

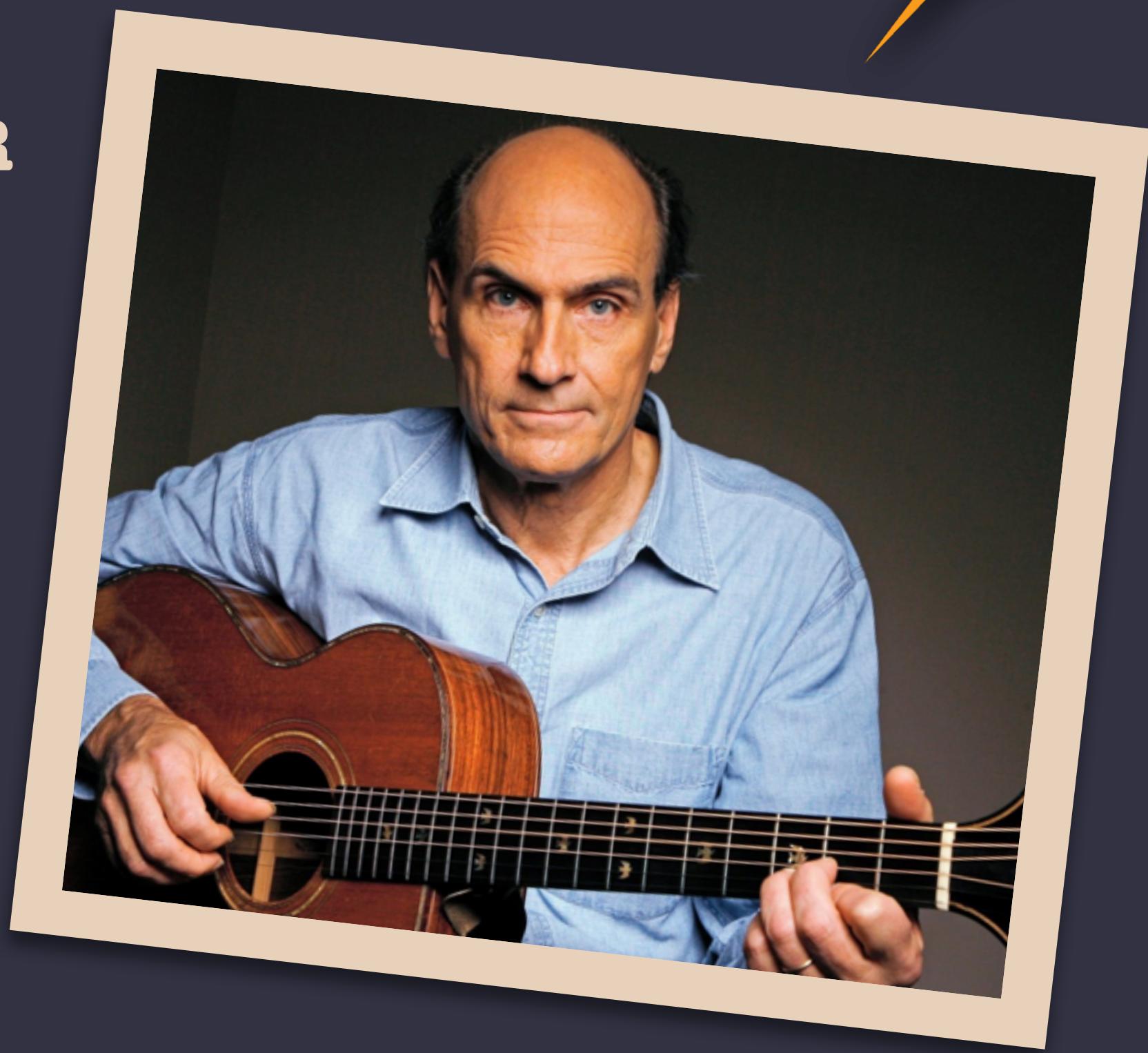
NON-FUNCTIONAL REQUIREMENTS



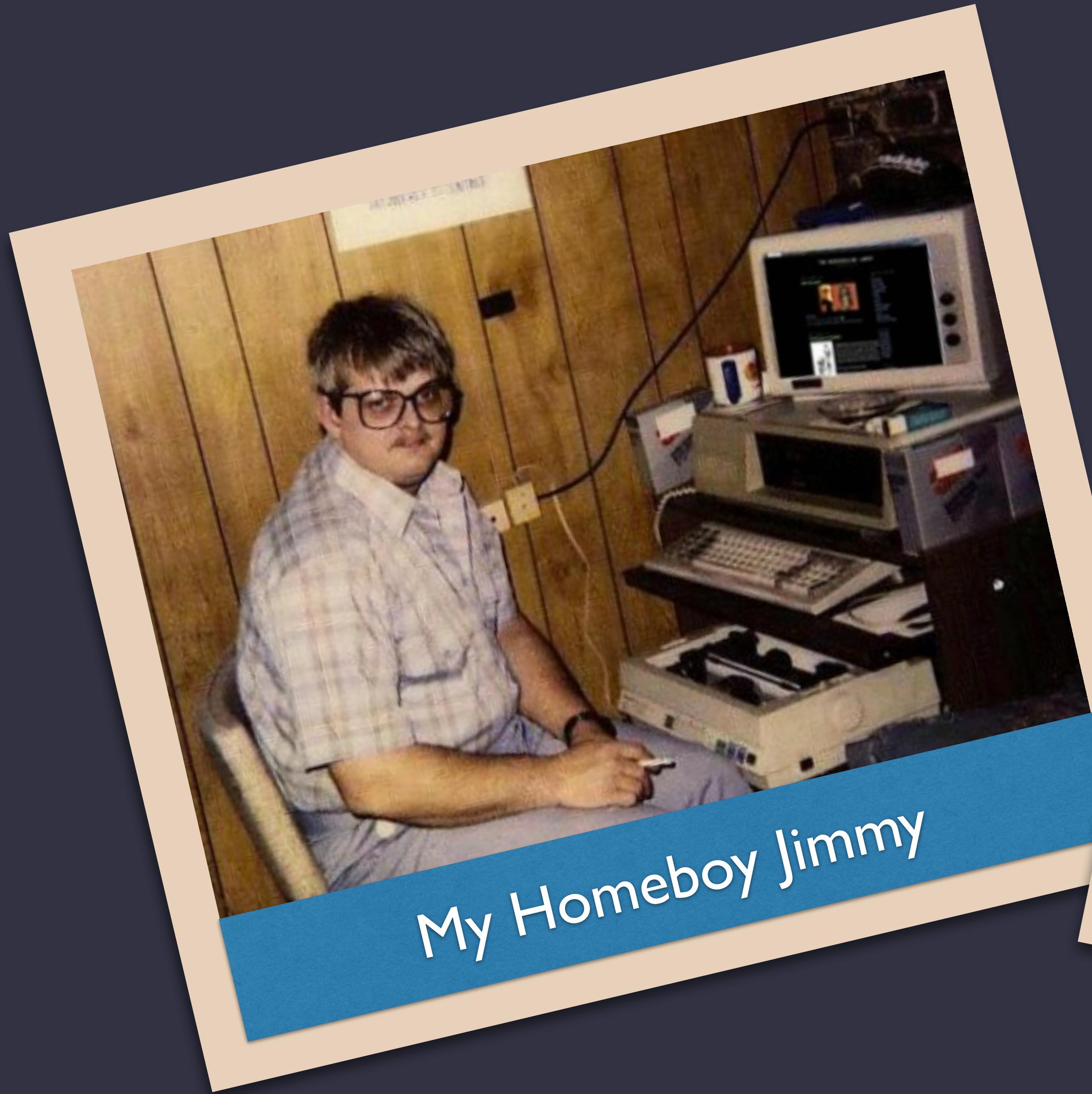
HOW WELL DOES YOUR
