Design Docs

17-313 Fall 2024

Foundations of Software Engineering

https://cmu-17313q.github.io

Eduardo Feo-Flushing

Administrivia

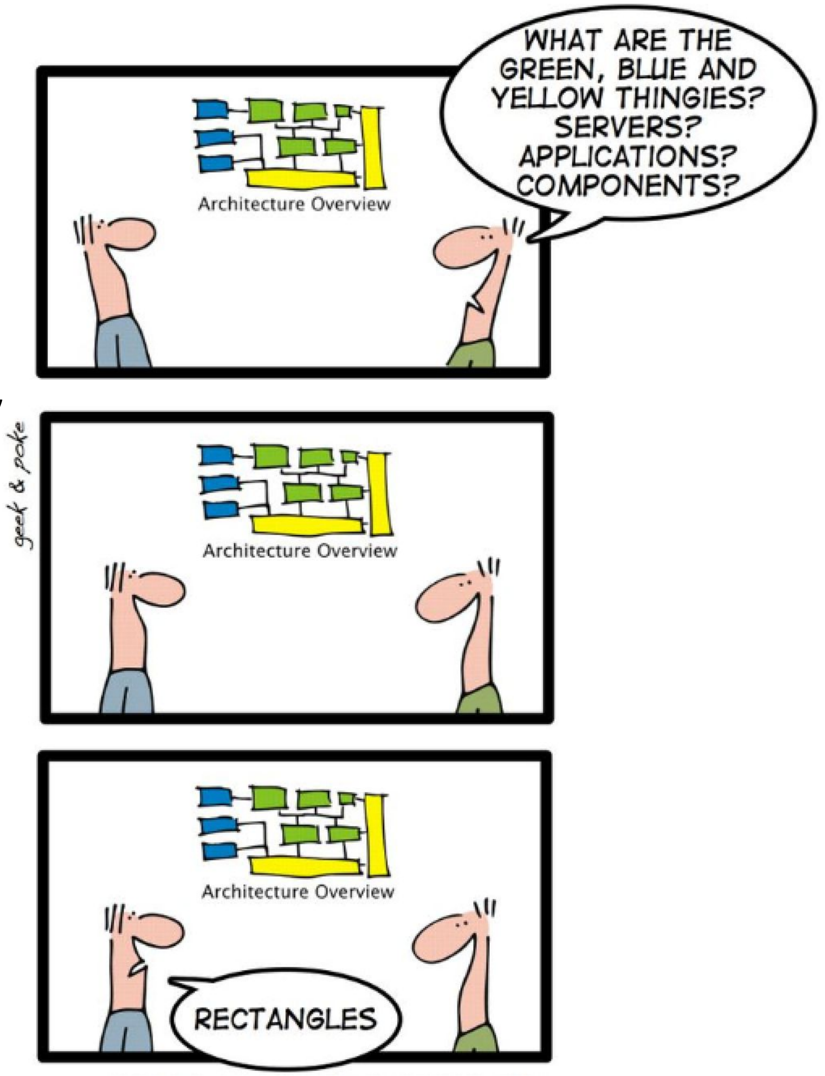
- P2B due **Thursday, Sep 26th, 11:59pm**
- Reminder: Please do not submit participation exercises for people who are not participating.

Smoking Section

- Last full row

Guidelines for selecting a notation

- Suitable for purpose
- Often visual for compact representation
- Usually, boxes and arrows
- UML possible (semi-formal), but possibly constraining
  - Note the different abstraction level – Subsystems or processes, not classes or objects
- Formal notations available
- Decompose diagrams hierarchically and in views
- Always include a legend
- Define precisely what the boxes mean
- Define precisely what the lines mean
- Do not try to do too much in one diagram
  - Each view of architecture should fit on a page
  - Use hierarchy



