are specifically about making Web applications more powerful, more maintainable, and generally more like real, rich client applications.

And not only that, given that the most common business driver for moving to the modern Web is to support more devices, it is not even really an option to try to specify and lock down which version is used to access your application.



We strongly recommend that enterprises move to and adopt a policy of providing modern, “evergreen” browsers to the machines that they manage. All new Web application development should be targeting these Web standards for maximum device accessibility, maximum maintainability, maximum capabilities, and maximum future proofing.

### User Requirements



Although user needs will be elicited and analyzed in the next step (see “Design Research” below), it is beneficial at this point already to analyze the body of knowledge about user feedback towards the legacy desktop application. Typically, documentation or at least anecdotal evidence from the support desk about users’ complaints, suggestions and comments about the legacy product exists. This information is a vital input to the subsequent design research as it already identifies critical areas of the user interface that need to be investigated in more detail.




## Design Research

Once the business sets the initial vision, the next step is to engage Design/UX professionals to begin design research. This can also be known as “user research,” but “design research” is more encompassing of not just user but also other aspects that may inform the design such as business requirements, competitive/vendor evaluations, technical constraints, and in the case of moving from an existing system to a new one, deep validation and evaluation of the existing system against current needs. Often legacy systems contain a heavy mix of old solutions that don't quite fit current needs along with adaptations and workarounds. When moving a system to a new technology, it is prime time to re-evaluate it and ensure the new system is not just built on new technology but also best fits the current and anticipated future realities of the business and user needs. For example, a web-based system is accessible through a much wider range of devices and form factors than a desktop solution. How does the web solution have to present itself on a smart phone vs. a tablet vs. an office PC to be useful for its users? Design research identifies what workflows and thus what features map to the different device classes.

During this phase as research activities are completed, the design researchers will typically meet with business stakeholders to review the results of their research, clarify and expound on the vision based on the research, and potentially engage further research, depending on the findings.

This does not have to be a long and expensive process—the research goals, timeframe, and budget should be agreed upon up front and can be adapted to the real business situation. Especially for internal systems, design research iterations can be fast and less expensive than open/consumer-oriented efforts largely due to a well-defined user base and the availability of both users and affected parties to participate in the research.



**The outcome of the design research phase is an increased understanding of appropriate solutions through:**

- Validation of assumptions/known needs and often discovery of new, important needs that were not known.
- Increased clarity on who the users are, what their contexts of use are, and what their needs and goals are.

## Evaluating And Documenting The Current System

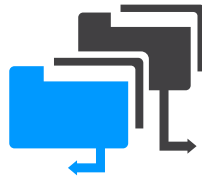
A key step in moving a system to a new technology is, of course, evaluating and documenting the current system architecture and behaviors. What follows are some concrete areas that should be examined during this process. As noted above, keep in mind that current behaviors/design may not be optimal for the current business and user needs and also, depending on the current architecture, existing desktop application architecture may need significant modification to fit a healthy Web architecture.



### Business Logic Layer

Many legacy applications contain business logic that resides in more than one location. Stored procedures and “decision engines” are examples of business logic that are outside of traditional code modules. It is important to both note the location and interface of these as well as the business rules and rationale for those rules, which can be used in validation against research with the business and users.

Typically, the business logic of a legacy application will be the most up to date and accurate and reusable part of the application architecture; however, it can be that even this aspect has suffered over time due to inflexibility of architecture that necessitated business/user-based workaround rather than system adaptations.



### Services Layer

If you are fortunate enough to have a well-defined services layer in your desktop application architecture, it can greatly contribute to the reusability of both your business and data layers. Whether it is a true Service-Oriented Architecture (SOA) or something like ReST or other simple, CRUD-like services, encapsulating within a service rather than, e.g., directly using class libraries or database connections, can greatly increase reusability likelihood.

The main things to watch out for at this layer, much like the Business Logic Layer as well, are if the services are up to date and map well to the business and user needs discovered through the design research. It can be that they have aged and no longer reflect that. But in any case, identifying the services, their locations, interfaces/contracts, and data format options are key to this step.



### Data Layer

The foundation of most business applications is its data, and this is also typically the least mutable layer in a system. This is true when migrating to new technologies as well—it’s far more common for the data layer to remain the same and to change business, services, and UI layers. In many cases, it is also the easiest to inspect and document, depending on the technology being used.

