Documentation

CS 346 Application Development

https://student.cs.uwaterloo.ca/~cs346

Contents

Introduction

Why Documentation?

We want to build maintainable software, that can be modified, update and remain useful over a long period of time.

However, writing well-structured code isn't enough!

- The person writing the code may not be the person maintaining it.
- Even if you are maintaining code that you wrote, you will not remember the reasons for design decisions you made 6+ months ago.
- Code isn't accessible to everyone in your organization! What about sales? Marketing? You can't expect them to "read the source code" to understand how something works.

Effective documentation is critical for the communication of complex ideas.

- You can't rely on documenting source code to communicate everything.

Types of Documentation

Project documentation

- Tracking project details to help us remember our project constraints. Useful for planning later phases.
- e.g., Issues lists; Milestones; Project plans; Gantt charts.

Design documentation

- Why we made specific design decisions; materials to help new developers understand rationale.
- e.g., UML diagrams; design documents.

Code documentation

- Inline documentation (code comments) to explain peculiarities of an implementation.

User documentation

- Help users understand how something works!
- e.g., how to install; what features exist; what has changed in a new release.

Docs as Code

Documentation as Code ("Docs as Code") is the philosophy that you should be writing documentation with the same tools you use to author and maintain source code [1].

- **Issue tracking**: use this to track doc changes.
- **Version control (Git)**: version your docs with your code.
- **Plain text documents**: you can diff them, use Git.
- **Code reviews**: docs should be included in feature reviews.
- **Automated tests**: unit test your docs!

Documentation an important output from a software release, no different than an installer, release notes, unit tests – *all should be under source control*.

"[Docs as Code] means following the same workflows as development teams and being integrated in the product team. It enables a culture where writers and developers both feel [collective] ownership of documentation and work together to make it as good as possible." - writethedocs.org

Markup Languages

What is a Markup Language?

A markup language is a system of annotating a document to describe its structure and presentation. It uses tags or codes to define elements such as headings, paragraphs, lists, images, links, and more.

- Markdown languages include HTML, AsciiDoc, reStructuredText, Markdown.
- TeX and Typst are document typesetting languages. They are more complex, and meant to handle print publishing e.g., books, journals, articles.

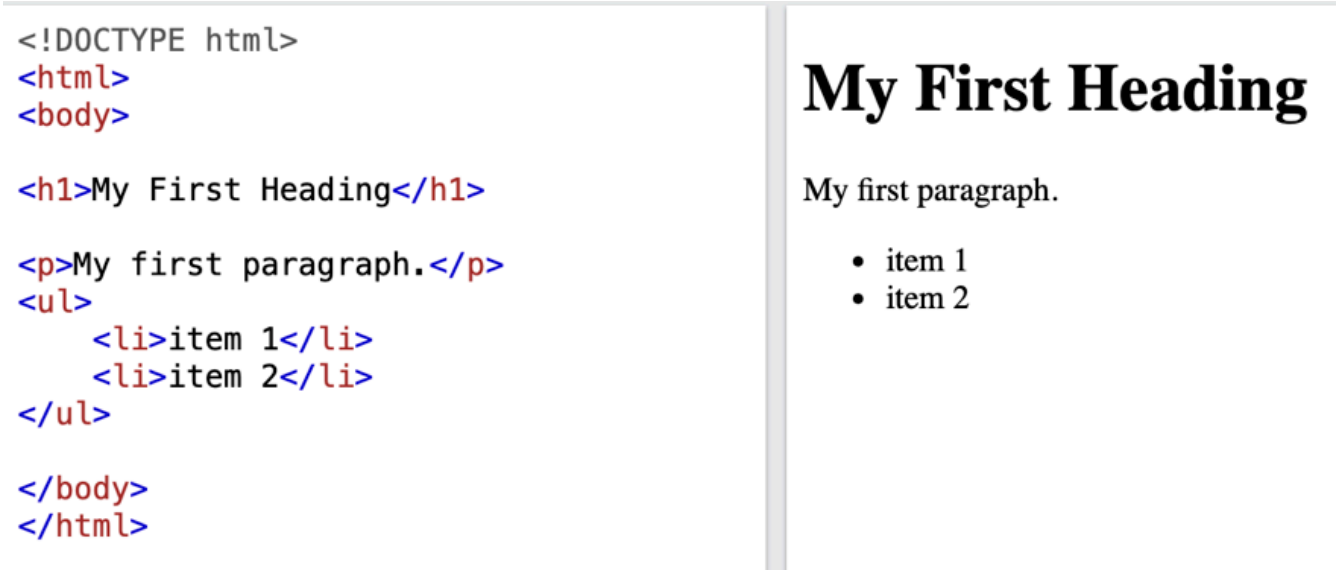

Figure 1: Do you really want to write your documentation in HTML? There are better options.

