Documents

Every document should

1. have a title, a date, and its authors (everyone who contributed to its content);
2. be thoroughly checked for spelling and grammar errors; and
3. use fonts, colors, and alignment consistently (same fonts for same-level headings, …).

The Requirements Document

The purpose of this document is to explain the requirements of the product to be constructed. It should be structured with the following sections.

Project Overview: Write a short overview of the project in a paragraph with 5-6 sentences on (a) the overall problem that the project plans to address; (b) the main functionality that the envisioned system will deliver; (c) the community of users for the envisioned software; and (d) the operational context in which the system is to be embedded. Consider this as the summary pitch that you would give to attract investment for your project.

Project Glossary: Define any special terminology in the application domain (i.e., client terms, not developer, programming, or implementation terms).

Storyboarding

Low-fidelity sketches/wireframes of a few key screens should also be developed to communicate the overall look-and-feel of the application, as well as the high-level user-interaction design. A good document to consult on how to develop your storyboards is at http://www.agilemodeling.com/artifacts/uiFlowDiagram.htm

Use Cases or User Stories

You can choose either notation to describe the functional requirements of your system. The use-case diagram (see https://en.wikipedia.org/wiki/Use_Case_Diagram) should include all use cases, all actors and all the relations among them. The use case descriptions should be complete with pre- and post-conditions, and all flows (typical and exceptional) specified. A simple, yet complete, template for use-case specification can be found at https://en.wikipedia.org/wiki/Use_case#Examples.

The user stories should be organized according to their relevant users and the features to which they most-closely correspond. User stories are described in https://scalingsoftwareagility.files.wordpress.com/2009/11/user-story-primer_1.pdf (as mentioned in the requirements slide deck).

Technical Resources: List useful informational resources around the tools and frameworks you plan to use.

Similar Products: Point to competitive or similar products that your system may use as reference and/or inspiration for requirements.

The Software-Design Document

The design document should be consistent with the requirements document (it should document and explain how the design will actually deliver the required features). The document should include a short description of the overall architecture style of the system, its high-level system components, and their logical (what data they exchange) and control (how they invoke each other) dependencies.

In principle, your design document should include a description of three system views: (a) a description of the main conceptual modules, the platforms on which they are deployed, and their interdependencies; (b) a static view of the logical entities of the domain; and (c) and a description of the dynamic control and information flow among these entities.

If you adopt an object-oriented style for your system, these three views can be captured with the following three models:

> **a)** the component model (seehttp://agilemodeling.com/artifacts/componentDiagram.htm) describing the high-level modules/packages of your system and the APIs through which they exchange information among them;
> **b)** the logical model of the system domain (see http://www.agilemodeling.com/artifacts/classDiagram.htm); and
> **c)** the core dynamic behaviors of the system (see http://www.agilemodeling.com/artifacts/sequenceDiagram.htm).

If your system is not naturally object oriented, you can use another modeling notation for each of the three views required. You should review I like Documenting Software Architectures: Views and Beyond, by Paul Clements (see http://proquest.safaribooksonline.com.login.ezproxy.library.ualberta.ca/book/software-engineering-and-development/9780132488617).

In any case, make sure that this document is meaningful with respect to your system and be selective in what you portray about the architecture--not overly detailed, but not too high-level either. Suppose a new developer comes to the project: describe what is not obvious and definitely important to know. As for practical guidelines, each diagram should be described and motivated; you should use a consistent naming convention; and key elements should be annotated with further comments to explain their roles.

Project Management

There are two sections in this document: (a) the project plan, and (b) the team members' roles.

The project plan describes the various project tasks, their dependencies and the dates by when they are expected to be completed. It can be specified in terms of a GANTT chart w. tasks and deadlines, or with a storymap. Storymaps are nicely discussed nicely in http://winnipegagilist.blogspot.ca/2012/03/how-to-create-user-story-map.html. There are many resources on GANTT charts and wikipedia gives a reasonable overview.

In addition, this document should describe the roles of each team member and the general task decomposition among them. Ideally, each task mentioned in the GANTT chart (storymap) should be associated with the team member responsible for completing it (and possibly a second team member to serve as a helper).

This document should be viewed as a high-level organization of the issues in your repository's issue-tracking module. A good way to do that is through Gantt charts with hierarchical edges (see http://www.gantt.com/creating-gantt-charts.htm), where use cases correspond to the top level in the hierarchy, and they get decomposed in issues in the issue tracker. If you prefer a "storymap" as your planning document, you should make sure that your user stories correspond to issues, and are organized in (vertical) features and horizontal phases (sprints).

Test Documentation

This document reports on the test suite that you have developed to validate that your system delivers all the user stories you originally identified for your project in your requirements document.

Some tests many need to be manual, and some will be automated through a testing framework (Behat, Lettuce, ...). For each manual test, you should include (a) a reference to the use case (user story, feature) tested, and (b) a sequence of captured screens (or console output) with intermediate sentences describing the interaction leading from one step to the next. Automated tests should be implemented in a special folder in your repository.

In your document, you should also comment on which original user stories you did not test (and/or developed).

Client Documentation

Under this heading, we include (a) the user manual, and (b) a document useful to the future administrators (and developers) of your system.

The form and format of the user manual should satisfy the needs of the client and user community. Generally, it should be organized by user tasks described step-by-step (not by screen elements, for example). It could be an evolution of the storyboard included in the requirements document. If multiple versions of this document exist, such as an offline document and an on-line manual to be available as the system is used, they should be consistent with each other, as well as with any help materials.

The administrator manual should make it possible for your client to install and run your system on their own machines.

Presentation

Your presentation should be about 10-15 minutes long and it should slides on
(a) the purpose and value of your project;
(b) its architecture;
(c) the technologies used in its implementation;
(d) a segment of the screencast demonstrating its key functionalities;
(e) the key challenges you faced and how (if) you addressed them; and
(f) the lessons you learned through the process.
The key criterion is that your presentation should be engaging (do not lose the audience), informative, and without platitudes.

Screencast

A **narrated** screencast of your system demo should be uploaded on YouTube.