That said, it is not uncommon for business logic to be pushed into the data layer in, e.g., stored procedures and functions, and for a services layer to be completely bypassed (or non-existent) as in the common case of using ADO/ADO.NET or similar technology to directly connect and execute commands and queries against the database. If that is the case, moving to the modern Web will necessitate a new, or at least modified, architecture because it is impossible for browser-based applications to connect to databases this way.



It is also generally considered to be not a good practice, because we lose separation of concerns—business logic is distributed throughout the app in DB commands, on the database in stored procedures, which makes them more tightly coupled and harder to test, adapt, and scale. Moving to Web is a great opportunity to re-evaluate this approach and separate the layers more cleanly.



## UI Layer

This layer is, of course, the main reason a business would be looking to move from desktop to Web, and indeed, in general, it is by far the most mutable layer in a typical business application. It changes more both because of evolving user and business needs and because the platform technologies themselves change the most over time.

In terms of reusability, very little code itself would be reusable because of the underlying platform change from desktop to modern Web. It may be higher if you are moving from a legacy, server-side desktop-oriented Web application (e.g., one created in ASP.NET Web Forms) in that you could potentially port some JavaScript, but chances are that it will be minimal.

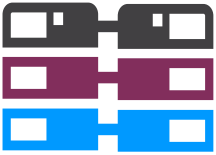
Still, there is a lot that could move forward from an interface design point of view—forms, validation, interaction flow, and UI patterns (as manifested in controls such as grids, charts, combos, etc.). It is very useful to document current interaction flows, forms design, UI elements, visual treatment, and any validation or other business logic contained in this layer. Its reusability largely depends on the UI design of the new system and how much is brought forward. We recommend re-evaluating this in the Design phase to ensure that the design maps best to current user and business needs.

One area where moving to the modern Web can really be different from an interface point of view is when dealing with mobile phones. We strongly recommend evaluating how much of your users' needs revolve around mobile phone use. It can be relatively straightforward to adapt a desktop interface to a responsive Web interface suitable for laptops, desktops, tablets, and similar devices, but phones tend to call for significantly different designs due to both form factor and contexts of use. Typically, businesses will decide to expose a subset of their full LOB applications on phones—but that decision should be made based on user research (as discussed above).

And if the research reveals the need for phone use, we typically recommend a custom phone application rather than trying to create one responsive Web app that does it all. Not only does this make the application complexity more manageable, but it also allows you to more feasibly optimize the interaction design for that phone usage scenarios, which can be significantly different from desktop/tablet. We also recommend considering native for mobile phone applications and using responsive Web for desktop/tablet device classes.

No matter what approach and architecture you choose, Infragistics offers controls and components that will help you be successful in this layer—both modern, responsive Web in Ignite UI and native mobile controls for iOS, Android, and Xamarin/Xamarin.Forms.

## Overall Current System Architecture



As you go through the different layers of your system, it is useful to build up an overall architectural view and construct an application architectural diagram. This always makes systems much more understandable, and it will help you to highlight and communicate key areas of change based on this evaluation and the design research. The use of UML or a similar diagramming approach is recommended to facilitate clear communication to the entire team.



## System Security

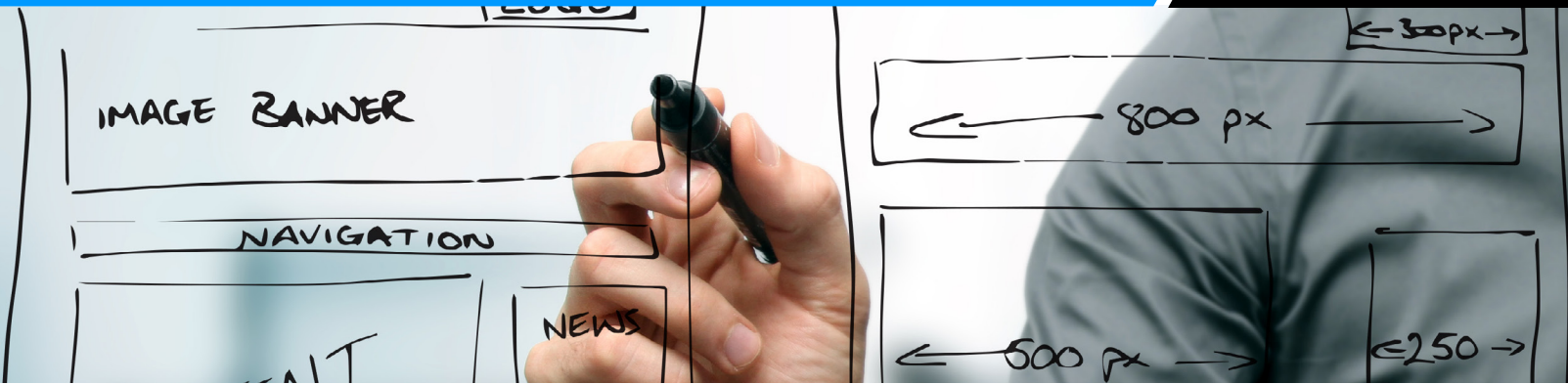
Authentication and authorization are critical areas of every enterprise application. It is important to identify security mechanisms early on. This is also often an area for improvement to consider when moving to new technologies. And when moving to the Web, in particular, there are additional security concerns.