SYSTEM DO WHAT IT DOES?

“I WAS A FUNCTIONAL ADDICT.”

- JAMES TAYLOR



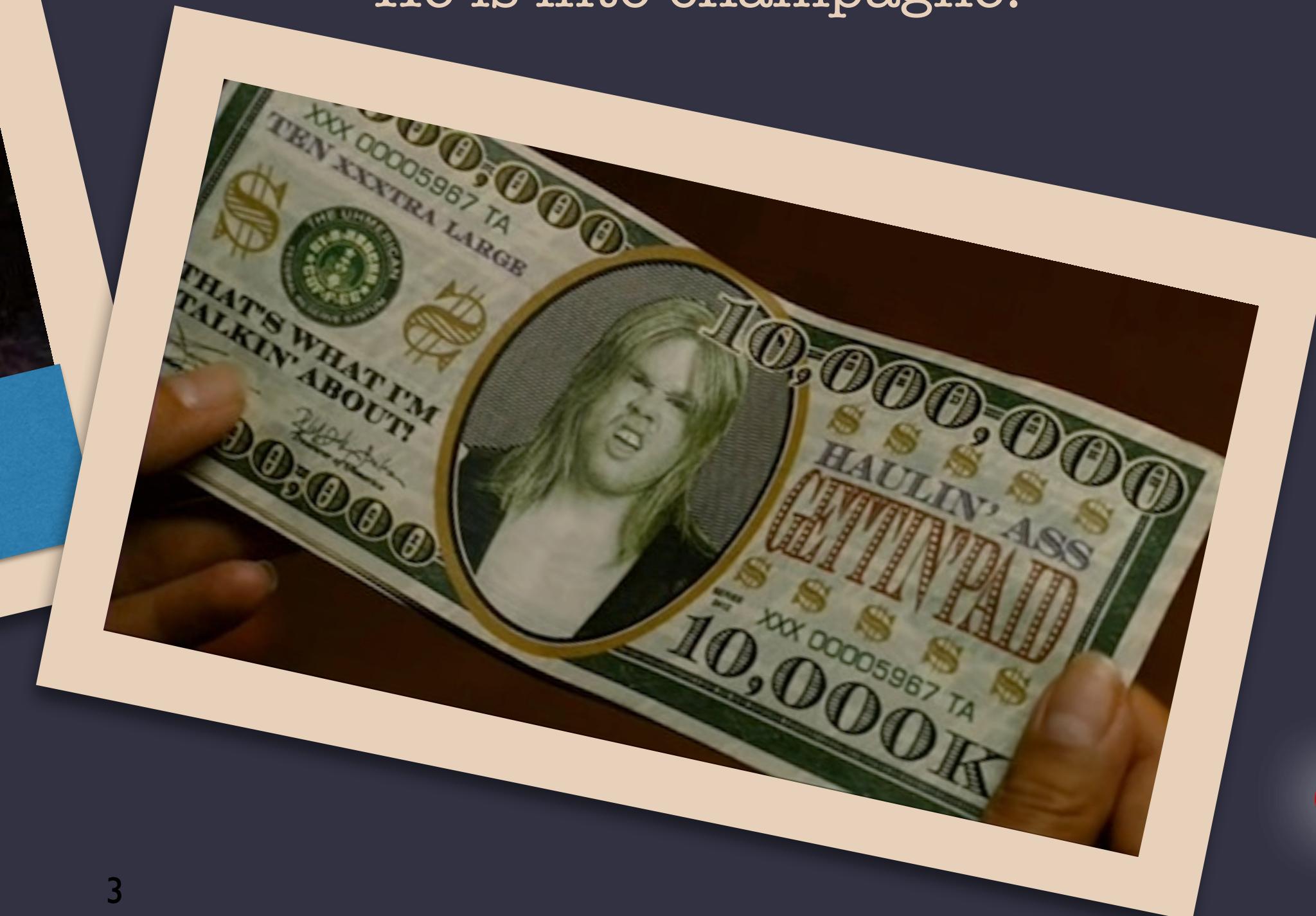
THIS IS JIMMY!