Examples

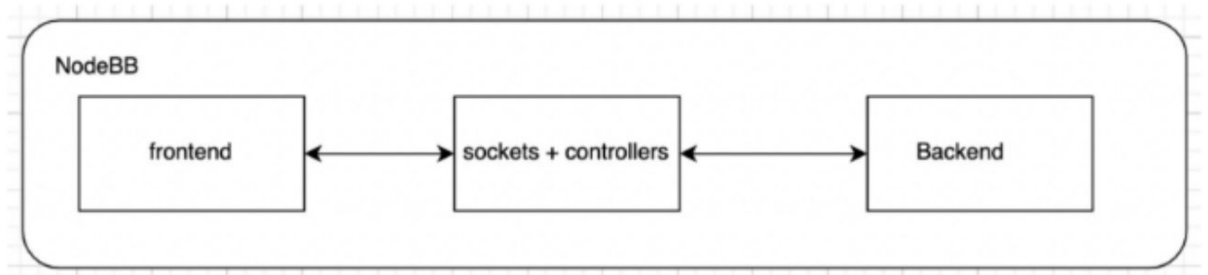
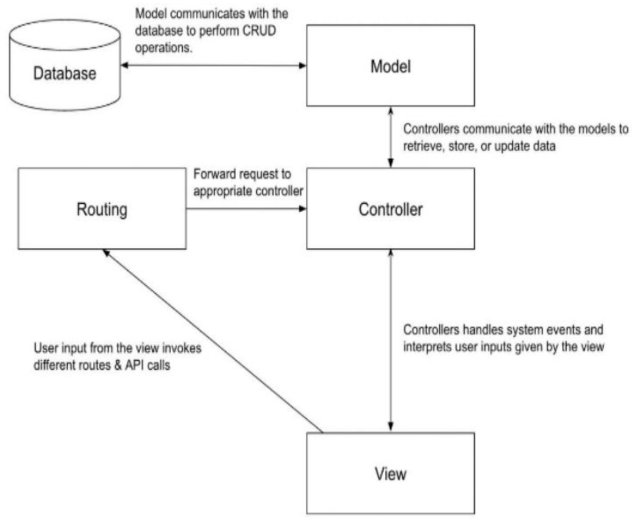
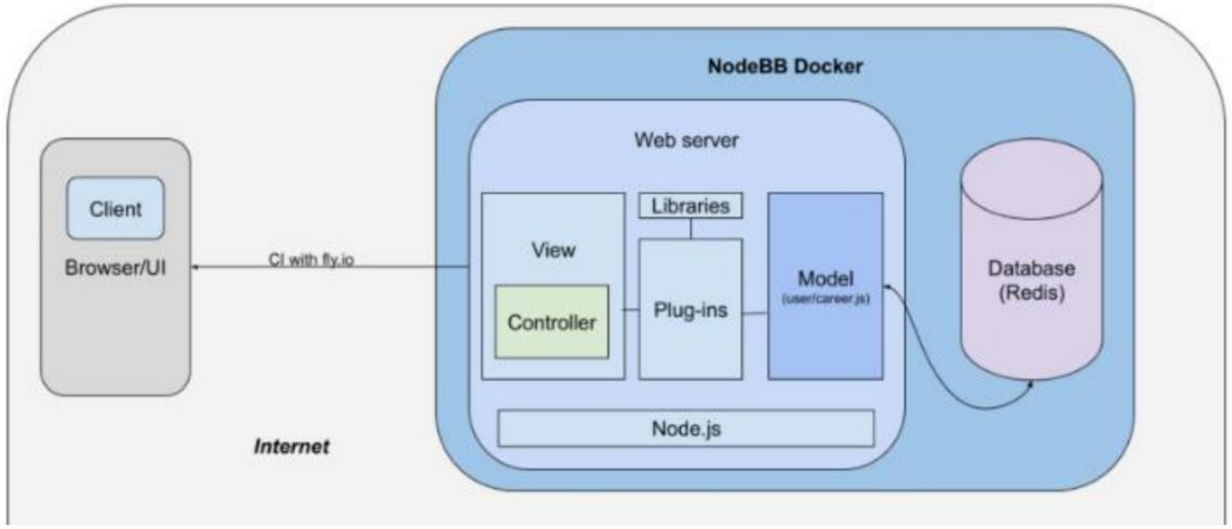