Identify application level security concerns such as role/owner-based authorization and the kind of authentication in use. Most CRUD/LOB systems involve a combination of authorization techniques as well as varying degrees of granularity, and often these are intermixed with business logic.

The Web will necessitate a change in authentication. One of the key drivers for moving to modern, responsive Web is to make applications available on a variety of devices, many of which have no support for integrated authentication using Kerberos and the like. Authorization can also be trickier because you typically don't want to rely on client code (because it runs in the browser). So you will need to evaluate other options, such as OAuth2, which is currently the most popular protocol for the Web. Choosing the right flow and implementing it can be tricky.

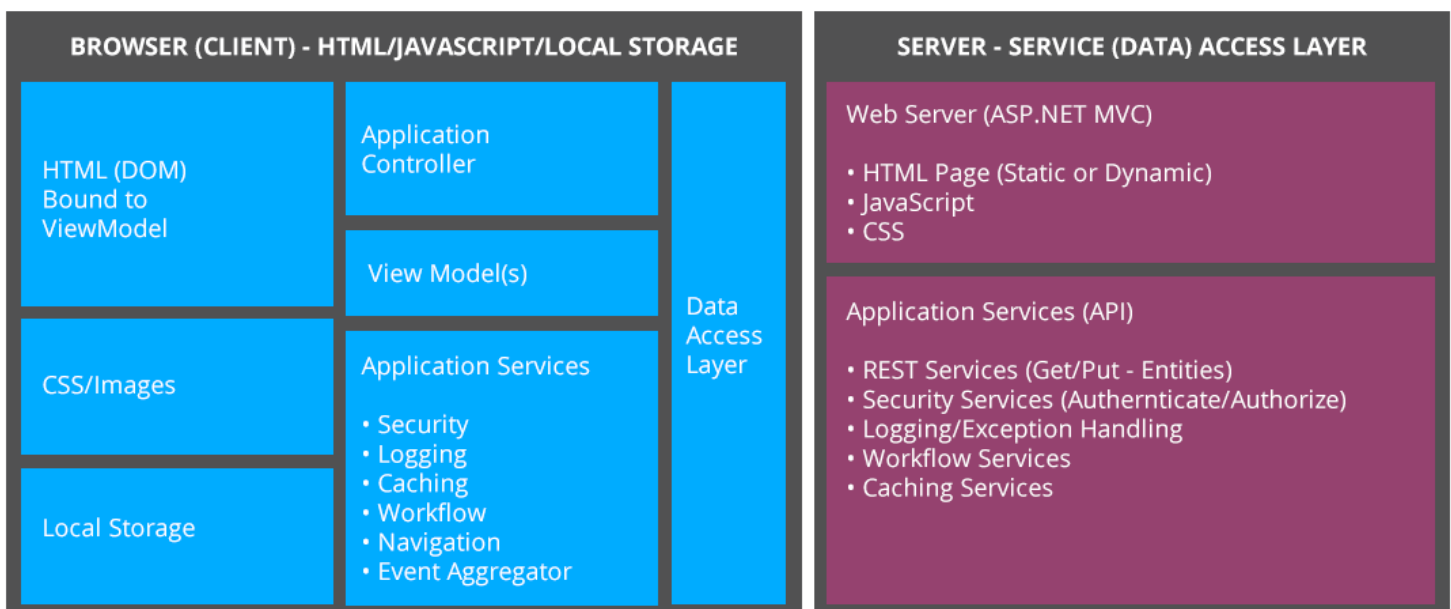
Additionally, you need to consider whether the application will be accessible from the internet or only on private networks. You will need to think about transport-level security using SSL, and you will need to address common Web application vulnerabilities such as cross-site scripting and SQL injection.





