1 Problem Analysis and Validation

	Description
	Are the requirements stable and has the entire team agreed to them?
	How frequently is the system expected to adapt to change?
	How quickly are changes expected to be deployed? Hot patching? Long Q/A? etc.
	What are the clear "unknowns" that require research?
	What is the benefit of the requirement/outcome to the stakeholder: critical, important, useful?
	Have all tactical objectives and constraints been specified with clear goals and go/no-go marks?
	What risks need to be mitigated: performance, availability, adaptability, modifiability, security?
	Must the system respond within a specific time window?
	Must the system handle a certain number of concurrent requests?
	Must the system handle a certain size or type of payload data?
	Are there any other performance benchmarks that need to be achieved for success (PageSpeed, YSlow, Grinder)?
	Have all performance sensitivities been specified? eg: Throughput, Real-time response, recovery time?
	Is there a need for encryption? What needs to be encrypted, when, and to what degree?
	Are there any additional security guidelines or policies needed for the feature?
	Does the system need to distinguish different roles or permissions across users?
	Does the system need to audit itself on transactions? Do any invariants need to be checked and enforced during runtime?
	For all mission-critical features, were availability benchmarks explicitly defined?
	Are there any extra constraints the developers want to place on the system?
	Were quality attributes extracted by analyzing what constitutes a system failure?
	Have all risks and quality attributes been ordered or prioritized?
	Were sound validation methods used to elicit the specification?
	Was the appropriate data/research used to support the specification?
	Was the user experience validated with user interviews?
	Is the data from user input and user interviews conclusive?
	Is there a process or point-of-contact for further specifying incomplete specifications discovered later in the project?
	Was the data architecture, data collection, and data points for the feature/requirement identified and specified?
	Was a business-level "success" metric defined with a clear threshold identified, supported with objective data?
	Were all the data architecture and system architecture changes reviewed by more than one person, documented, and aligned with the project's quality attributes?
	All guards, pre/post conditions, and invariants for loops and classes/data have been specified and captured in tests?
	Where possible, was an FSM is used to describe and model the feature/requirement?
	Are all external interfaces known?

2 Planning and Estimation

	Description
	Have the specifications been analyzed for completeness?
	Have incomplete specifications been noted and more details requested?
	Have all input points and trust boundaries been identified?
	Is there a list of possible malicious user input?
	Is there a strategy for handling malicious input when it is detected?
	Is the system size and complexity a concern?
	What "unknowns" or technologies need to be carefully managed?
	Do we need to hire a specialty contractor or to learn a new skill?
	Given the risks, how much architecture is needed?
	How do you determine the feasibility of algorithms and designs? eg: Prototyping, Modeling, Analysis, Simulation
	What benching infrastructure do you need in place to ensure success? Does it need to be built?
	Have multiple engineers decomposed the problem?
	Have multiple engineers crafted/influenced/reviewed/approved the potential API changes?
	If needed, have estimations been generated using democratic, bottom-up techniques?
	If needed, have estimations been generated using data and top-down techniques?
	Have tickets been created for all foreseeable pieces of work and assigned to a developer?
	Have the tickets noted the impact to the Stakeholder? High, Medium, or Low
	Have all engineers voiced their biggest concern or worry with the requirements/proposed feature?
	Are all deliverables and milestones clearly defined, understood, and agreed to by the team?
	Do all team-members understand their roles and responsibilities?
	Do all high-risk areas already have a pairing strategy/team identified?
	Do all high-risk areas have at least a loosely defined alternative plan?
	If a schedule exist, is the schedule realistic?
	Are there external dependencies which are likely to impact the schedule?
	Is this a replacement for an existing system? If so, is there a plan for data migration and switching control over?

3 Architecture

	Description
	What is the expected lifespan of the system?
	Will the system need to respond to technological changes over time, such as new versions of Middleware, APIs, or other products?
	What is the architectural impact? None, extends, modifies
	What changes can we anticipate in the future, and how can we make them easier to accommodate?
	Is the architecture adequately designed to age and evolve gracefully?
	Does the architecture consider constraints and restrictions imposed by the physical deployment environment?
	Have all needs for clustering, sharding, caching, offloading been specified and modeled appropriately?
	Are all DB/data interactions split against CRUD operations and organized by resource?
	Do all introduced services/processes have dedicated monitoring?
	Are there any updates needed for system monitors or visibility?
	Is there an appropriate capacity plan for the new feature?
	Is there graceful degradation of the service?
	Have appropriate failover strategies been built in to the system?
	Are all technical risks mitigated or addressed in a contingency plan?
	Is availability accounted for with the architecture?
	Does the new service have integrated logging? Is it configurable?
	Is the system and its components isolated appropriately from outside concerns?
	Are behaviors asynchronous at a macro/system level?
	Have all deployment environments been identified? DB servers, API servers, web machines, load balancers, etc
	Is the description of the architecture complete, meaningful, and clear?
	Is the architecture an appropriate level of detail, given the objectives, risk, and impact?
	Does the architecture clearly convey not only the solution but also the motivation and objectives related to the decisions that have been made in shaping the architecture?
	Have the architectural goals been clearly described?
	Have architectural patterns and styles been clearly noted?
	Has the architecture been appropriately partitioned to reduce complexity and risk?
	Are the architectural partitions all loosely coupled and highly cohesive?
	Where possible, does the architecture promote a "share-nothing" design?
	Are all interfaces to components well-defined?
	Is the architecture free of single points of failure?
	Do all models include a legend?
	Are the models accurate: self consistent, free of dangling references, and falsifiable?
	Are the models' precision such that stakeholders can understand it enough to identify flaws/provide feedback?
	Can the team deliver the architecture?