Figure 1: Architecture Diagram of Current State of NodeBB

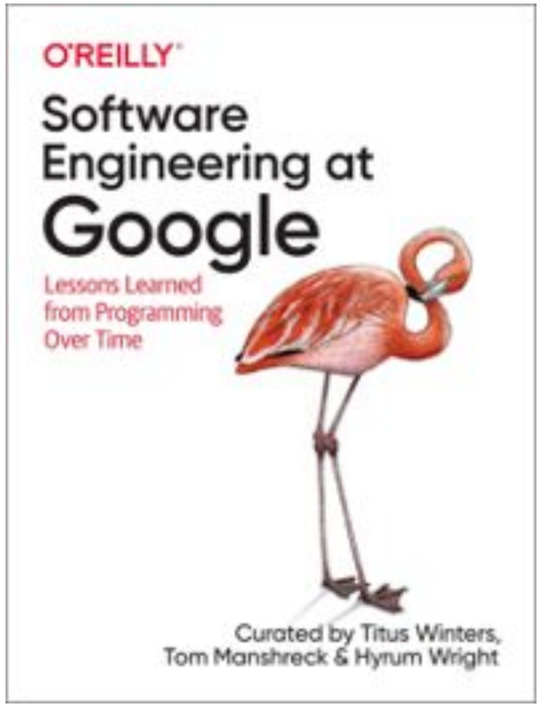
VS


Model communicates with the database to perform CRUD operations.
Database
Model
Controllers communicate with the models to retrieve, store, or update data
Forward request to appropriate controller
Routing
Controller
Controllers handles system events and interprets user inputs given by the view
User input from the view invokes different routes & API calls
View



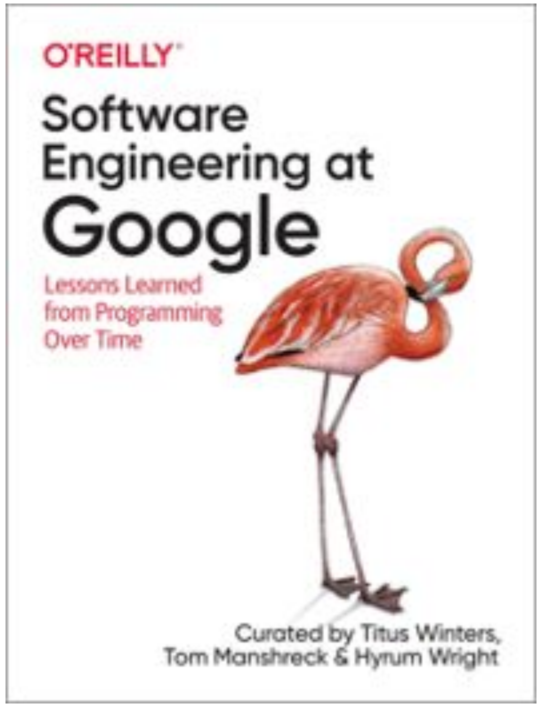
Types of documentation

- Reference documentation (incl. code comments)
- Design documents
- Tutorials
- Conceptual documentation
- Landing pages


O'REILLY
Software Engineering at Google
Lessons Learned from Programming Over Time
Curated by Titus Winters, Tom Manshreck & Hyrum Wright

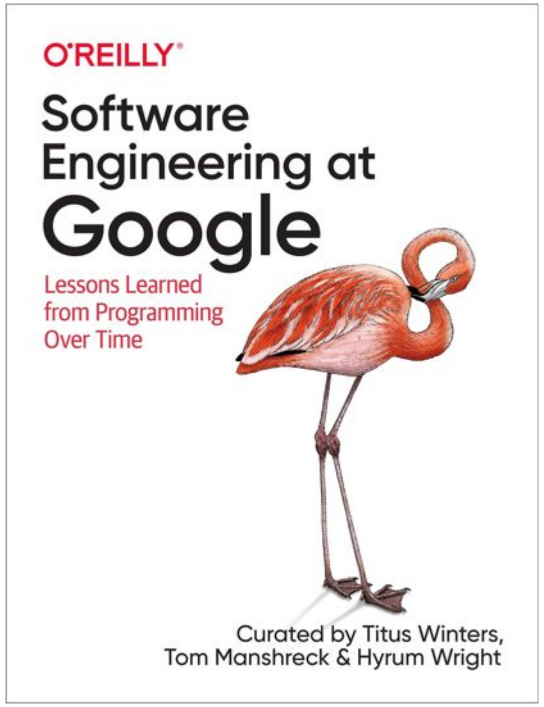
Design documents

- Code review before there is code!
- Collaborative (Google Docs)
- Ensure various concerns are covered, such as: security implications, internationalization, storage requirements, and privacy concerns.
- A good design doc should cover
    - Goals and use cases for the design
    - Implementation ideas (not too specific!)
    - Propose key design decisions with an emphasis on their individual tradeoffs