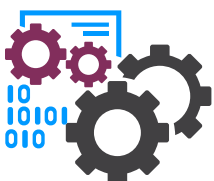
# Design Phase

The technical blueprint for the application will be constructed during the Design phase.



## System Architecture Design

Each layer of the system should be designed with practical reuse in mind. While certain components, and subsystems can technically be reused, careful consideration should be taken when deciding to reuse or build. For example, while certain legacy data services could be used, it may make more sense to build a service using newer technology that web apps can leverage more effectively.



### Data Layer

As noted above, this layer can have high reuse probability, but it may call for some changes to support the modern Web. If you were relying on calling stored procedures or directly executing commands and queries, you should definitely consider moving those out into services.



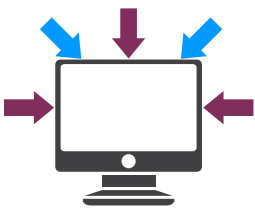


## Services/Business Logic Layer

Depending on how you design it, your services layer and business logic layer can be very interrelated. If you have a Service Oriented Architecture, there is strong guidance for this to be the case. But a ReST-based approach to services has become much more popular in recent years as a way to easily expose entity-related operations over HTTP. There are many modern Web tools that can take advantage of this, and it is up to you to decide if that makes sense or if you prefer a more SOA or RPC-style approach to your services.

No matter what, though, you will need to expose services over HTTP for them to be consumed in modern Web. If your experience is with .NET and/or ASP.NET, then ASP.NET Web API is a great choice. You can even leverage Entity Framework to easily expose ReST-based services through it.

You will also need to be thinking about service-level security, which typically involves some flavor of OAuth2.



## External System Interfacing

Determine which external interfaces may be reused and which must be built. If you need to surface the systems services/data to other apps, now is the time to be thinking about that as well. The good news is that if you do a ReST-based API, then it is highly reusable for other apps as well as the main UI client you are designing.



## UI Layer

This is where the rubber meets the road for modern Web, and this is where moving to modern Web from desktop causes the most trouble for developers who are new to it—there are so many choices! Your decisions should be based on UX and Interaction Design research. This will be covered in the next section.

# UX / Interaction Design

Once the design research—including current system and business evaluation—has been completed and reconciled, there should be an abundance of great information upon which to base the design of the new system. This is the point at which the project can transition from research/understanding into design.

As noted above, it is important to not simply take the current system design and port it directly to a new technology. Not only is this often simply not possible in terms of system architecture, but it is usually undesirable. Unless the research revealed that the current system design maps perfectly to current user and business needs, engaging in interaction design activities at this stage will lead to a much better result. And even if the current system is well designed and up to date, there are usually areas for improvement, not to mention the potential for complementary mobile phone-oriented designs.

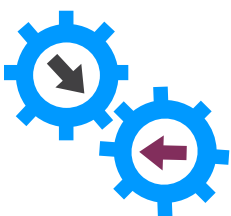
Going through a rigorous interaction design and validation process will almost always result in a much more usable design, which means fewer user errors, fewer misinformed decisions, greater efficiency and productivity, and greater customer and employee satisfaction. Validation sessions with future users allow to understand the level of comfort and usability of the interaction design concept and at what areas fine-tuning is necessary (see “Usability Testing” below).

An integral part of the UX design is ensuring that the product has a contemporary, attractive and professional look that is in line with your company’s branding requirements. As both branding and visual design styles continuously evolve and change, the chance is that the look of your product has to be updated.

The outcome of the UX/interaction design will provide much more confidence and clarity when going into the implementation phase and so also results in lower cost in that phase, because developers can work off validated and agreed upon design specifications.

