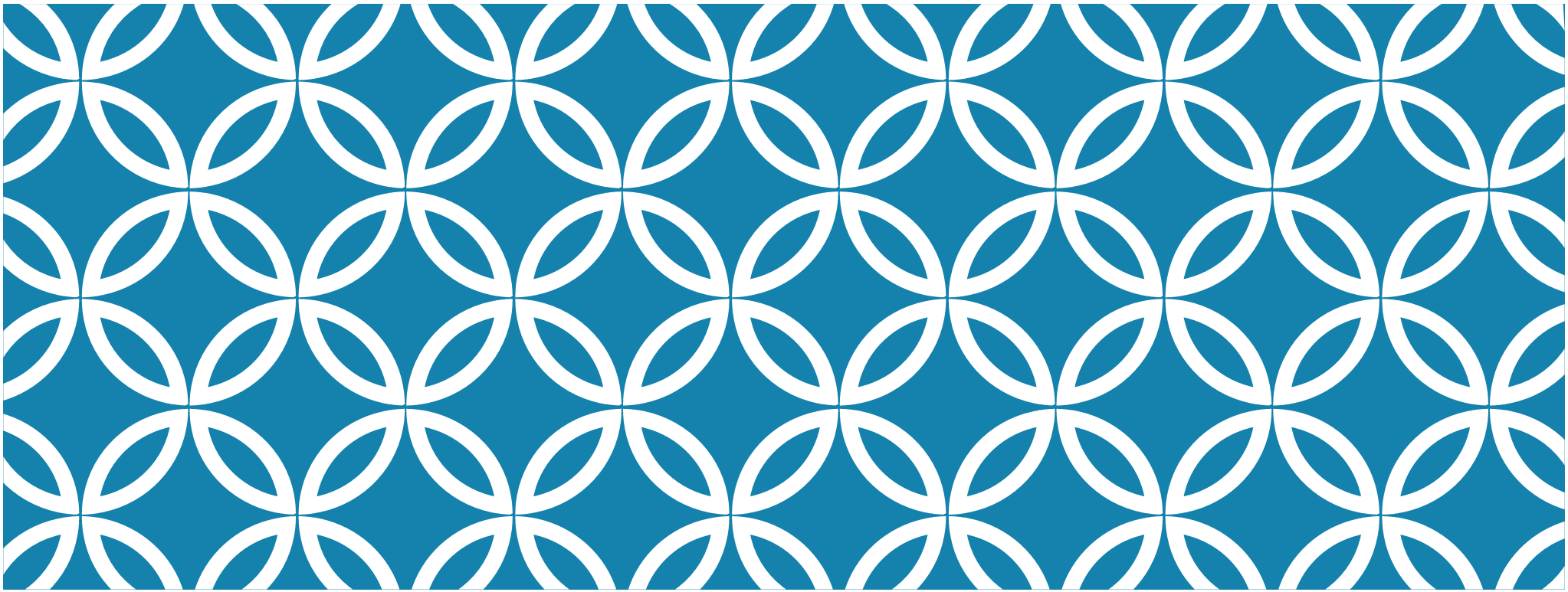


DOCUMENTATION

CS 346: Application Development

WRITING CODE ISN'T ENOUGH

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

- This is the motivation for modular, flexible software design.

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

- The person writing the code may not be the person maintaining it over time.
- Even if you are maintaining code that you wrote, you will not remember your reasons for the design decisions you made 6+ months afterwards.
- Code isn't accessible to everyone in your organization! What about sales? Marketing?

Generating and maintaining documentation is essential to long-term success.

Effective and complete documentation is critical for the *communication of complex ideas*.

WHAT IS DOCUMENTATION?