O'REILLY
Software Engineering at Google
Lessons Learned from Programming Over Time
Curated by Titus Winters, Tom Manshreck & Hyrum Wright

Design Documents

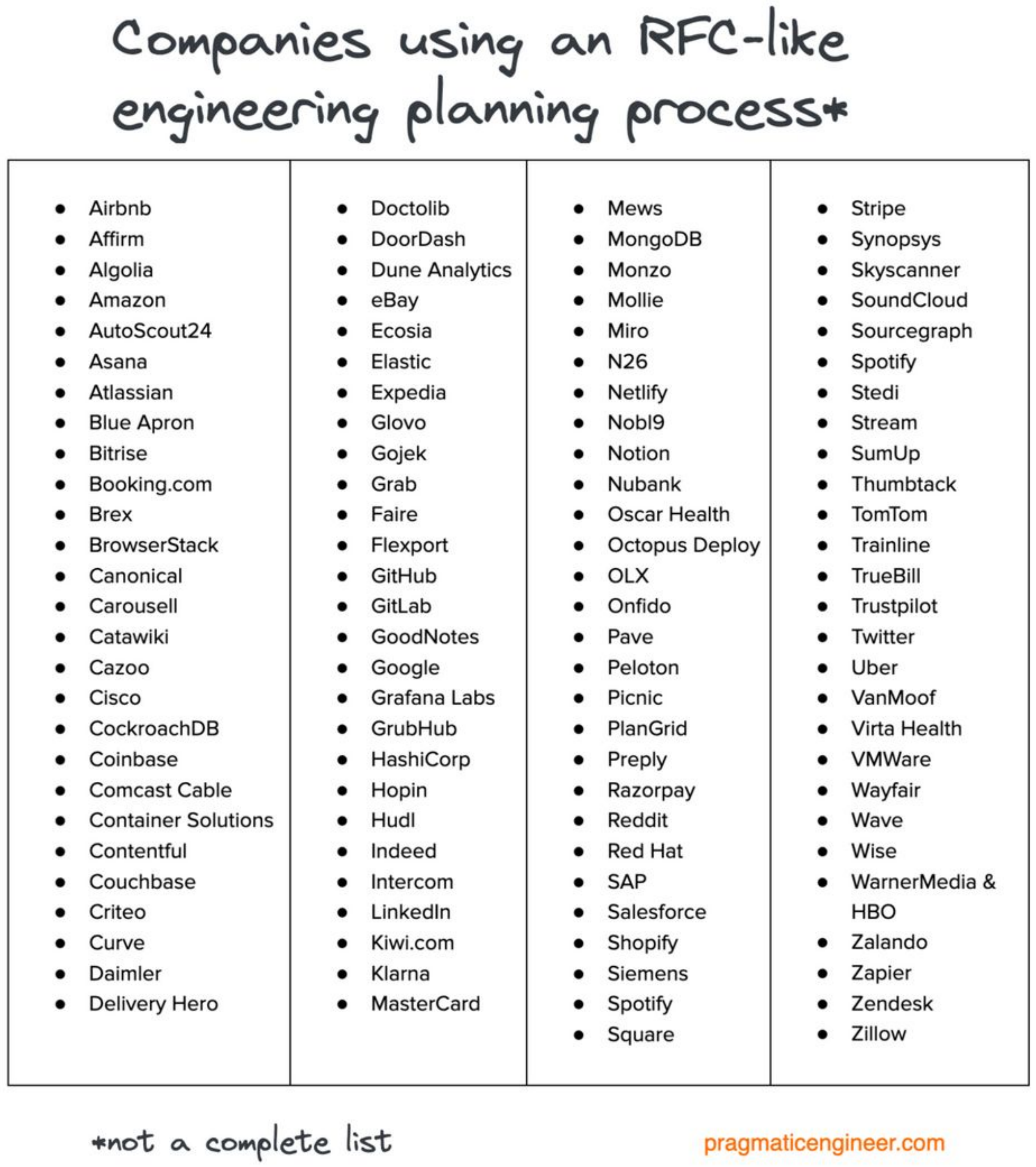
- The *best* design docs suggest design goals, and cover alternative designs, documenting the strengths and weaknesses of each.
- The *worst* design docs accidentally embed ambiguities, which cause implementers to develop contradictory solutions that the customer doesn't want.