### Don't Have UX Professionals?

Many teams do not have access to UX/Design professionals who can help them optimize their solutions to best fit their users and business needs. Infragistics can help you with that! We have dedicated, accredited professionals with years of experience who can work with you to determine the right level of UX/Design investment and do the necessary research and design to ensure you get the best possible ROI.



### Technical Feasibility

As part of the interaction design process, the technical team should always be involved in order to verify and validate that the design being proposed is technically feasible. Sometimes this is not immediately obvious, and in that case, some technical prototyping should take place to ensure designs are feasible.



### New to the Modern Web?

Because modern Web is so new and still largely un-adopted in the enterprise, it's important to ensure you have people who are experienced to offer the right insights and be able to quickly and effectively prototype design ideas, and Infragistics is there to help you with that as well. Our architect consultants have tons of experience helping companies move to the modern Web and can both help you design the solution right as well as equip your development teams for success as they embark on grappling with these new technologies.



## Frameworks / Toolsets

Responsive Web Design (RWD) vs. Adaptive Web Design (AWD) vs. Complementary Apps  
A key high-level consideration is deciding how to deal with the challenge of running on multiple devices that have varying form factors, input modalities, and capabilities.

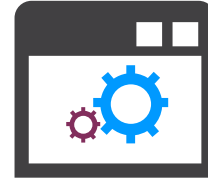
### RWD

- Dictates that you have one app that uses CSS media queries to alter the structure and layout to fluidly adapt to different devices.

### AWD

- Complements RWD by typically adding some sort of server-side or at least dynamically loaded (through JavaScript) changes to the interface to suit different devices.

Your decision to go with RWD or AWD should take into consideration the breadth of devices and native features that you intend to support.



## UI Frameworks/Tool sets Selection

To achieve a modern web application that is well adopted, increases user productivity and of high quality, third party User Interface components and components should be utilized. It is counter-productive for LOB developers to create charts, data grids and document frameworks when they are offered at high quality by third parties.

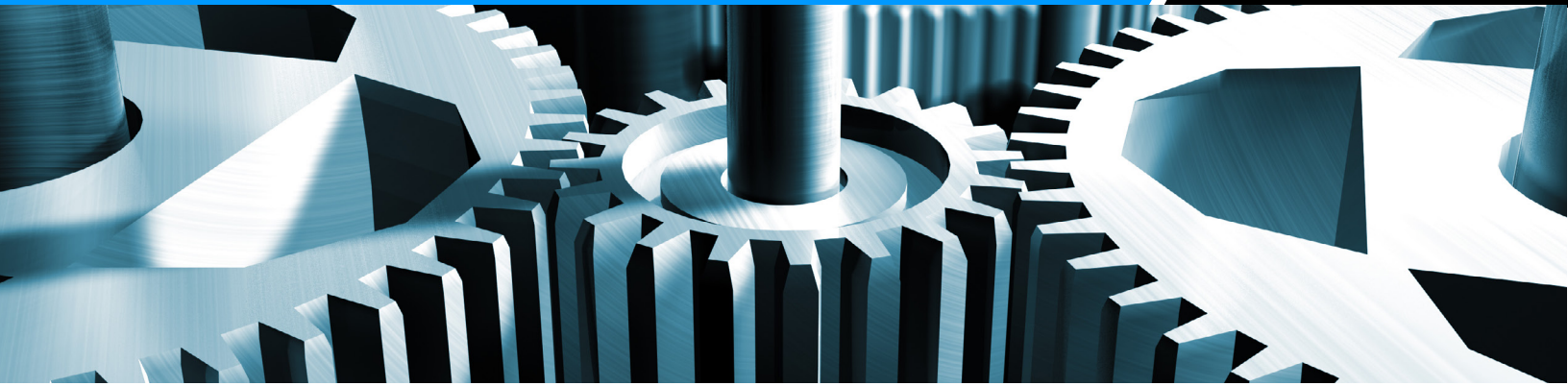
Standardizing on one third party control offering is recommended as the following benefits will be realized:

- Higher code reuse which leads to shorter development time
- Lower learning curve for developers
- Consistent look and feel of app (better User Experience)

The following criteria should be used when selecting a tool set:

- Integration with modern web application frameworks such as KnockoutJS or AngularJS
- Extensive, Advanced LOB and Data Visualization Components
- ASP.NET MVC—decide whether or not you want to use server-side ASP.NET MVC wrappers or pure client-side jQuery UI-based components with ASP.NET Web API.
- Developer Support – Support is more costly than application development. It is important to select a vendor that meets the support needs of your development team.
- Documentation / Community Support – This is critical to the productivity and the uptime of your application.
- Easy-to-use Tools such as Page Designers – Most desktop developers are accustomed to the Visual Studio “drag and drop” paradigm. This functionality does not exist in Visual Studio. Selecting a vendor that offers this functionality will improve developer productivity and may shorten the learning curve.