Jimmy is a software engineer.
Jimmy is awesome!

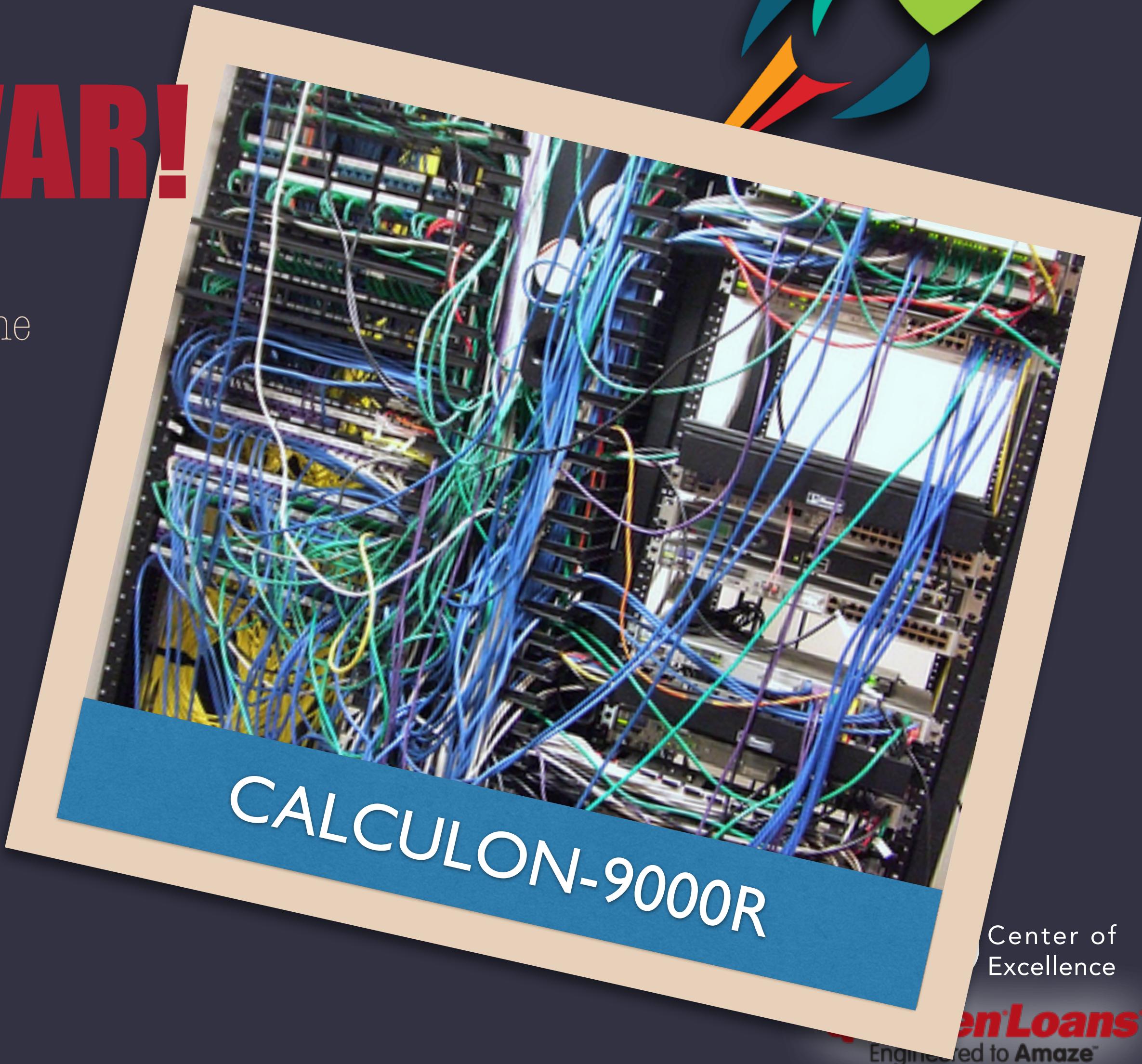
Jimmy likes Pina Coladas
and getting caught in the rain.