Markdown (Text)

We'll focus on `Markdown` as one of the most commonly used markup languages for this purpose.

Markdown is a simple markup language that allows you to add formatting elements to a text file. Markdown was designed with a focus on generating HTML [2]

In its original form, Markdown is both:
- A formatting specification, and
- A tool for converting markdown files to HTML for publication.

In recent years, Markdown has become the defacto standard for technical documentation. It is less complete than other markup languages (e.g., `AsciiDoc`) but is simpler to use. It's also supported by other development tools. e.g., Git.

Markdown (Text)

```
## Syllabus

{{< callout type="note" >}}
Introduction to full-stack
application design and
development. Students will work
in project teams to design and
build complete, working
applications and services using
standard tools. Topics include
best-practices in design,
development, testing, and
deployment.
{{< /callout >}}

The course outline is also
published on
[outline.uwaterloo.ca](https://
outline.uwaterloo.ca/viewer/view/
n4wt28).
```

Syllabus

> Introduction to full-stack application design and development. Students will work in project teams to design and build complete, working applications and services using standard tools. Topics include best-practices in design, development, testing, and deployment.

The course outline is also published on outline.uwaterloo.ca.

Figure 2: Our course website is written in Markdown and converted to stylized HTML. Slides are generated from Typst (a different document markup language), and converted to PDF.

Markdown (Text)

Syntax

Why would we use Markdown?

- You can write documentation in any text editor.
- Text, so you can version control it, diff it etc.
- VS Code, most IDEs, GitHub, GitLab support it.
- Defacto standard for software development.

Why not use Markdown?

- There is no standard specification (GitHub and a few organizations have produced extensions).
- Missing support for important features:
  - Footnotes
  - References
  - Floating images
  - Columns

See https://www.markdownguide.org/basic-syntax/

Markdown (Text)