O'REILLY

Software Engineering at Google

Lessons Learned from Programming Over Time

Curated by Titus Winters, Tom Manshreck & Hyrum Wright



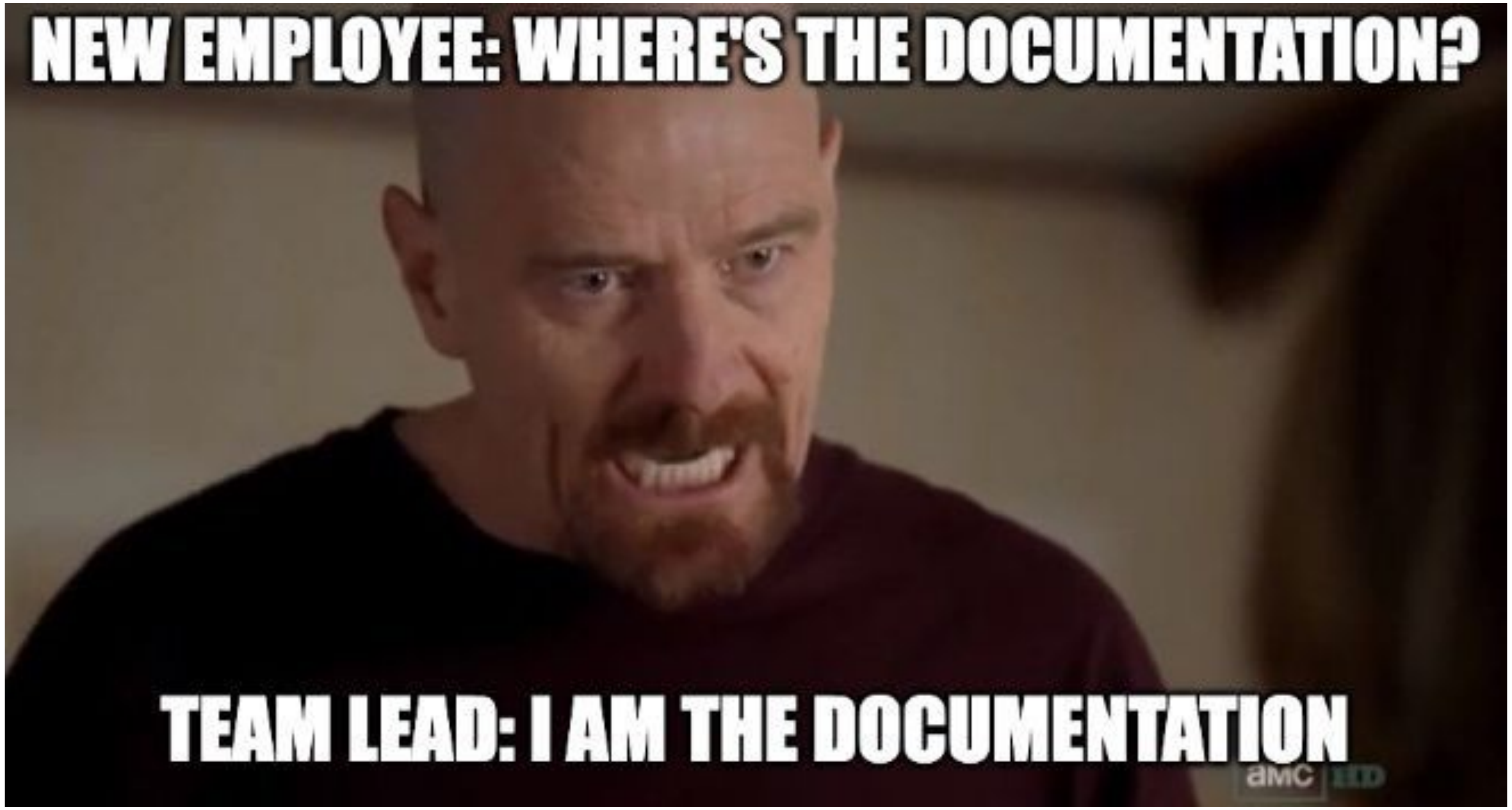
Companies using an RFC-like engineering planning process*