There are many forms 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

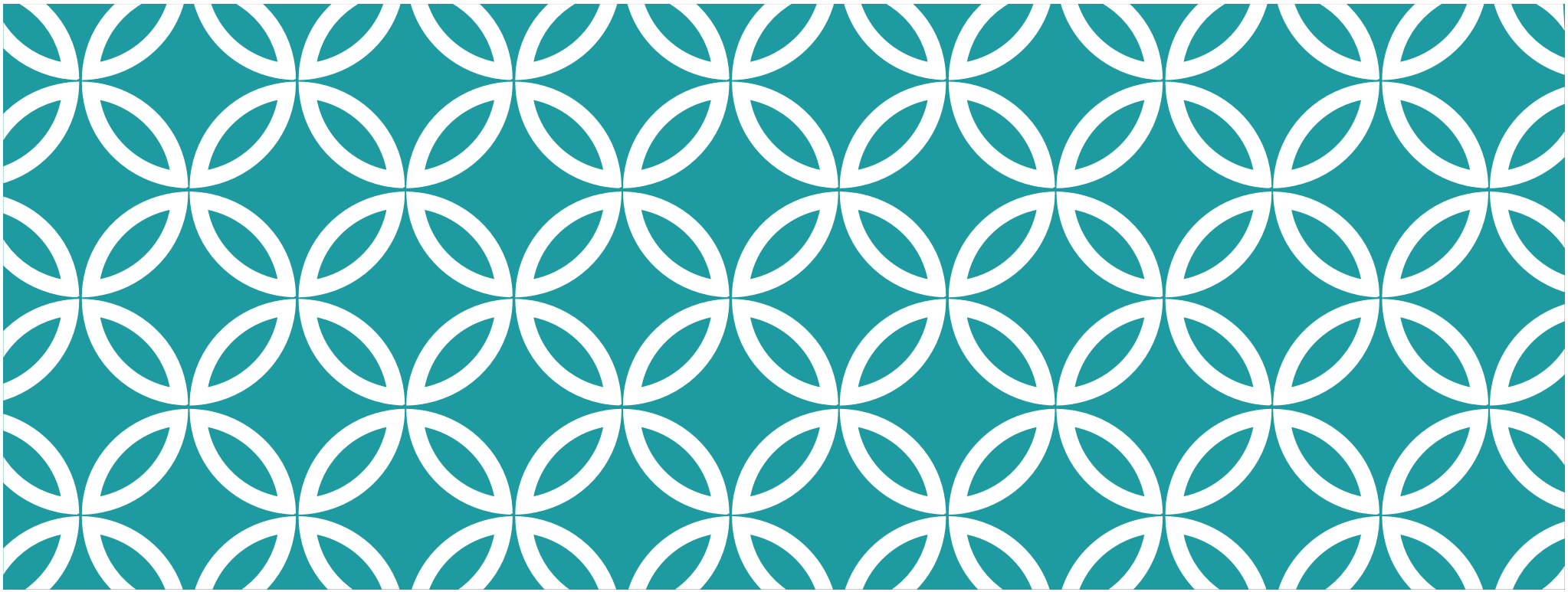
Documentation as Code (*Docs as Code*) refers to a philosophy that you should be writing documentation with the same tools as code.

- Issue Trackers
- Version Control (Git)
- Plain Text Markup (Markdown, reStructuredText, Asciidoc)
- Code Reviews
- Automated Tests

"This [also] 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

www.writethedocs.org



TOOLS > MARKDOWN

CS 346 Application Development

WHAT IS A MARKUP LANGUAGE?

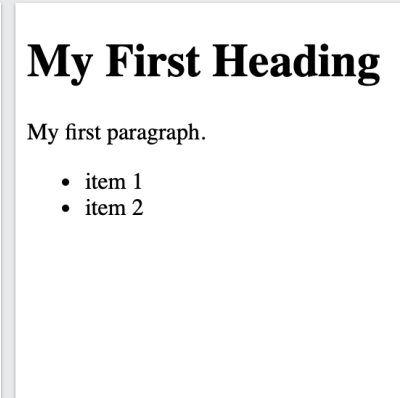
A markup language is a system of annotating a document to describe its structure and presentation. It uses tags or codes to define elements such as headings, paragraphs, lists, images, links, and more. Examples include HTML (Hypertext Markup Language), AsciiDoc, reStructuredText and Markdown.

