# Chrome: Feature Proposal

CISC/CMPE 322

Mackenzie Furlong
Alex Golinescu
David Haddad
William Melanson-O'Neill
Michael Reinhart
Lianne Zelsman

# Agenda

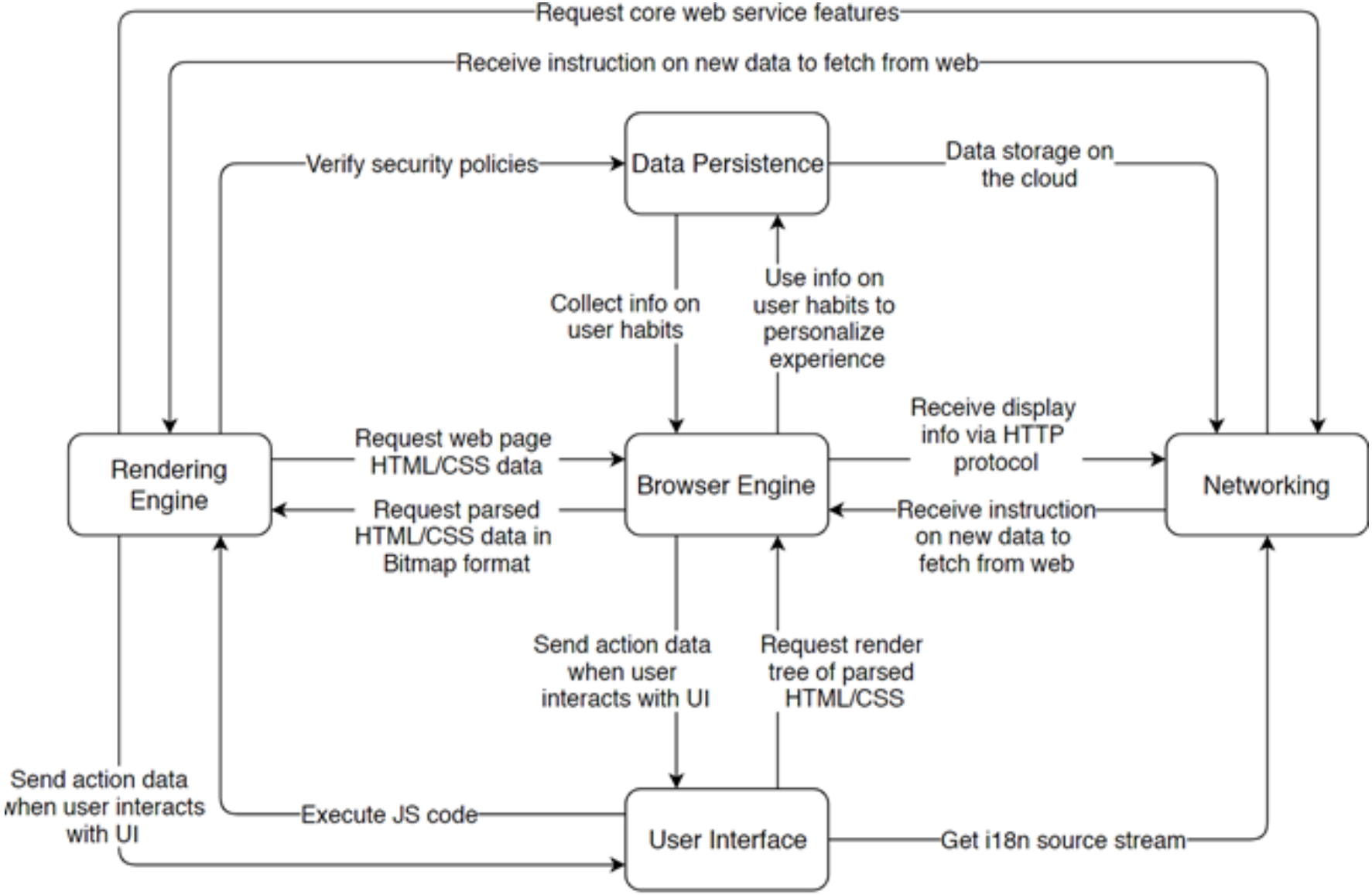
- Proposal Overview
- SAAM Analysis
  - Design Options
  - NFRs and Stakeholders
  - Advantages/Disadvantages
- Architecture Implementation
  - Sequence Diagram
  - Impact & Risks
  - Concurrency
  - Testing
- Team Issues
- Limitations & Lessons Learned
- Conclusion