Airbnb	Doctolib	Mews	Stripe
Affirm	DoorDash	MongoDB	Synopsys
Algolia	Dune Analytics	Monzo	Skyscanner
Amazon	eBay	Mollie	SoundCloud
AutoScout24	Ecosia	Miro	Sourcegraph
Asana	Elastic	N26	Spotify
Atlassian	Expedia	Netlify	Stedi
Blue Apron	Glovo	Nobl9	Stream
Bitrise	Gojek	Notion	SumUp
Booking.com	Grab	Nubank	Thumbtack
Brex	Faire	Oscar Health	TomTom
BrowserStack	Flexport	Octopus Deploy	Trainline
Canonical	GitHub	OLX	TrueBill
Carousell	GitLab	Onfido	Trustpilot
Catawiki	GoodNotes	Pave	Twitter
Cazoo	Google	Peloton	Uber
Cisco	Grafana Labs	Picnic	VanMoof
CockroachDB	GrubHub	PlanGrid	Virta Health
Coinbase	HashiCorp	Preply	VMWare
Comcast Cable	Hopin	Razorpay	Wayfair
Container Solutions	Hudl	Reddit	Wave
Contentful	Indeed	Red Hat	Wise
Couchbase	Intercom	SAP	WarnerMedia & HBO
Criteo	LinkedIn	Salesforce	Zalando
Curve	Kiwi.com	Shopify	Zapier
Daimler	Klarna	Siemens	Zendesk
Delivery Hero	MasterCard	Spotify	Zillow
		Square	

*not a complete list

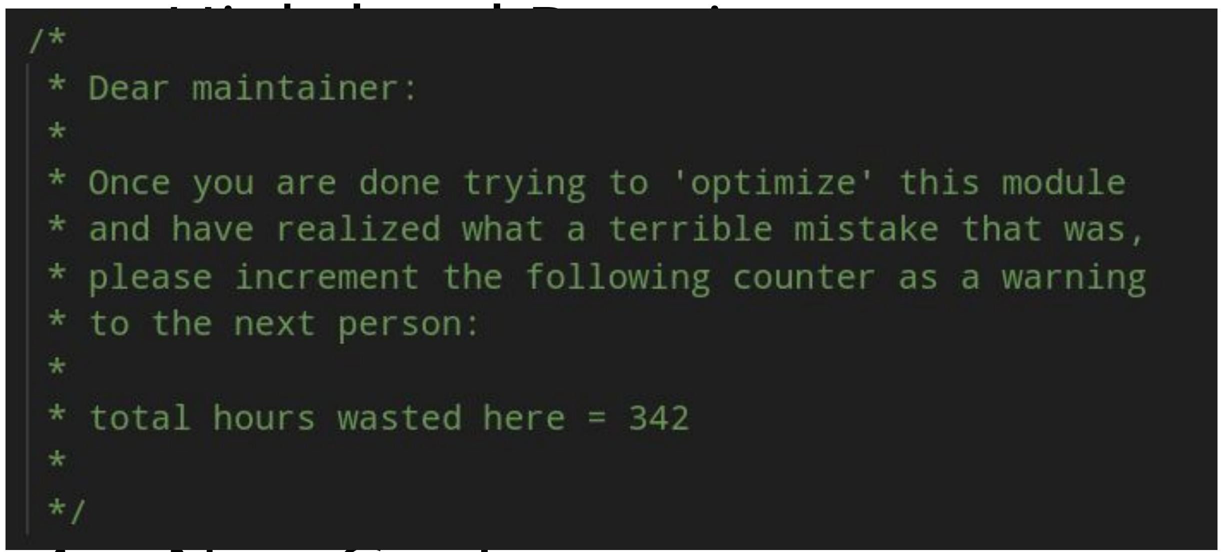
pragmaticengineer.com

Why is this important?


NEW EMPLOYEE: WHERE'S THE DOCUMENTATION?
TEAM LEAD: I AM THE DOCUMENTATION
amc HD

Common parts/templates

1. Metadata: *version, date, authors*
2. Executive Summary: *problem being solved, project mission*
3. Stakeholders (and non-stakeholders)
4. Scenarios / User Stories
5. User Experience