He's not into health food.
He is into champagne.



DANG, JIMMY! BEST SYSTEM EVAR!

- ★ The Calculon-9000R is going to revolutionize the entire industry!
- ★ **It performs every mortgage function EVER.**
- ★ An engineering marvel!
- ★ Every edge case is covered!
- ★ It can calculate everything!
- ★ OMG PONIES!
- ★ It's got electrolytes



Center of
Excellence

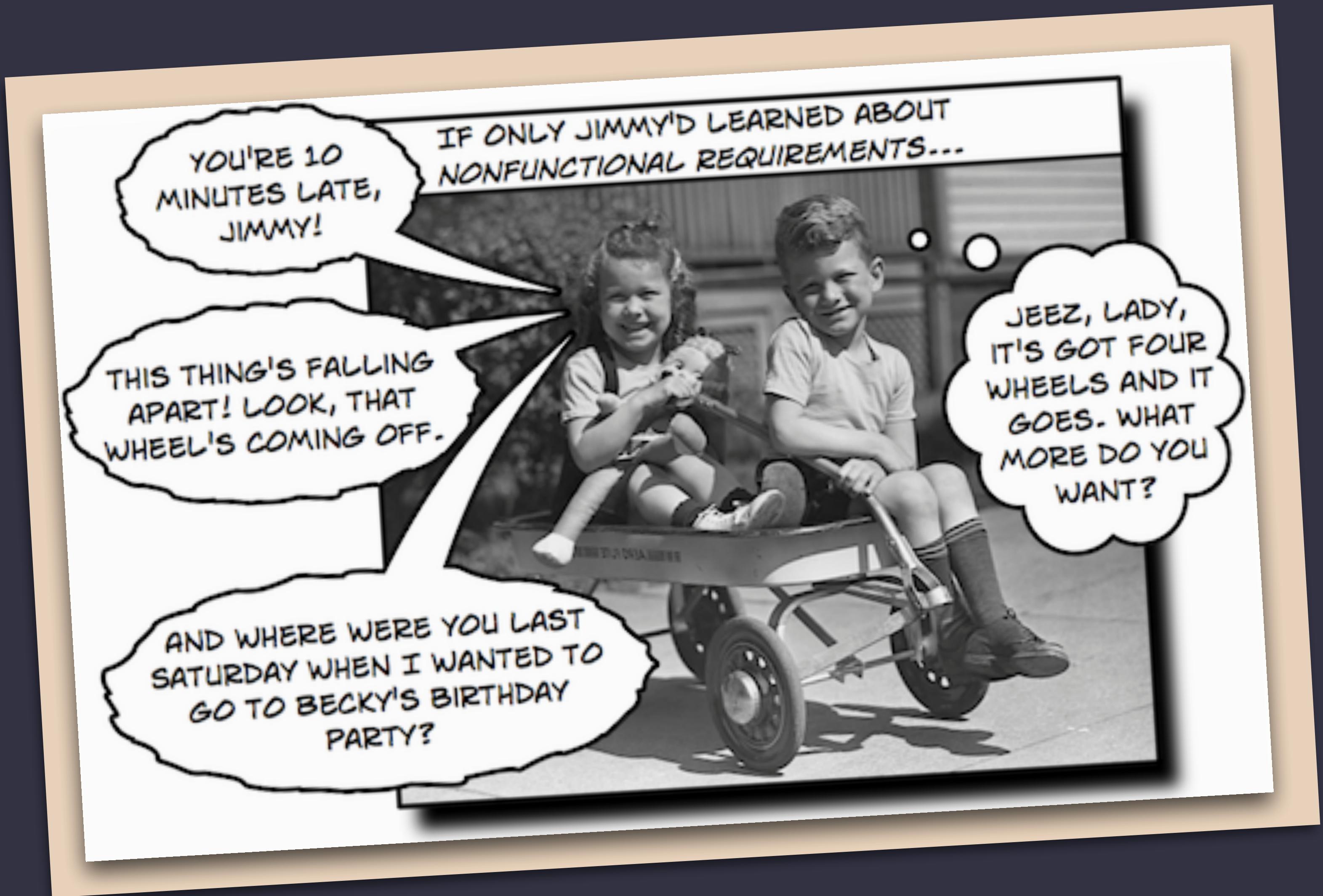
OpenLoans
Engineered to Amaze™

CALCULON-9000R PUT IN PRODUCTION!



WHY IS DAWSON SAD?





**NFR'S ARE NOT ABOUT
WHAT A SYSTEM DOES...**



THEY ARE ABOUT HOW IT DOES IT.

THIS IS A NON-TRIVIAL DIFFERENTIATION.





WHY IS DAWSON SAD?



HEY NOW!

- ★ Where's the documentation?
- ★ Does the system even perform?
- ★ What about testing?
- ★ Is it secure?



Quicken Loans
Engineered to Amaze™

ASPECTS OF SOFTWARE THAT DEFINE HOW IT FITS INTO THE ECOSYSTEM.