Editing Markdown

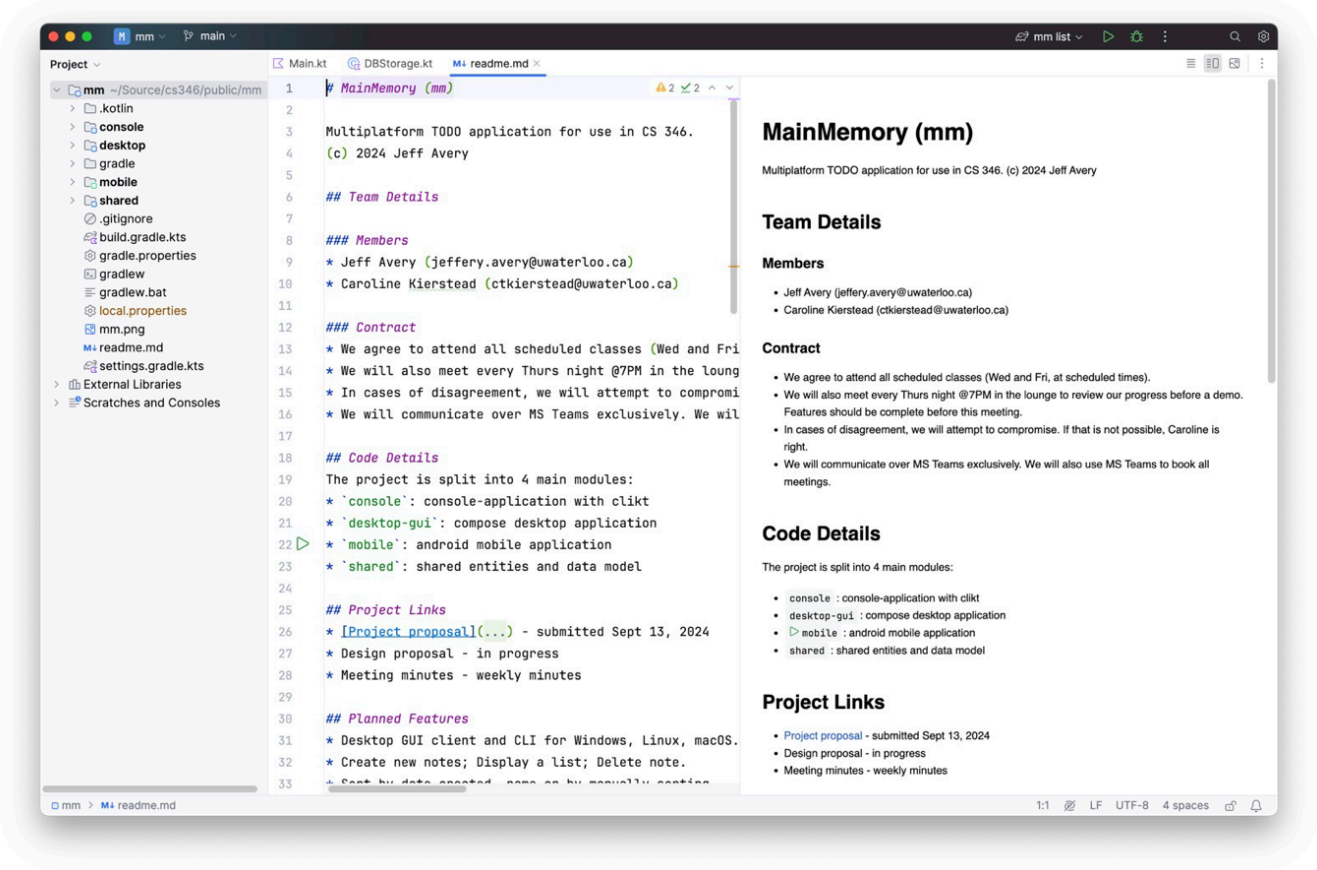

Figure 3: You can edit Markdown in any text editor. IntelliJ IDEA has a plugin that provides syntax-highlighting and other niceties.

Mermaid (Diagrams)

Documentation requires diagrams. We can imagine adding many different types of diagrams and charts to our documentation, including:

- Gantt charts to project management documents.
- Timeline charts to show milestones and your delivery schedule.
- UML diagrams for design, and to document implementation details.
- Component diagrams, class diagrams, sequence diagrams, state diagrams…
- Flowcharts, and requirements diagrams to explain features to customers.
- Pie charts to show results.

Examples of diagramming software:

- Omnigraffle (mac)
- Lucidchart (mac, windows)
- PlantUML (online)
- Mermaid.ai (online)

Mermaid (Diagrams)

Mermaid.ai

We'll focus on Mermaid, which is a diagramming tool that generates diagrams dynamically from markup. It also integrates really well with Markdown!

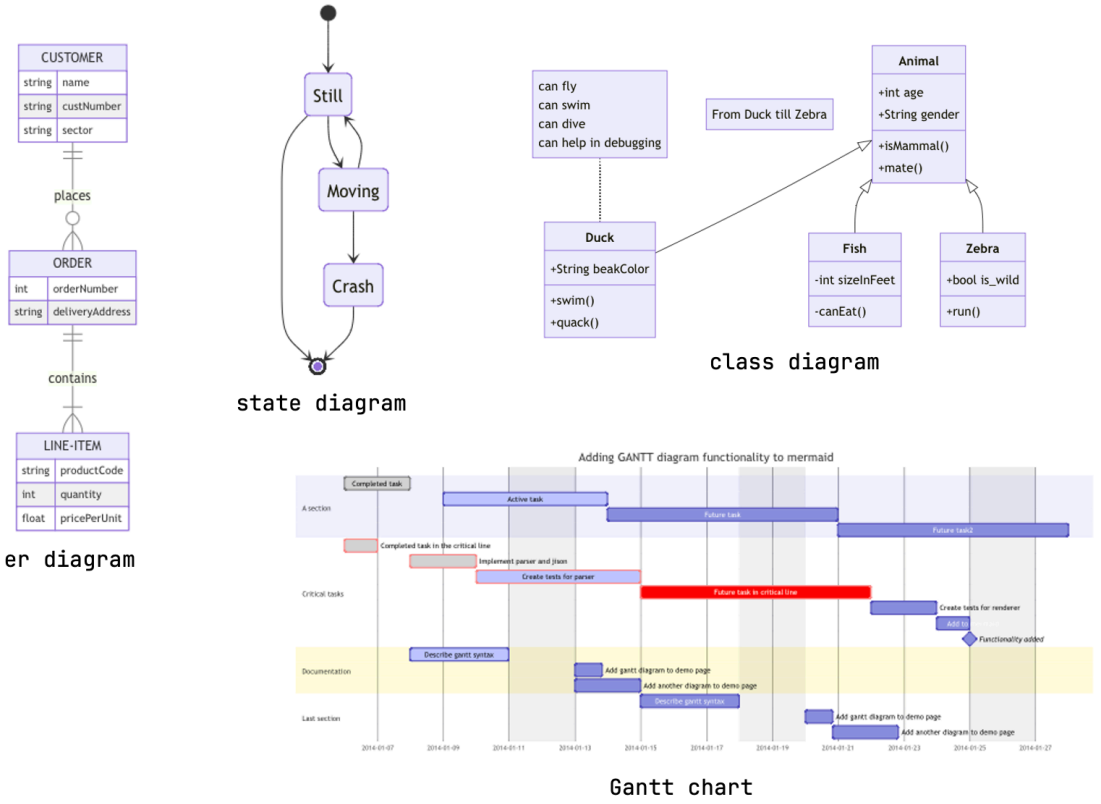

Figure 4: Mermaid can generate a range of diagrams, including project and design diagrams.