4 Data

	Description
	Is all necessary data accurately/appropriately captured for the feature?
	Have all BI and report requests been vetted by the data model?
	Is the data model flexible enough to accommodate new BI requests or data reports?
	Have hard validation rules been created for type, length, format, and range of fields?
	Is there any new data that can enhance old data or previous features?
	What sort of learning are you aiming to do with the data? What is the goal of this learning?
	What process is needed to clean or scrub the data?
	Is there any need for an external data set?
	Is there an understanding of what pieces of data are cacheable?
	Is there a general caching strategy in place?
	Is there a separation between data that needs to always be accurate vs. data that can be "good-enough" and eventually accurate?

5 User Experience

	Description
	Nielsen's heuristics are used as a baseline for UI/UX go/no-go.
	Additional guidelines are specified as part of the UI/UX/Spec elicitation process, determined by a product leader.

6 Managerial and Resources

	Description
	Is there a need for new hardware resources to reduce the scope or deliver the feature?
	Is there a need for a new software? eg: framework, tool, etc
	Is the budget capable of accommodating required resources?
	Are there appropriate controls to acquire resources in case of an emergency?

7 Construction

	Description
	There are no magic numbers in the code
	There is appropriate use of namespacing
	Module interaction is programmed against an interface and never to an implementation
	The code is self-evident and architecturally-evident
	Conditional If statements are avoided in favor of polymorphic dispatch
	I/O is isolated in a function call, and separated from the function call that constructs/processes the payload
	Has input validation been applied for all outside user input?
	Is all input validation done in a central location when it makes sense (to encourage reuse)?
	Does the validation address injection, XSS, CSRF, and hijacking concerns?
	Is the design free of client-side/untrusted validation?
	Has all user input been assumed malicious into the system?
	The validation approach is to constrain, reject, and then sanitize input. Looking for known, valid, and safe input is much easier than looking for known malicious or dangerous input.
	All input parameters are validated, including form fields, query strings, cookies, and HTTP headers
	HTTP header information is not relied on to make security decisions
	Input file names and file paths are avoided where possible
	Application trust and authentication boundaries are clearly defined, documented, and enforced
	Authentication identities and roles are clearly defined, documented, and enforced
	SQL/DB authentication is used
	The DB is not listening to an external IP connection, or is isolated within a network
	The design adopts a policy of using least-privileged accounts
	Authentication tickets (cookies) are not transmitted over non-encrypted connections
	The session cookie is http-only
	CSRF cookie protection is in a dedicated cookie, separate from the session cookie
	The application's login does not have permissions to access tables directly
	Access to system level resources is restricted
	Configuration secrets are not held in plain text in configuration files
	Configuration stores are secured

8 Construction; con't.

	Description
	Secrets are not stored unless necessary. Alternate methods have been explored at design time
	Secrets are not stored in code
	Sensitive data is not logged in clear text by the application
	Sensitive data is not stored in persistent cookies
	Sensitive data is not transmitted with the GET protocol
	SSL is used to protect authentication cookies
	The contents of authentication cookies are encrypted
	Session lifetime is limited
	Session identifiers are not passed in query strings
	No encryption is weaker then SHA256
	Password encryption makes us of advance methods like bcrypt
	Platform-level cryptography is used and it has no custom implementations
	Keys are periodically recycled
	When needed, view state is protected using MACs
	Application errors are logged to the error log
	An appropriate approach to exceptional handling is used
	Software design patterns have been appropriately used and clearly noted
	State is avoided when possible
	Higher-order functions are used when possible
	Composition is favored over inheritance
	Immutability is favored where possible
	Handling a request is modeled as applying transformations to a given set of data

9 Quality Assurance

	Description
	Was SeleniumIDE plugin used to capture all manual tests that couldn't be captured otherwise?
	Are all tests under revision control?
	Is there an established dataset used for testing?
	Were all internal interfaces tested with XUnit frameworks?
	Was the code analyzed with a lint/quality static checker tool?
	Was the code appropriately documented and organized based on resource, action, or some other classification?
	Has more than one engineer seen, ran, and approved the new code and tests?
	Have all internal and external integration points been exercised with tests? (via pair-wise or another method)
	Has the system successfully run in staging?
	Has the entire company seen a demo of the feature in staging and approved it?
	All internal and external interfaces are properly documented and make use of descriptive variable names
	All public API calls have an ideal example code prior to launch
	Have all architectural goals been met?

10 Deployment

	Description
	Has static content been placed on the CDN?
	Have optimal builds been automatically generated for deployment?
	If SSL is needed, is the CA in place and working?
	Have all deployment environments been prepared? DB servers, API servers, web machines, load balancers, etc
	If needed, have infrastructure resources been warmed (caches, JVM, etc)?
	File permissions are correctly set on deployment machines with security and necessity in mind
	No services run as root
	Least-privileged process accounts and service accounts are used
	Firewall rules are correctly set for new features
	Are all external business factors (email, branding, press release) ready fro deployment?
	Is there a single-action system rollback if something goes wrong?
	Is a systems operation team prepared to go live with the system?
	Do all engineers feel confident in the quality of the system?