Dev Spec Header

Version and Date

- List all versions and dates the document was ever edited.
- Last person to edit the document must update it.

Author and Role

- List all authors and roles.
- Never delete anyone.
- If some people are only associated with a specific version, write those versions next to the people's names.

Dev Spec Architecture Diagram

1. Draw one box per class.
2. Put a list of field and method names in each box.
3. If a field's "value" does not "visually fit" in the box, it is pulled out into its own box.
4. Method names should end in ()
5. Fields should be distinguished graphically from methods in the Class boxes.
6. All fields should have a data type next to them to say what kind of thing they hold.
   - If the type is rarely used, an example instance of the type can be added in a "legend" box on the side.

Dev Spec Architecture Diagram

1. Draw component/module boxes around groups of related classes/components.
   One component box per web page (MVC)
   Model has only fields which reflect the data that the view needs to show.
   Every class and component should be inside *some* module box.
2. Arrows connect related boxes together.
   Container boxes are connected to the class of the objects they contain
3. Legend box clearly explains the purpose of the colors and arrow types in the diagram.
4. Abbreviations should never appear. This is not the document to omit information for expedience.
5. Every class, component, and module should be labeled with a specific code, not just a name. (e.g. SA1.6 Admin Dashboard)

Dev Spec Class Diagram

1. Draw every class in its own box.
2. Label each box by the classname and the class's label.
3. Draw a directed edge from every superclass to its subclasses.
4. Be sure the most abstract superclasses are on top, and the subclass leaves are on the bottom.

Dev Spec List of Classes

1. List all classes from the Architecture Diagram.
2. Classes should be grouped by component and/or module.
3. Classes not related to concepts in the original design spec should be clearly indicated.
4. Data storage classes (i.e. structs) should be separated out and labeled as such.
5. All classes should have specific labels (e.g. SA2.3.1)
6. For each class, write down what its purpose is.
7. Classes should be labeled with the Design Document labels of the features and operations they will be implementing.
   - If the name of the class does not make it obvious which feature you're referring to, add it in English.
8. Don't list the methods. These go in the API section.

Dev Spec State Diagrams

1. List all data fields that make up your notion of the system state.
2. Every state should have a unique name and label.
3. The initial state for each scenario should be clearly labeled.
4. The actual state of the system at each state should be identified.
   - The parts of the state that change should be indicated along every outgoing edge.

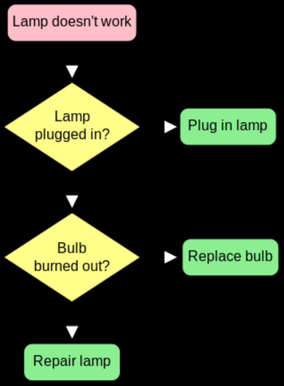
Dev Spec State Diagrams

1. Draw directed edges between states that can be reached from one another.
2. When a set of methods is used to get from one state to the next, write them down on the edge.
3. When method names appear, they should be prefixed by their scope. i.e. Module.Component.Class.
   E.g. Backend.Database.Security.FetchOAuthTokenForPerson()
4. When there are multiple edges you could take
   a. Label the bottom of the state by the predicate you use to decide.
   b. Label each outgoing edge by the predicate values that take you to the indicated state.
5. The legend box should clearly explain the purpose of the colors, fonts, graphics, etc in the diagram.

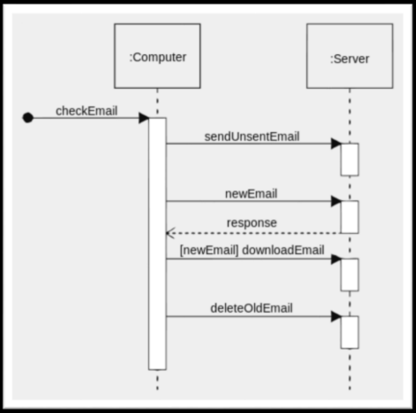
Dev Spec Flow Chart

1. Be sure there is at least one flow chart per scenario.
2. Label each flow chart with the name and label of the corresponding scenario from the UI/Design spec.
3. Each flow chart should be accompanied by prose from the UI/Design spec scenario to explain how the flow chart achieves the scenario.
4. Use standard flow chart symbology (http://en.wikipedia.org/wiki/Flowchart) or Sequence Diagrams (http://en.wikipedia.org/wiki/Sequence_diagrams)
5. The starting and final states of each flow chart should be clearly indicated.
6. The state diagram represents the union of every scenario shown in a flow chart.
   Each box in the flow chart should be labeled by the state in the State Diagram it represents.