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>
<ul>
    <li>item 1</li>
    <li>item 2</li>
</ul>

</body>
</html>
```



www.w3schools.com

WHAT IS MARKDOWN?

Markdown is a simple markup language that allows you to add formatting elements to a text file. Markdown was designed with a focus on mapping markup elements to HTML elements, to make conversion to HTML easier (see this blog post from 2004).

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 less formal technical documentation. It is less complete than other markup languages (e.g., AsciiDoc) but is simpler to use.

I think it's safe to claim that developers have been the primary adopters/drivers of Markdown, based on ease-of-use.

```
## Using mdbook
See the [mdbook guide](https://rust-lang.github.io/mdBook/for_developers/index.html)
for information on using `mdbook`.

There are several methods for navigating through the chapters of a book.

* The sidebar on the left provides a list of all chapters. Clicking on any of the chapter titles will load that page.
* The arrow buttons at the bottom of the page can be used to navigate to the previous or the next chapter.

This site supports the following keyboard shortcuts:

* `Arrow-Left`: Navigate to the previous page.
* `Arrow-Right`: Navigate to the next page.
* `t`: Jump to the top of the current page.
* `s`: Jump to the search bar (`ESC` to cancel).
```

Using mdbook

See the mdbook guide ↗ for information on using `mdbook`.