This is not a comprehensive list, but instead are examples of non-functional aspects of a system.



IT'S ALL ABOUT THE “ILITIES”

- ★ Security... ummm... securitility
- ★ Maintainability
- ★ Testability
- ★ Scaleability
- ★ Extensibility
- ★ Availability
- ★ Reliability
- ★ ...

- ★ Accessibility
- ★ Accountability
- ★ Adaptability
- ★ Administrability
- ★ Affordability
- ★ Agility
- ★ Availability
- ★ Capability
- ★ Composability
- ★ Configurability
- ★ Compatibility
- ★ Demonstrability
- ★ Deployability
- ★ Durability
- ★ Executability
- ★ Extensibility
- ★ Evolvability
- ★ Fidelity
- ★ Flexibility
- ★ Functionality
- ★ Integratability
- ★ Interoperability
- ★ Interpretability
- ★ Maintainability
- ★ Manageability
- ★ Mobility
- ★ Modifiability
- ★ Operability
- ★ Performability
- ★ Portability
- ★ Practiblity
- ★ Practicality
- ★ Predictability
- ★ Producibility
- ★ Recoverability
- ★ Reliability
- ★ Repeatability
- ★ Responsibility
- ★ Reusability
- ★ Scalability
- ★ Serviceability
- ★ Stability
- ★ Supportability
- ★ Suitability
- ★ Survivability
- ★ Tailorability
- ★ Testability
- ★ Traceability

SO... MANY...



SO... MANY....

- ★ Demonstrability
- ★ Performability
- ★ Maintainability
- ★ Scalability
- ★ Testability
- ★ Understandability
- ★ Trainability
- ★ Vulnerability