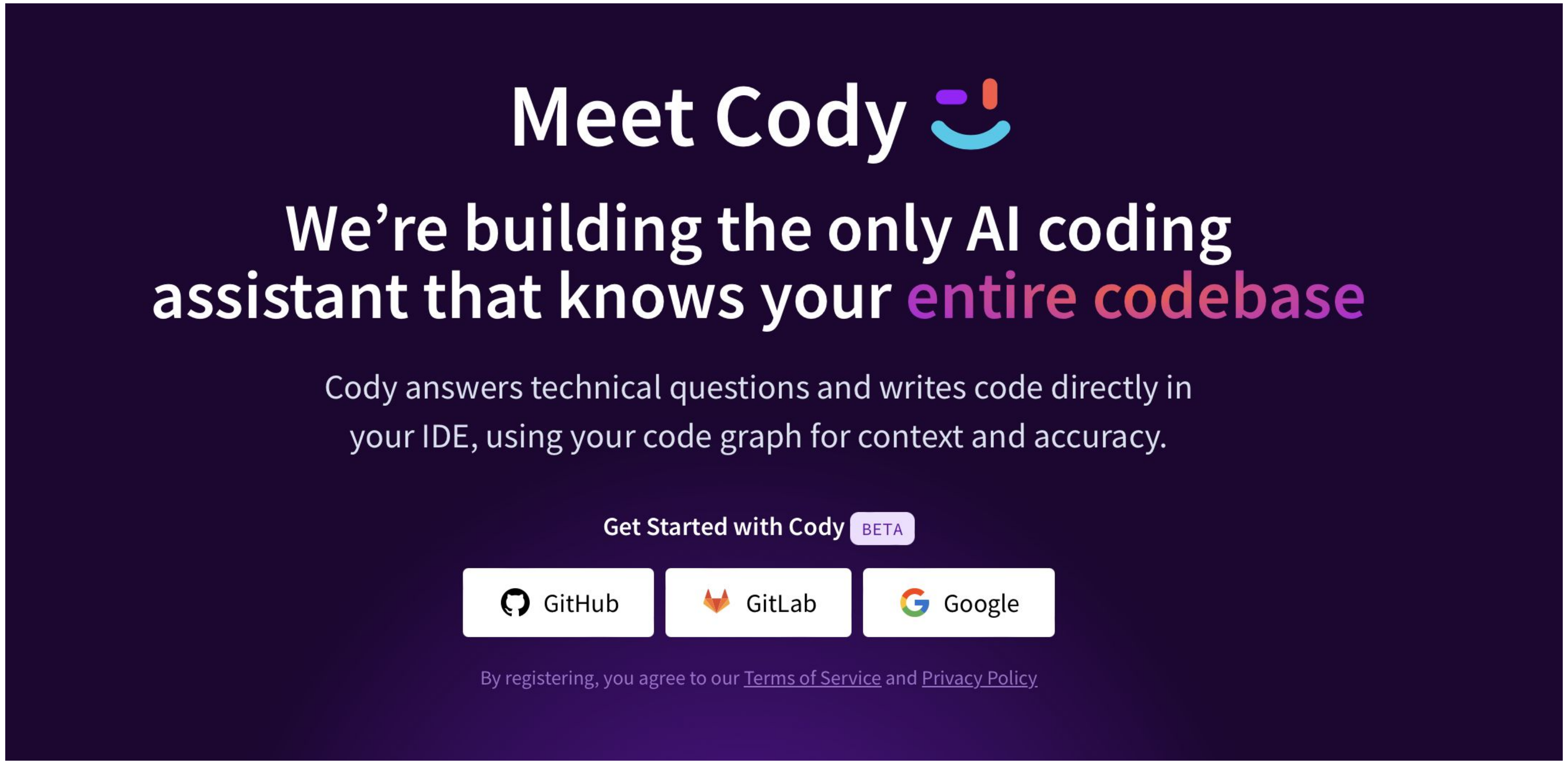
```
/*
 * Dear maintainer:
 *
 * Once you are done trying to 'optimize' this module
 * and have realized what a terrible mistake that was,
 * please increment the following counter as a warning
 * to the next person:
 *
 * total hours wasted here = 342
 *
 */
```

4. Non-Goals
5. Roadmap / Timeline
6. Open Issues

Examples: SourceGraph RFCs

Requests for Comment

Sourcegraph


Meet Cody

We're building the only AI coding assistant that knows your entire codebase

Cody answers technical questions and writes code directly in your IDE, using your code graph for context and accuracy.