# Implementation Phase

Once you have the research and design, of course the next step is making it happen.

## Identify Training Needs

The first step to making it happen is ensuring you have the knowledge to do so. While being an experienced desktop developer definitely gives you a leg up—especially when working on familiar problem domains, if your teams have not worked on Web, and especially modern Web, there will be a huge learning curve for them to adapt to. Just throwing them in and hoping they figure it out is a recipe for project failure and dissatisfaction amongst your team members. You really need to ensure your teams are equipped for grappling with modern Web complexities.

We recommend that the training include:

- Comprehension of modern web technology:
  - o HTML
  - o CSS
  - o JavaScript
  - o Any frameworks/libraries that will be used such as jQuery or Angular
- Structuring Code
- Building/consuming web data services
- Testing Procedures/Tools
- Builds
  - o Continuous Integration
  - o Test
  - o Staging
  - o Production/Release (Continuous Delivery)



Infragistics also provides in-depth, in-house training to get your development teams going faster on these projects. In addition to training, we offer mentoring throughout the migration process. This entails Infragistics specialists being available to review progress, provide guidance, and help with decision making.

## Software Development

The recommendations in this framework can be applied to any software development methodology such as Agile or Waterfall.

Regardless of what methodology is used, we strongly recommend that prototyping and usability testing be employed. Both are discussed in subsequent sections.

## Prototyping

Effective prototyping will greatly improve the chance of a successful project. It is important to present prototypes to the stakeholders early in the process. The granularity and type of feedback that users provide with a prototype is of higher value than feedback obtained from static images or mock up screens.

The following benefits will be realized by adding rich, actionable feedback into the cycle:

- Reduced cost due to less rework during development cycle
- Greater chance of stakeholder acceptance
- Increased user productivity
- Fewer usability-based errors, which can be devastating and costly
- Increased employee effectiveness and job satisfaction


Prototyping tools such as Indigo Studio can bring value to this effort by allowing rapid prototyping and providing a mechanism to receive rich and rapid feedback from stake holders and end users.

## Usability Testing

How do you know whether future users can engage with your web application in an effective and efficient way? Usability testing is about exposing products or even just concepts to a small number of representative users and observing how they cope based on defined test tasks. Where do they struggle? What do they like? Even a supposedly small difference between the desktop and web version of the same product, for example changing the name of a push button on the UI, can create a big usability issue that may be hard to discover during design but is easily unveiled by test users. Based on the empirical results from usability testing, you know what aspects of your application work well and – more importantly – what aspects still need work.

Usability Testing can be done at various points during development, but the earlier a product is tested with end users and then iteratively refined, the better it is from a cost-benefit perspective. Thus, usability testing oftentimes follows a prototyping phase (see above) instead of being done at the very end of a project.

Infragistics UX experts can help you with usability testing as well. There are many ways and options for doing it that can be adapted to your company's particular circumstances.



No matter how it is done, usability testing is a hugely valuable way to improve the quality of your solutions. It can be done both with prototypes, like those from Indigo Studio, as well as with running apps.

## Summary

Migrating desktop applications to the web requires careful, comprehensive planning. Using the framework outlined in this document will help ensure the success of your project by avoiding common pitfalls. This methodical approach helps reduce costs by maximizing reuse and minimizing rework – while at the same time never losing sight of the user experience. Please keep in mind that no two migrations are the same, but the principles of this framework apply universally.

If your team is new to these projects, we highly recommend engaging third-party professional services team be engaged early in the project. Infragistics is uniquely positioned to help you by partnering with you during the analysis and design phases, guiding and training your teams, and providing commercial-grade software tools and components to help you build the best solutions with reduced cost and high ROI.



To find out more about anything raised in this paper please get in touch at: [www.infragistics.com](http://www.infragistics.com) or contact Frank Sacco, Vice President of Sales at 609.297.4182