SECURITY

- ★ Vulnerability

PERFORMANCE

- ★ Testability
- ★ Performability
- ★ Scalability

DOCUMENTATION

- ★ Understandability
- ★ Maintainability
- ★ Trainability
- ★ Demonstrability

**WE SELECTED THREE INITIAL
NON-FUNCTIONAL REQUIREMENTS**



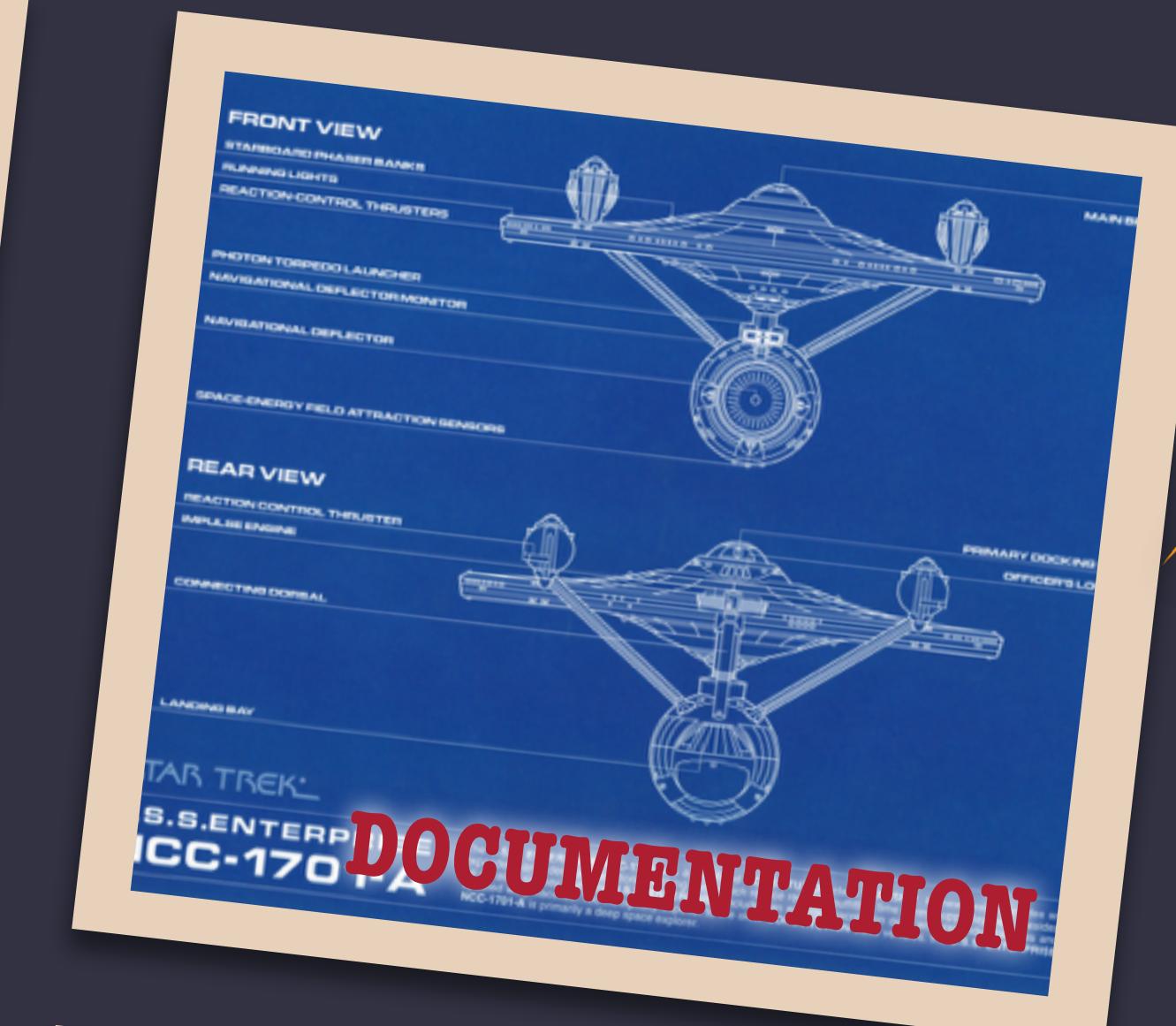
memegenerator.net

COE Center of
Excellence

Quicken Loans®
Engineered to Amaze™

WE SELECTED FOUR INITIAL NON-FUNCTIONAL REQUIREMENTS

The Architecture teams collaborated and decided that there are four primary areas to focus on as a starting point...



SECURITY

- ★ Data Handling
 - ★ PIFI Encryption
 - ★ Transport over any network must be encrypted using QKS
 - ★ Must be encrypted with QKS prior to persistence
 - ★ Authentication & Authorization
 - ★ Authentication using the Auth Service
 - ★ Authorization is currently managed by apps after authentication



PERFORMANCE

- ★ Performance Metrics
 - ★ Transactions/Second
 - ★ Response Time
 - ★ What are your SLA's to the business?
 - ★ ...other metrics
- ★ Teams are asked to perform:
 - ★ Baseline Load Testing - get a set of starting benchmarks
 - ★ Regular Load Testing - see if their system is getting better or worse

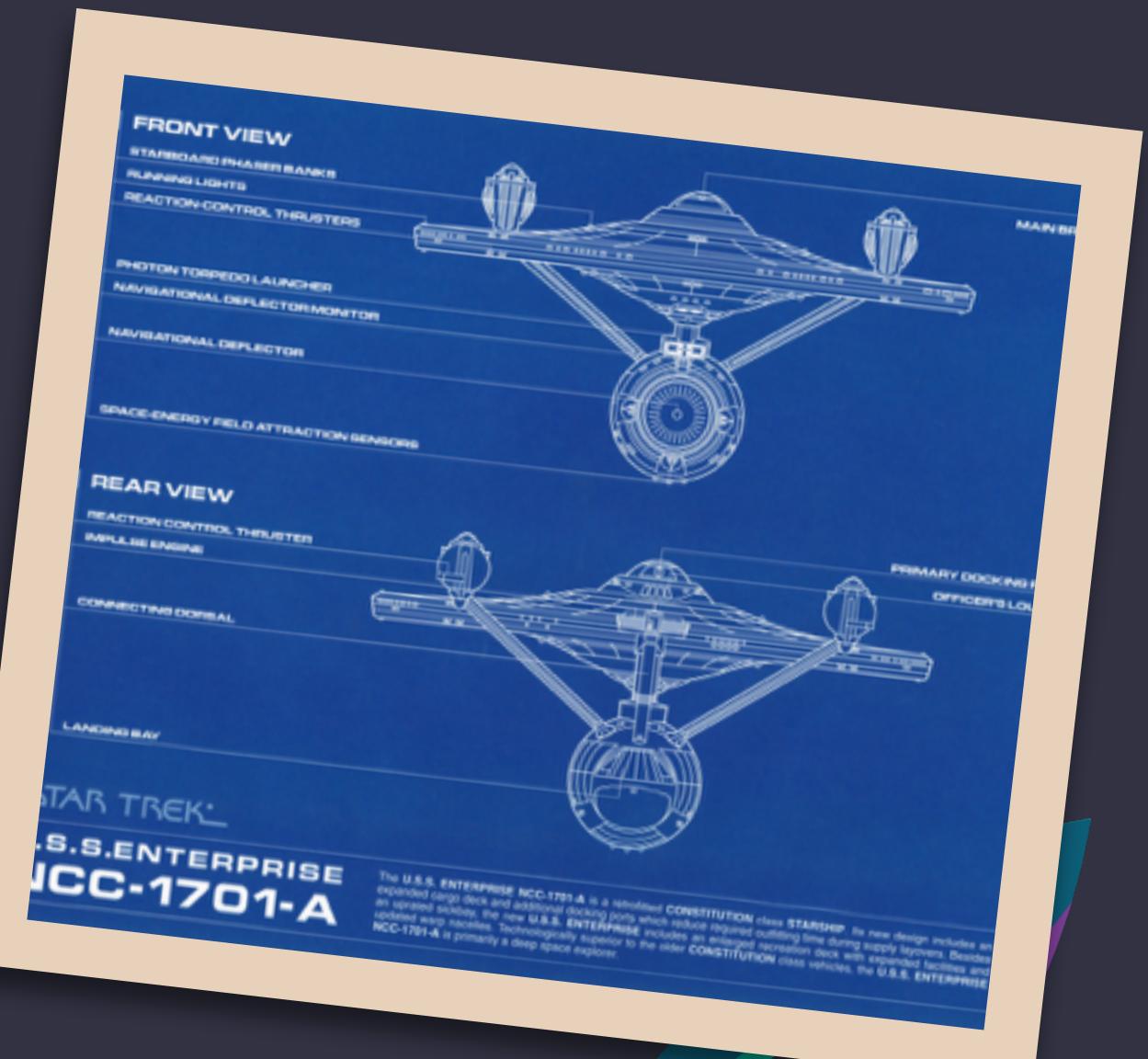


Excellence

Quicken Loans
Engineered to Amaze™

DOCUMENTATION

- ★ Architecture
 - ★ Inner workings of application
 - ★ External interactions and dependencies
- ★ Data
 - ★ Data Flow Diagram
 - ★ Data Dictionary
- ★ Each time a change is introduced, teams need to update their documentation