There are several methods for navigating through the chapters of a book.

- The sidebar on the left provides a list of all chapters. Clicking on any of the chapter titles will load that page.
- The arrow buttons at the bottom of the page can be used to navigate to the previous or the next chapter.

This site supports the following keyboard shortcuts:

- `Arrow-Left`: Navigate to the previous page.
- `Arrow-Right`: Navigate to the next page.
- `t`: Jump to the top of the current page.
- `s`: Jump to the search bar (`ESC` to cancel).

The course website is generated from Markdown! It's also used for documentation on GitLab, GitHub etc.

BASIC SYNTAX

Symbol	Meaning
#	Heading 1
##	Heading 2
###	Heading 3
text	*Emphasis*
text	*Emphasis alt.*
text	**Embolden**
* item	Bulleted list
1. item	Numbered list
(title)[URL]	Link to a URL
!(title)[URL]	Embed an image

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.

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
- Works best at generating simple-HTML docs.

HOW DO I USE IT?

Editing Markdown

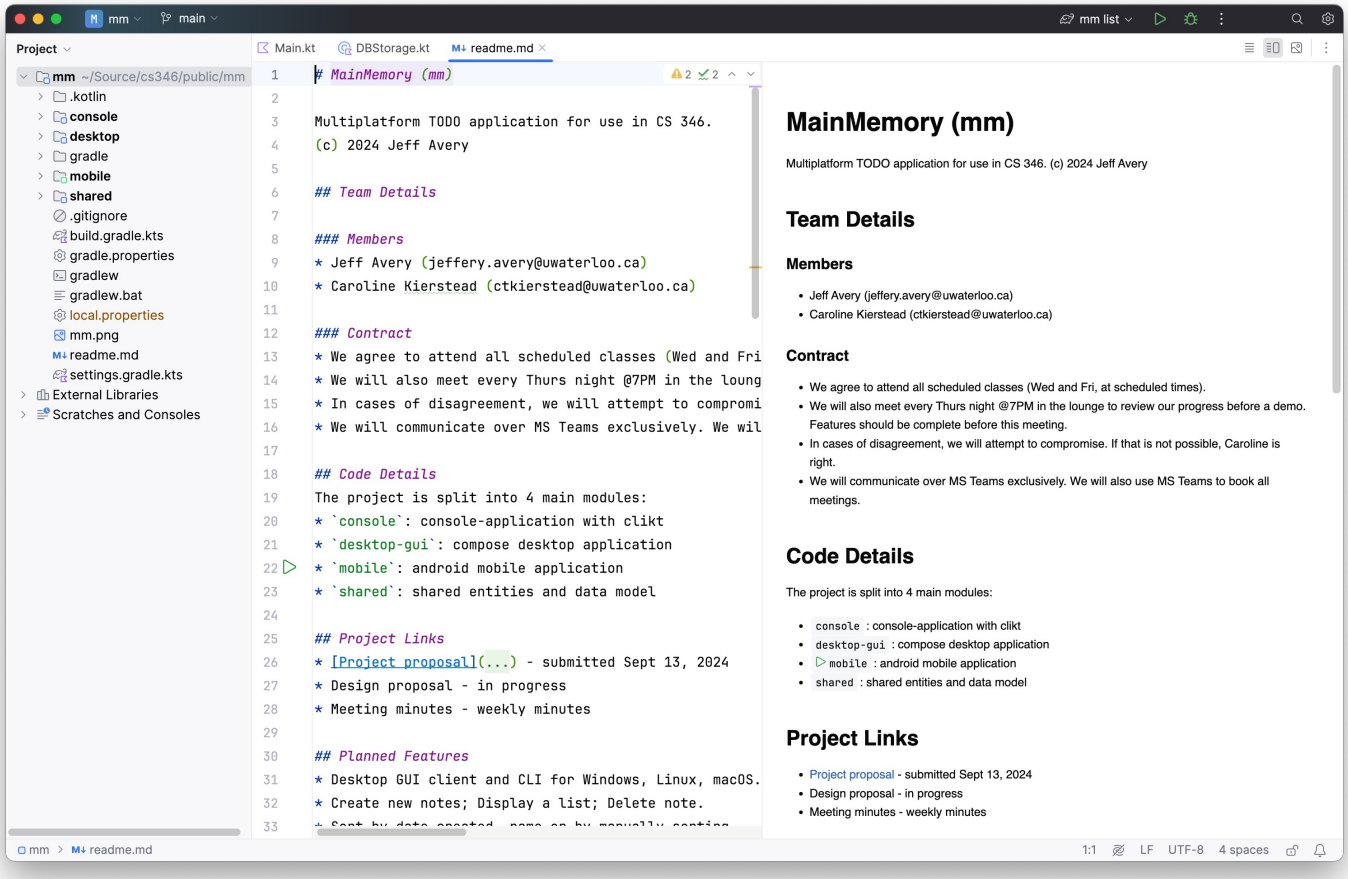
- VS code, and most editors have support for Markdown. This includes syntax highlighting etc.

Integrating into your code/documents:

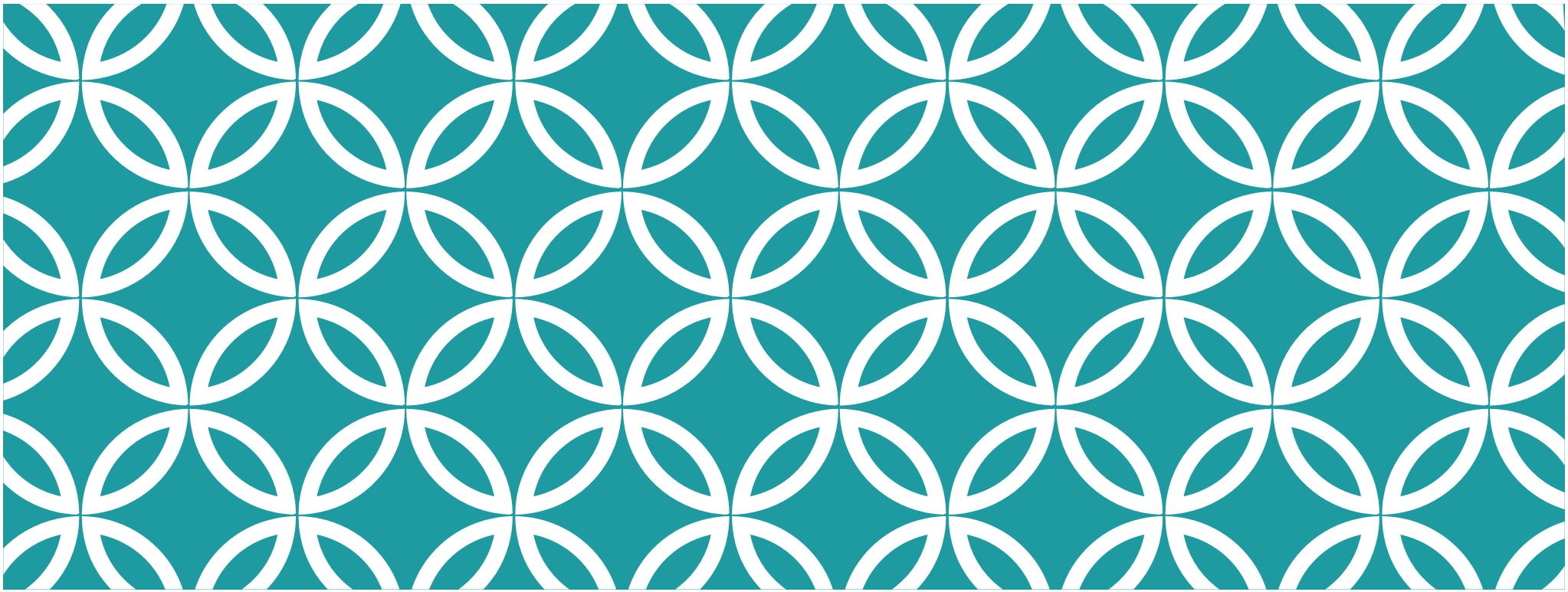
- Online sites like GitLab, GitHub have built-in support i.e., you can enter text as markdown, and it will be shown "pretty-printed" when possible.
- You can even embed diagrams into MD in your code projects!

Generating HTML?