Example flow chart



Example Sequence Diagram



Dev Spec Possible Threats and Failures

1. For each component and module in the system, list the user-visible and internally-visible effects (even if none) if it
   a. Runtime
      a. *crashed its process*
      b. *lost all its runtime state*
      c. *erased all stored data*
      d. *jumped to an unexpected place in the state diagram or flow chart*
      e. *noticed that some data in the database appeared corrupted (i.e. failed its rep invariant)*
      f. *remote procedure call failed*
      g. *server overloaded*
      h. *server out of RAM*
      i. *database out of space*
   b. Connectivity
      a. *lost its network connectivity*
      b. *lost access to its database*
      c. *network traffic spikes*
      d. *lost access to Twitter and/or Facebook*
   c. Hardware
      a. *server down*
      b. *loaded up a bad configuration file*
      c. *system moved to a new geographic and/or URI*
   d. Intruder
      a. *denial of service attack*
      b. *someone broke in and trashed the operating system*
      c. *someone broke in and rewrote your project code*
      d. *someone broke in and copied your database*
      e. *bot signs up and spams users*
      f. *Http session hijacked*

Dev Spec Possible Threats and Failures

1. Every failure mode should be labelled uniquely.
2. For every failure mode listed, describe the recovery procedure required to get the system back in a sane configuration.
3. The test spec version of this section lists the diagnostic procedures required to notice each of these failure modes. Cross reference the failure mode label.
4. Rank the failures by likelihood of occurrence and impact to system and business model.

Technologies

1. Provide a complete list of technologies used in your system that you aren't writing yourself.
   Indicate what the technology is being used for.
   Indicate why that technology was picked over others.
2. Provide a URL to source location, author, and documentation for each technology.
3. Don't leave out technologies like language, common libraries, or necessary tools.
4. Do mention the required version number for each technology.
5. Every technology should have a unique label.

APIs

1. List all methods, grouped by class, component and module.
2. Each method should have a return type and parameter types.
3. Method overloads or overrides should be pointed out.
4. Public and private methods should be indicated and grouped within each class.
5. Every method, class, component, and module should be labelled with the label used in the Architecture Diagram.

Public Interfaces

1. Similar to the API listing, but only includes the public methods.
2. Within each class, group methods by whether the method is used by classes within the same component, across components in the same module, or across modules.
3. For each component box, list the classes and methods that it uses from classes defined in other components.
4. For each module box, list the components, classes, and methods that it uses from components defined in other modules.
5. If your APIs are accessible in more than one language or interface, indicate how to call the APIs from each (e.g. C# and REST).

Data Schemas

1. List each data type you store in a table in your database and indicate which class(es) hold this data at runtime.
2. For each database data type, list the columns and associated database type used to store it.
3. If there is a non-obvious mapping from field type to database type, or field values to database column values, explain it next to the field.
4. For every data type, guestimate its storage requirements in bytes, or provide a function to compute it in terms of the number of elements stored.
5. Every database data type should have a unique label.

Risks to Completion

For each module, component, class, method, data schema, and technology used in the dev spec, talk through the difficulty a developer (you) would have to learn to use it, design it, implement it, verify that it works, maintain it over time, and update it.

- If it is off-the-shelf (built by someone else) it's likely to be easier to build and maintain, but it may not be as easily customized to your purposes. Talk about if this is needed.
- What criteria do you use to determine whether to take or rollback an updated version?
- Do you get the source code with it? Or just a binary?
- If you can't figure out how to make something work, is there support from the sellers? Or do you have to find someone who knows how to use it yourself?
- If there is a bug, is there support from the authors? Or do you have to fix it?
- If it needs security fixes, is there support from the authors? Or do you have to fix it?
- How long is your support contract and how much does it cost compared with the original purchase price?