CODE COVERAGE

- ★ Identify your code coverage levels!
- ★ Code coverage NOT available?
 - ★ Determine which frameworks are available
 - ★ Write at least one test in the framework
- ★ Code coverage IS available?
 - ★ Define an acceptable threshold for each release
 - ★ Make sure your coverage at least remains constant
- ★ Define a plan to for **continuous, incremental improvement!**
- ★ Follow the plan!

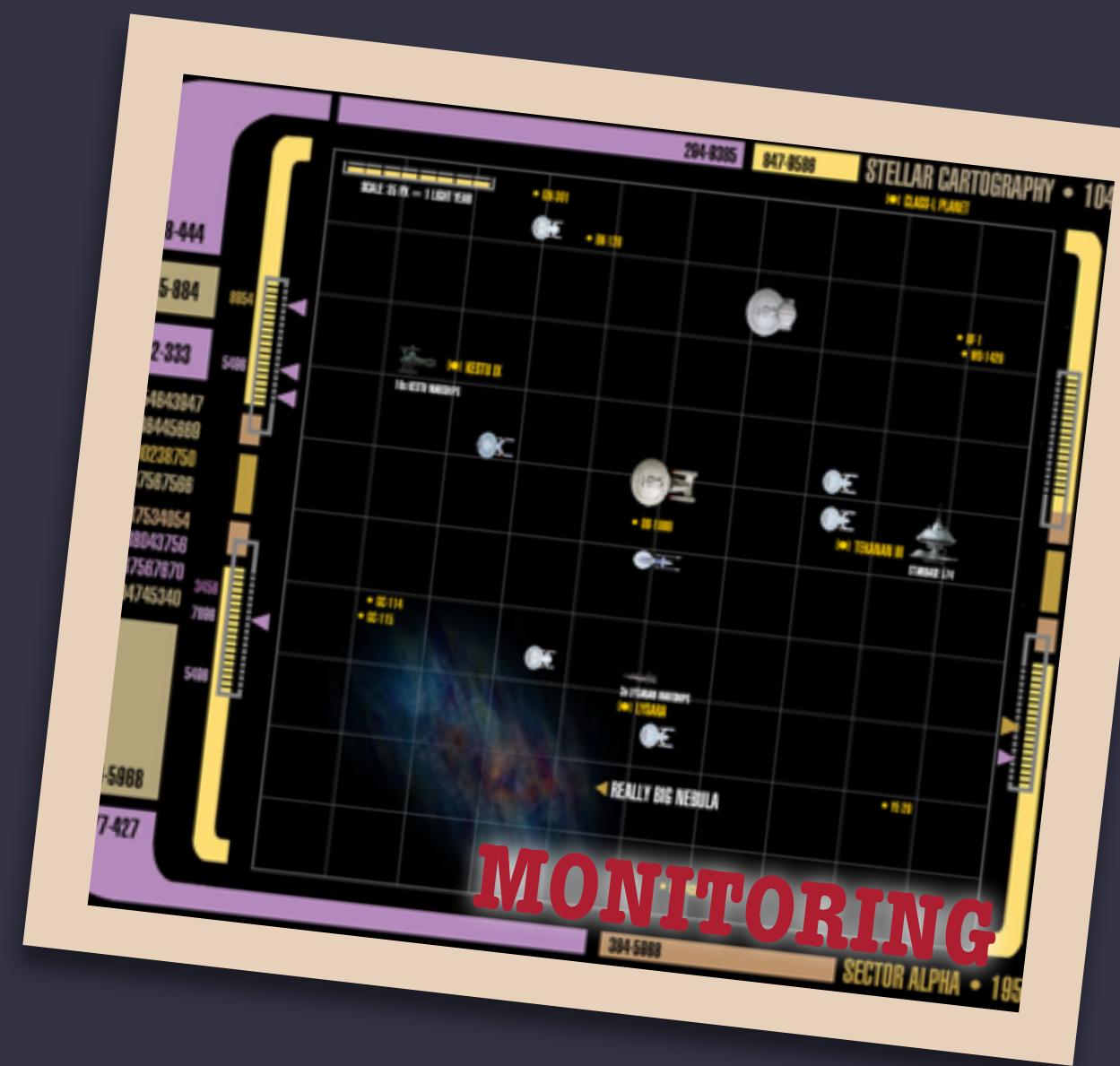


Make it so!



AS ENGINEERS BECOME ACCUSTOMED TO NFR'S...

We would like to introduce three more high-level concepts to each system and project.