- Tools like `pandoc` and `Marked` can convert markdown to HTML.
- Static site generators: Jekyll, Hugo, Retype all generate websites from markdown.



Most development tools will work with Markdown. IntelliJ IDEA for example has support for Markdown syntax, and will even pretty-print the output.



TOOLS > MERMAID.JS

CS 346 Application Development

DIAGRAMMING

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.

DIAGRAMMING TOOLS

There are many types of diagramming tools. Most will support a range of diagrams or charts like these.

Two main types:

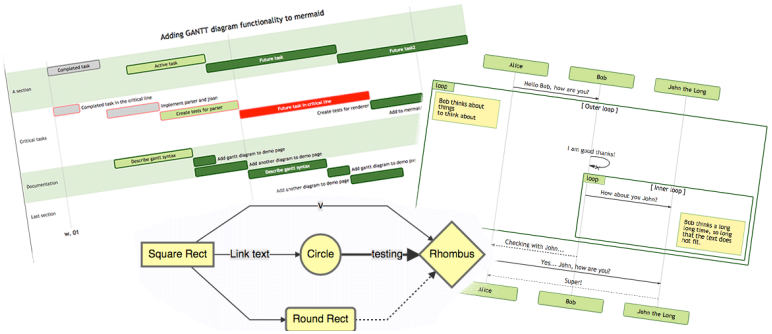
1. **Vector drawing tools**
   - Produce SVG files or a similar format, which you can embed into document as images.
   - Very precise; complete control over the results!
   - e.g., Affinity Designer, Adobe Illustrator

2. **Markup-based drawing tools**
   - You use a markup language to describe your diagram.
   - A diagram "engine" decides on format, layout etc., so it's less precise.
   - e.g., PlantUML, Mermaid.js

MERMAID.JS

Mermaid lets you create diagrams and visualizations using text and code.

It is a JavaScript based diagramming and charting tool that renders Markdown-inspired text definitions to create and modify diagrams dynamically.

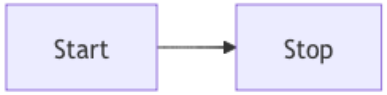


-- Mermaid.js.org

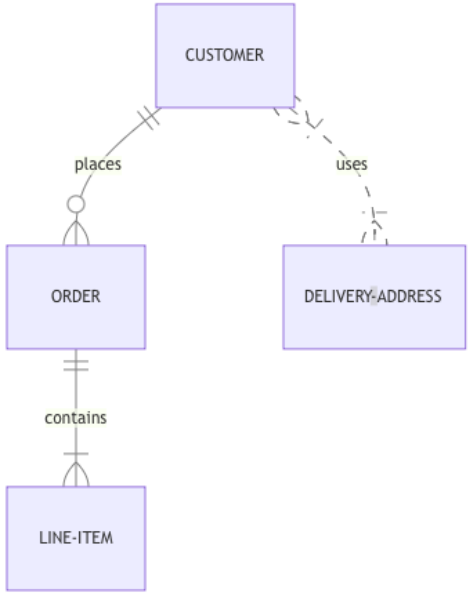
Mermaid supports a HUGE range of diagrams, including all UML diagrams, project charts, etc.

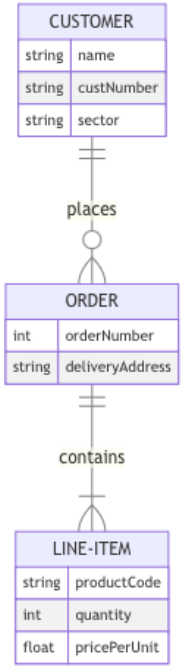
MERMAID.JS DIAGRAM SYNTAX