Mermaid (Diagrams)

Mermaid syntax can be embedded into Markdown documents to generate inline diagrams.

```
mermaid
classDiagram
  class BankAccount
  BankAccount : +String owner
  BankAccount : +Bigdecimal
balance
  BankAccount : +deposit(amount)
  BankAccount : +withdrawal(amount)
```

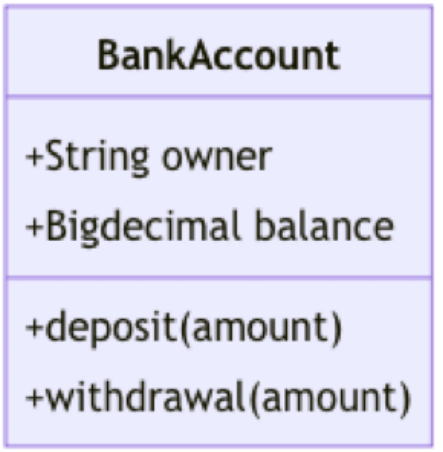

Figure 5: Mermaid syntax is relatively simple, and be used to generate many styles of diagram and chart.

Most environments that support Markdown also support Mermaid.

- This includes GitLab, GitHub, VS Code, IntelliJ IDEA, pandoc, …

Mermaid (Diagrams)

If you install the Mermaid plugin, you can write your documentation directly in IntelliJ IDEA, and it will render diagrams in the Preview pane.

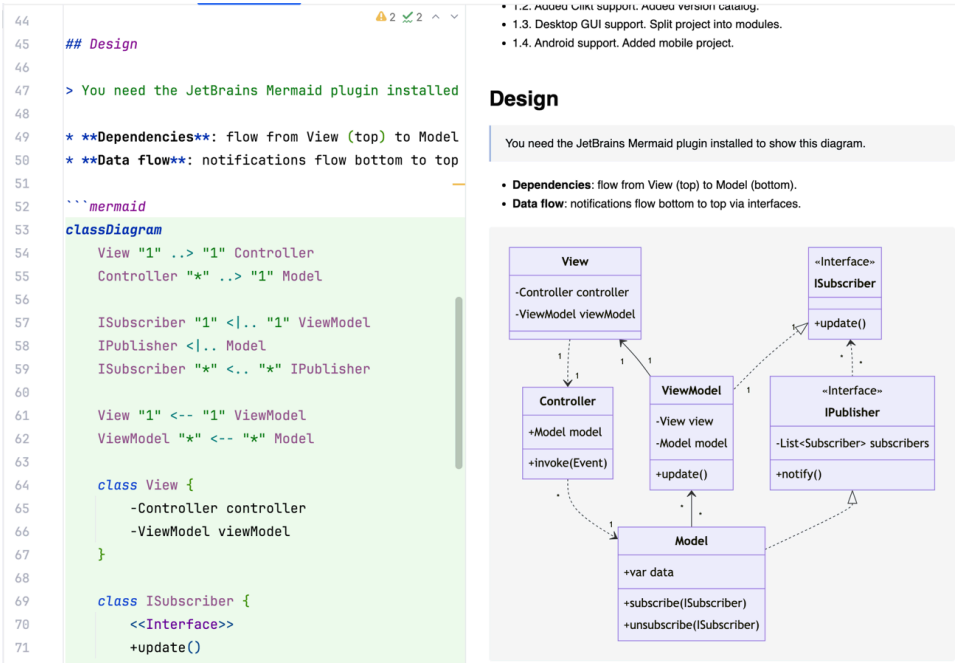

Figure 6: This is an example of a Markdown document containing Mermaid syntax to generate class diagrams. Developer documentation is much easier to produce when you have the correct tools.

Figma (Prototyping)

A prototype is a mock-up of your solution, that is built to demonstrate functionality and elicit feedback from your users.

It helps you determine

- Which features are interactive, and how the user can utilize them.
- What input is required for a screen, what output makes sense.
- The order of screens! How the user will navigate (does it make sense?)

Prototypes are useful to show to your user and walk through all the features to get initial “buy in”.

- You will not get everything right! The goal is to get feedback and make corrections earlier in the cycle.