LOGGING

- ★ Accountability
- ★ Manageability
- ★ Recoverability
- ★ Supportability
- ★ Verifiability

MONITORING

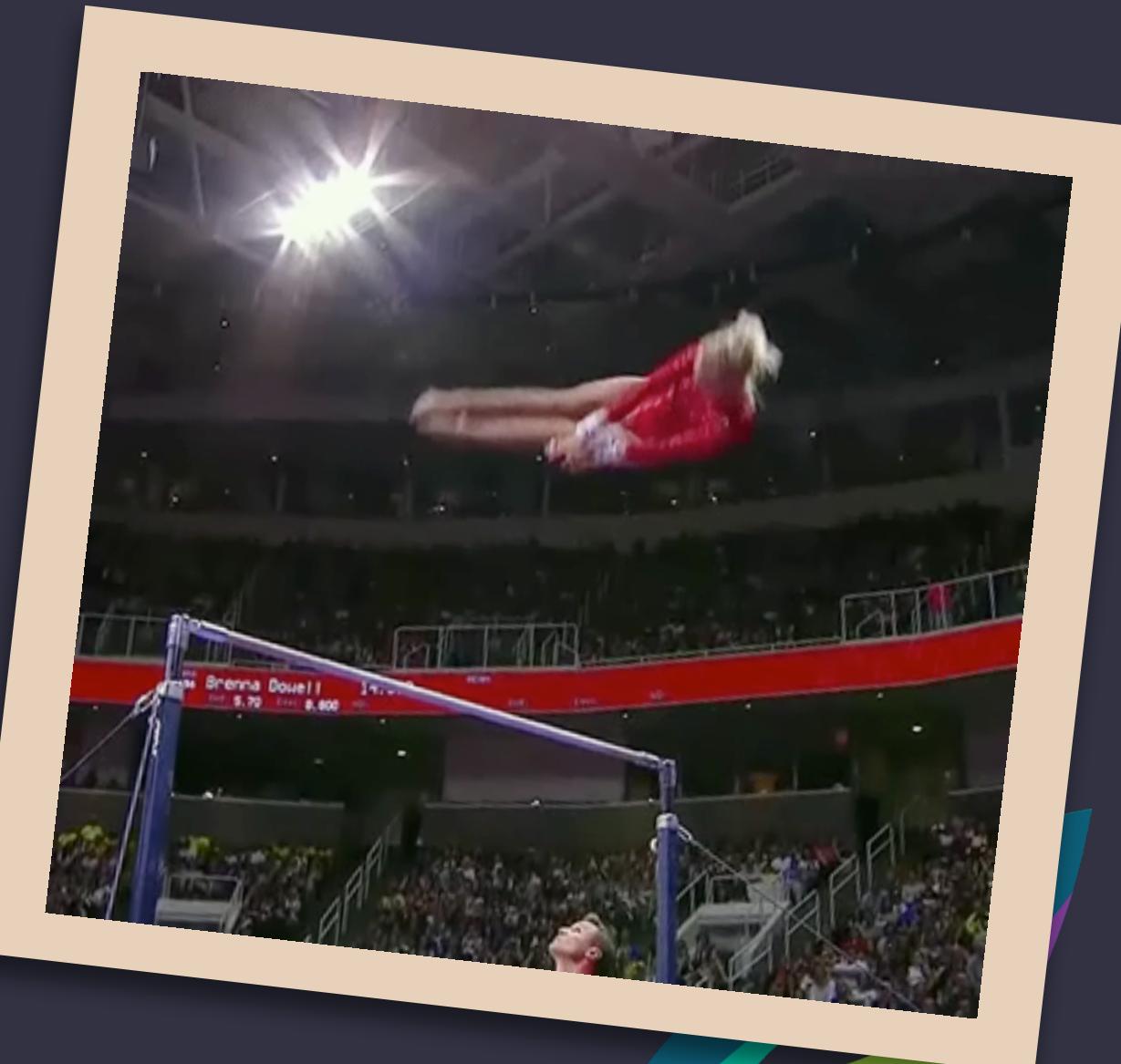
- ★ Availability
- ★ Capability
- ★ Durability
- ★ Manageability
- ★ Performability
- ★ Reliability
- ★ Scalability
- ★ Supportability
- ★ Survivability
- ★ Testability
- ★ Verifiability

ALERTING

- ★ Understandability
- ★ Maintainability
- ★ Trainability
- ★ Demonstrability
- ★ Reliability
- ★ Stability
- ★ Supportability
- ★ Verifiability
- ★ Vulnerability

AGILE & NFR'S

- ★ Should NFR's be prioritized separately from functional requirements?
- ★ Should NFR's be in a separate backlog?
- ★ If there is a separate backlog, how are NFR's prioritized against functional requirements?
- ★ How should NFR's and FR's be sequenced?



WHEN DO WE START?

- ★ **NOW!**
- ★ No, really. **NOW!**
- ★ All leaders are on board, top down!
- ★ It's the right thing to do!
- ★ **Did we mention... NOW?!**



HOW DO WE MEASURE SUCCESS?

- ★ Starts with project managers
- ★ Spreads to the team
- ★ Existing and new projects!
- ★ NFR's become integrated into every aspect of our culture
- ★ **Think globally - act locally**
- ★ **It's not about who's right, it's about what's right**
- ★ **Do the right thing**





shorty/NFR



-ilities

**WANT TO KNOW MORE?
shorty/NFR**



THANKS!