```mermaid
flowchart
    LR Start --> Stop
```

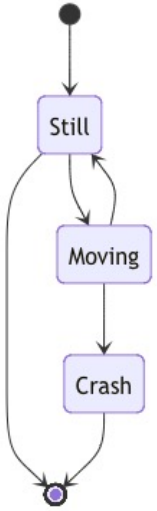



```mermaid
erDiagram
    CUSTOMER ||--o{ ORDER : places
    ORDER ||--|{ LINE-ITEM : contains
    CUSTOMER }|..|{ DELIVERY-ADDRESS : uses
```

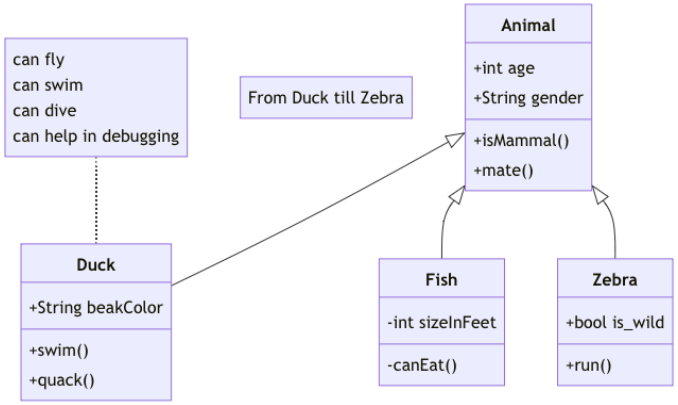




er diagram

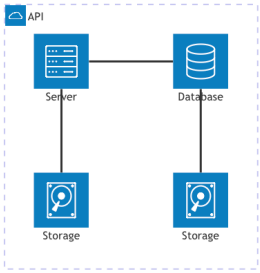


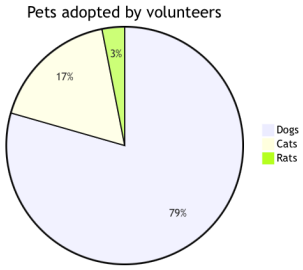
state diagram



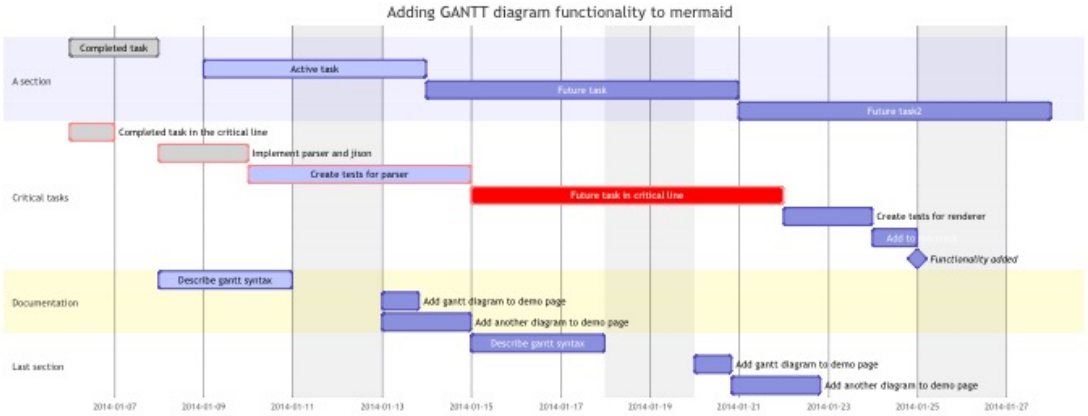
class diagram

See [Mermaid.js documentation] for examples





pie chart



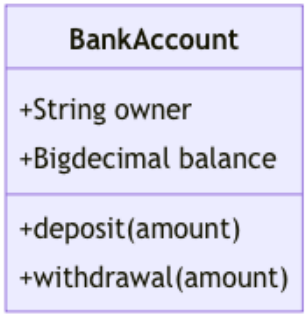
Gantt chart

MERMAID.JS + MARKDOWN

Most environments that support Markdown also support Mermaid IN Markdown.

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

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



You can wrap Mermaid expressions in code blocks in your Markdown documents and they will be rendered inline.

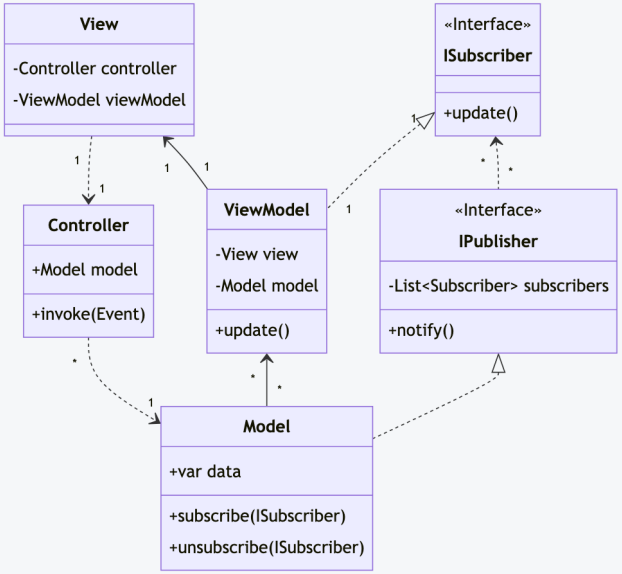
```
44
45  ## Design
46
47  > You need the JetBrains Mermaid plugin installed