Figma (Prototyping)

Low-Fidelity is Key

Fidelity refers to the level-of-detail of your prototype.

Low-fidelity prototypes are deliberately simple, low-tech, and represent a minimal investment.

- You can sketch them on paper.
- Many online tools help you build wireframe diagrams that you can demo e.g., Figma.
- Higher fidelity can be semi-interactive to test progression through the interface.

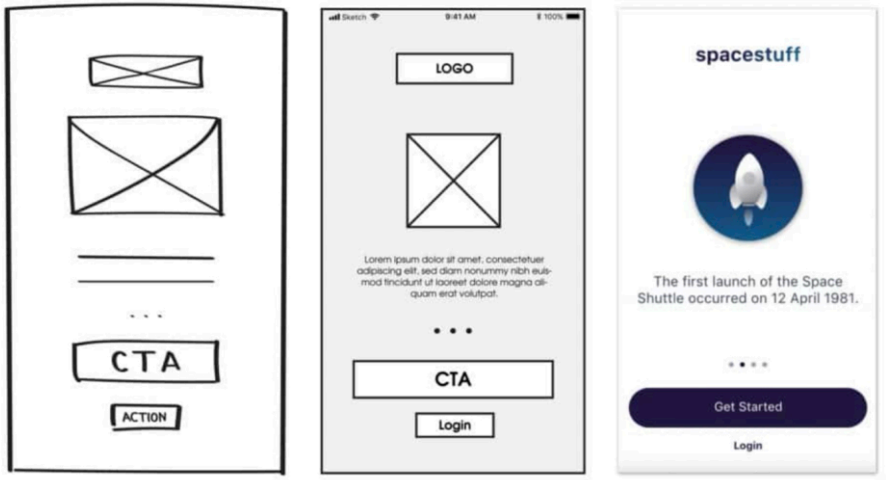

Figure 7: Low-fidelity prototypes provide the best value for your investment.

Figma (Prototyping)

Figma

Figma is a very popular prototyping tool (desktop, mobile, web).

- Used to mock screens & interactions.
- Can build specific UI designs (e.g., iPhone, iPad, desktop).
- Iteration is much easier compared to paper prototyping.

Other options include:

- Omnigraffle (mac)
- Balsamiq (win, mac, web)
- Miro (online, web)

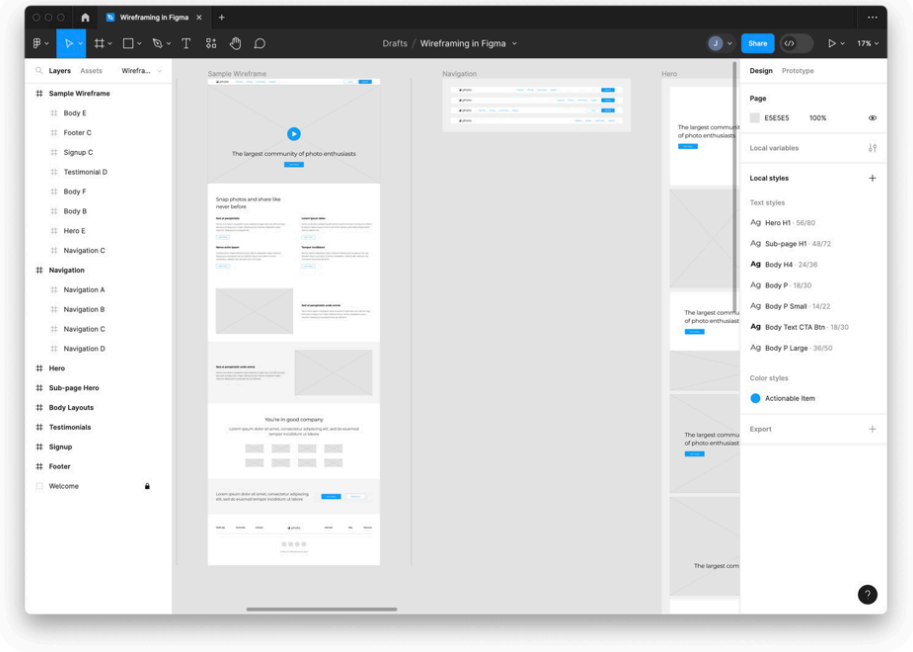

Figure 8: Figma has emerged as the most popular online prototyping tool. It helps that there's a free tier.

Bibliography

[1] E. Holscher, “Docs as Code.” [Online]. Available: https://www.writethedocs.org/guide/docs-as-code/

[2] J. Gruber, “Markdown.” [Online]. Available: https://daringfireball.net/projects/markdown/