Security and Privacy

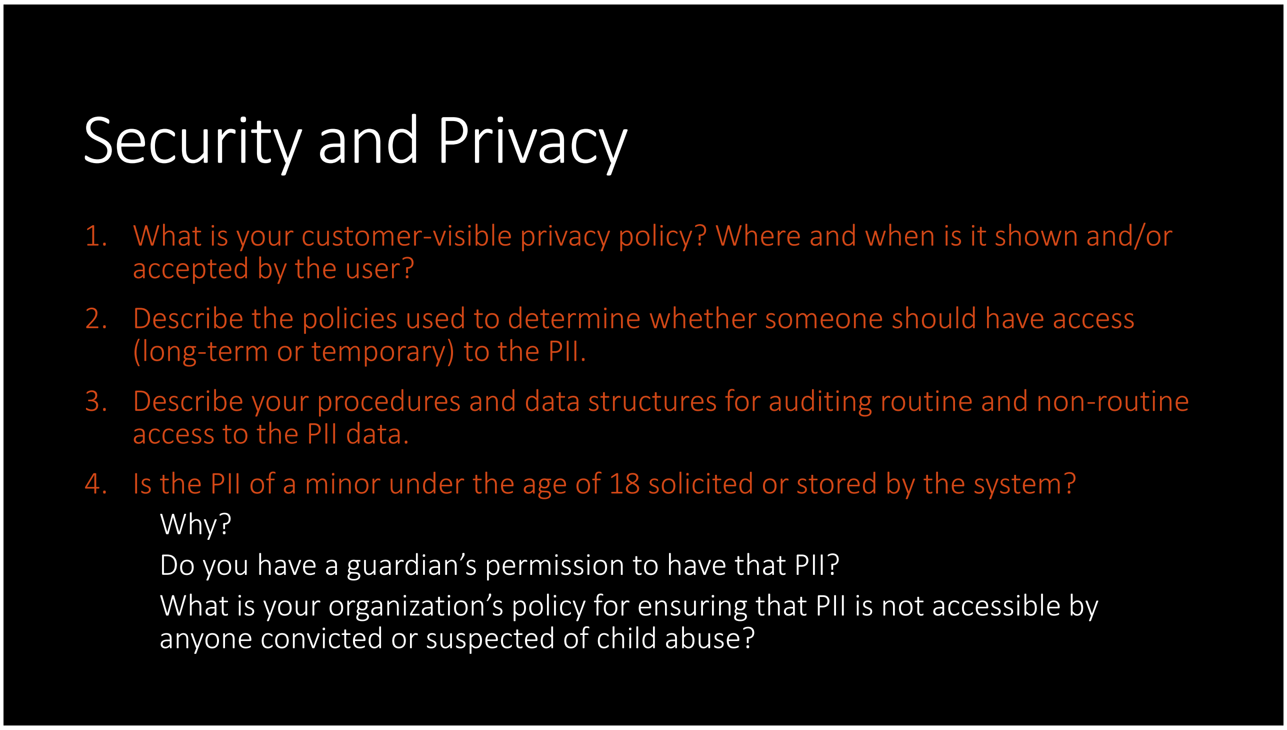
1. List all PII temporarily stored by your system.
   - Justify why you need to have it, even on a short term basis (e.g. password)
   - Where did the data come from prior to entering your system?
   - Through what modules, components, classes, methods, and fields did it move after entering your system but before it is used?
   - Which modules, components, classes, methods, and fields actively make use of the PII data? What operations are performed on it?
   - Through what modules, components, classes, methods, and fields will the PII move after being used by your system?
   - How is the PII disposed?
   - How did you protect the temporarily stored PII from attacks on your process' runtime memory store or network connection?

Security and Privacy

1. List all Personally Identifying Information (PII) stored in long-term storage in your system.
   - Justify why you need to keep it in storage.
   - How exactly is it stored?
   - Where did the data enter your system?
   - Through what modules, components, classes, methods, and fields did it move before entering long-term storage?
   - Through what modules, components, classes, methods, and fields will it move after leaving long-term storage?

Security and Privacy

1. List the people who have responsibility for securing each unit of long-term storage/database.
2. List the security officer in your organization with responsibility for auditing that security practices are being followed by all people with access to secure data.



Security and Privacy

1. What is your customer-visible privacy policy? Where and when is it shown and/or accepted by the user?
2. Describe the policies used to determine whether someone should have access (long-term or temporary) to the PII.
3. Describe your procedures and data structures for auditing routine and non-routine access to the PII data.
4. Is the PII of a minor under the age of 18 solicited or stored by the system?
   - Why?
   - Do you have a guardian's permission to have that PII?
   - What is your organization's policy for ensuring that PII is not accessible by anyone convicted or suspected of child abuse?