48
49  * **Dependencies**: flow from View (top) to Model
50  * **Data flow**: notifications flow bottom to top
51
52  ```mermaid
53  classDiagram
54      View "1" ..> "1" Controller
55      Controller "*" ..> "1" Model
56
57      ISubscriber "1" <|.. "1" ViewModel
58      IPublisher <|.. Model
59      ISubscriber "*" <.. "*" IPublisher
60
61      View "1" <-- "1" ViewModel
62      ViewModel "*" <-- "*" Model
63
64      class View {
65          -Controller controller
66          -ViewModel viewModel
67      }
68
69      class ISubscriber {
70          <<Interface>>
71          +update()
```

- 1.2. Added Clikt support. Added version catalog.
- 1.3. Desktop GUI support. Split project into modules.
- 1.4. Android support. Added mobile project.

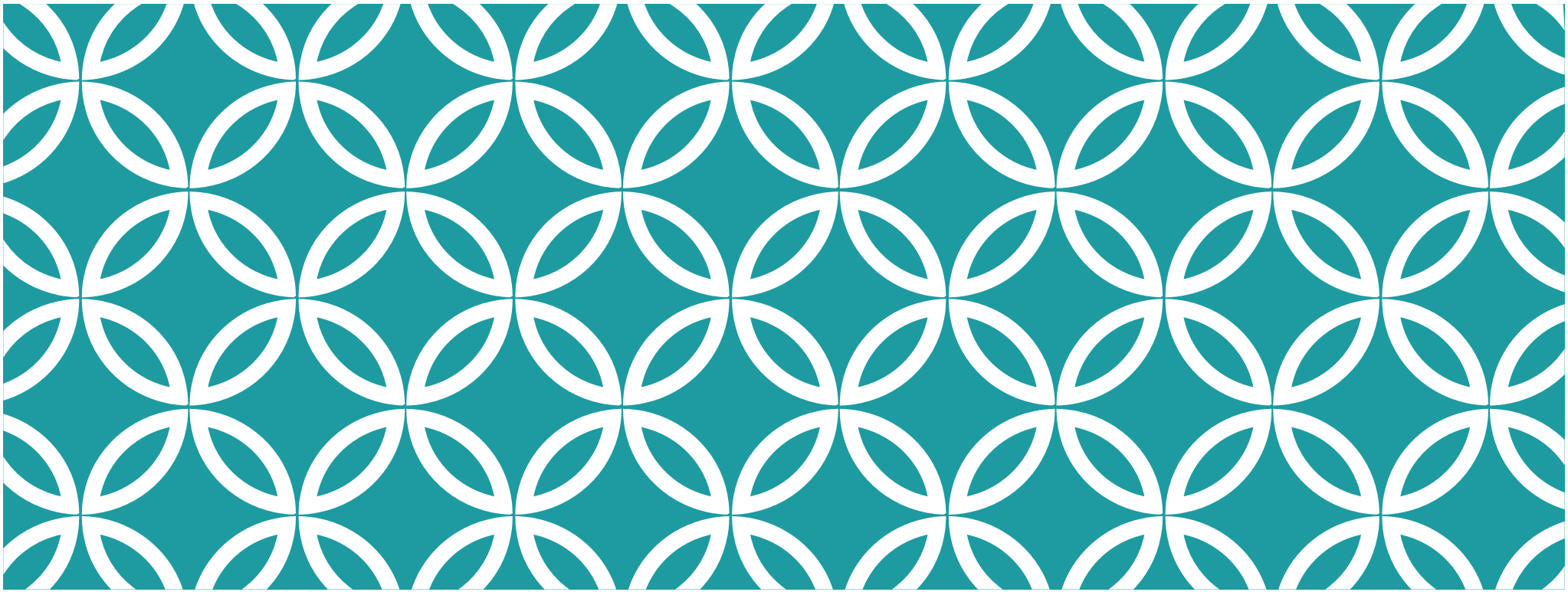
Design

> You need the JetBrains Mermaid plugin installed to show this diagram.

- **Dependencies**: flow from View (top) to Model (bottom).
- **Data flow**: notifications flow bottom to top via interfaces.



Most tools support Mermaid diagrams in Markdown documents. IntelliJ IDE above shows this diagram inline with Markdown documentation.



PROJECT DOCUMENTATION

CS 346 Application Development

WHAT (MINIMAL) DOCS DO YOU NEED?

See the Final Release page.

README
- "Why does this project exist?", "What problem does it solve?", "How do I use it?"
- Video/screenshots/description recommended.
- Link to other topics: installation instructions, user-guide, releases (release notes/installers).

Design document
- Diagrams! Architecture, class hierarchy and detailed class diagrams for data/business classes.

Source code
- Inline code comments and useful git commit messages.

Release notes
- Should help users clearly identify changes and make it obvious how to install/upgrade.

User guide
- Consider expanding with more detailed instructions than what you had in the README.
- "Is this sufficient to explain how to use the application"? Balance with feature discovery.