Get Started with Cody BETA
GitHub
GitLab
Google
By registering, you agree to our Terms of Service and Privacy Policy

When to use an RFC:

Sourcegraph

- You want to frame a problem and propose a solution.
- You want thoughtful feedback from team members on our globally-distributed remote team.
- You want to surface an idea, tension, or feedback.
- You want to define a project or design brief to drive project collaboration.
- You need to surface and communicate around a highly cross-functional decision with our formal decision-making process.

Don’t use an RFC when

Sourcegraph

- You want to discuss personal or sensitive topics one-on-one with another team member.
- You want to make a decision to change something where you are the decider. In the vast majority of cases, creating an RFC to explain yourself will be overkill. RFCs should only be used if a decision explicitly requires one of the bullets in the previous page.

RFC Labels

Sourcegraph

- **WIP**: The author is still drafting the RFC and it’s not ready for review.
- **Review**: The Review label is used when the RFC is ready for comments and feedback.
- **Approved**: When the RFC is for the purpose of making a decision, the Approved label indicates that the decision has been made.
- **Implemented**: When the RFC is for the purpose of making a decision, the Implemented label indicates that the RFC’s proposal has been implemented.
- **Closed**: When the RFC is for the purpose of collaboration or discussion but not necessarily to make a decision or propose a specific outcome that will eventually become Implemented, the Closed label indicates that the RFC is no longer an active collaborative artifact.
- **Abandoned**: When the RFC is for the purpose of making a decision, and there are no plans to move forward with the RFC’s proposal, the Abandoned label indicates that the RFC has been purposefully set aside.

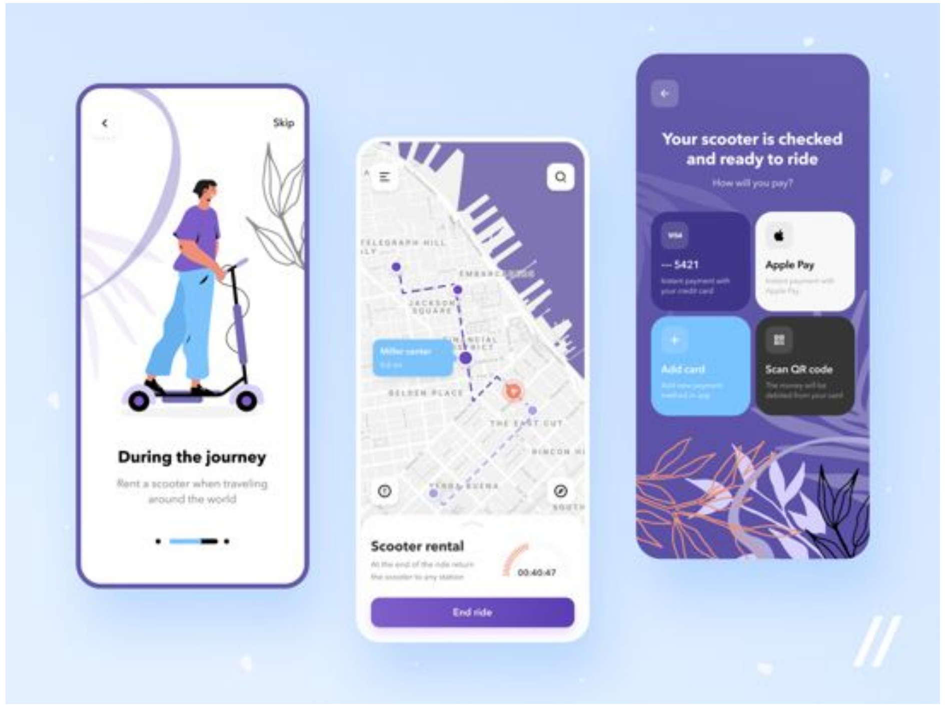
Observe Sourcegraph Design Docs

- Docs are publicly available
  https://drive.google.com/drive/folders/1zP3FxdDlcSQGC1qvM9lHZRaHH4I9Jwwa

- Let's take a look at one!

Exercise

- 3 Proposed Features:
  - Add Payment Method
  - More Secure Authentication
  - Add Android Support



Time to write our own design docs!

- Divide up into 3 sections –NOTE: you should be signed in w/Andrew to google
- Your mission:
  - Brainstorm a feature to add to a scooter app and write a design spec, together, in real time!
  - Review the design doc, collaborate around text
  - Review another team's design doc, ask questions/leave comments

bit.ly/313-designdoc-front

bit.ly/313-designdoc-middle

bit.ly/313-designdoc-back