

Design Documentation

SWEN-261

Introduction to Software Engineering

**Department of Software Engineering
Rochester Institute of Technology**

Guessing Game Home

Welcome to the Guessing Game Project!

Team

- Bryan Basham
- Jim Vallino

Design Documentation

Click above for details of the Guessing Game design documentation.

Setup Guide

Click above for details about how to setup your development environment to work on this project.



Design documentation can be a valuable communication tool.

A design document is a way for you to communicate to others what your design decisions are and why your decisions are good decisions.

From [How to Write an Effective Design Document](#) by Scott Hackett

- Design documentation should be short and easy to read.
- It should communicate key architecture and design decisions.
- It should generally move from high-level to low-level.
- It should provide justification for design decisions.



We recommend a simple design document structure.

- Executive Summary
 - *Purpose*
 - *Glossary and Acronyms*
- Requirements
 - *Definition of MVP*
 - *MVP Features*
 - *Roadmap of Enhancements*
- Application Domain
 - *Overview of Major Domain Areas*
 - *Domain Area Detail*
- Application Architecture
 - *Summary*
 - *Overview of User Interface*
 - *Tier Designs (UI, Application, Model)*
 - ♦ Summary
 - ♦ Static Model(s)
 - ♦ Dynamic Model(s)

These general tips for effective writing apply to your design documentation too.

- Create a narrative to engage the reader.
- Writing a spec is like writing code for a brain to execute.
- Write as simply as possible.
 - *Use the active voice.*
 - *Use short, declarative statements.*
- Review and reread several times.
- Balance text with diagrams.
 - *Don't have long stretches of text.*



You should follow these tips to maximize the effectiveness and professionalism of your models.

- Define a purpose for each model/diagram and use a level of abstraction appropriate for the purpose.
- Use standard modeling techniques (ie, UML).
- Use non-standard models when they are clearer than the alternatives.
- Use a professional modeling tool.
- Create a layout that is easy to comprehend.
- Use color, fonts and styles that enhance understanding (high-light important elements)
- *BUT...* do not use such stylistic frills for solely aesthetic purposes.

The software design is just one aspect of a project that can be documented.

- Others include:
 - *Setup guide*
 - *UI and UX design and style guide*
 - *Acceptance test suite*
 - *Online and in-system help docs*
 - *Training docs and video tutorials*
- Project documents must *live*:
 - *Use collaborative, version-able documentation tools.*

Keeping your design documentation up-to-date will now be part of your standard workflow.

☒ **Definition of Done Checklist** [Delete...](#)

0%

- ☐ acceptance criteria are defined
- ☐ solution tasks are specified
- ☐ feature branch created
- ☐ unit tests written
- ☐ solution is *code complete*, i.e. passes full suite of unit tests
- ☐ design documentation updated
- ☐ pull request created
- ☐ user story passes all acceptance criteria
- ☐ code review performed
- ☐ feature branch merged into master
- ☐ feature branch deleted