# Proposal

# Facial Recognition

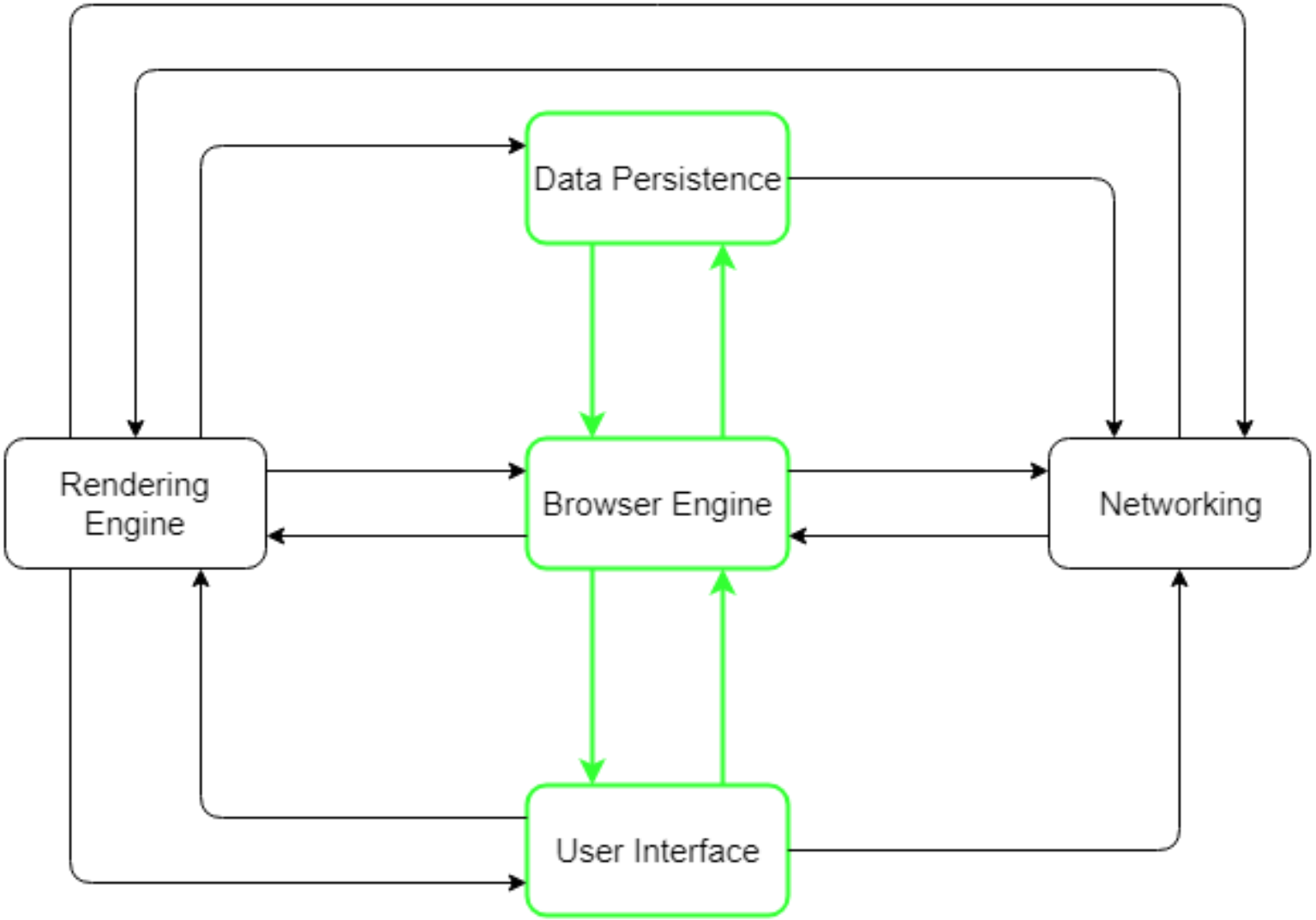
- Two use cases
  1. Protect autofill data
  2. Bypass Chrome-prompted login requests
- Value
  - Improve security
  - Increase speed

# Concrete Architecture



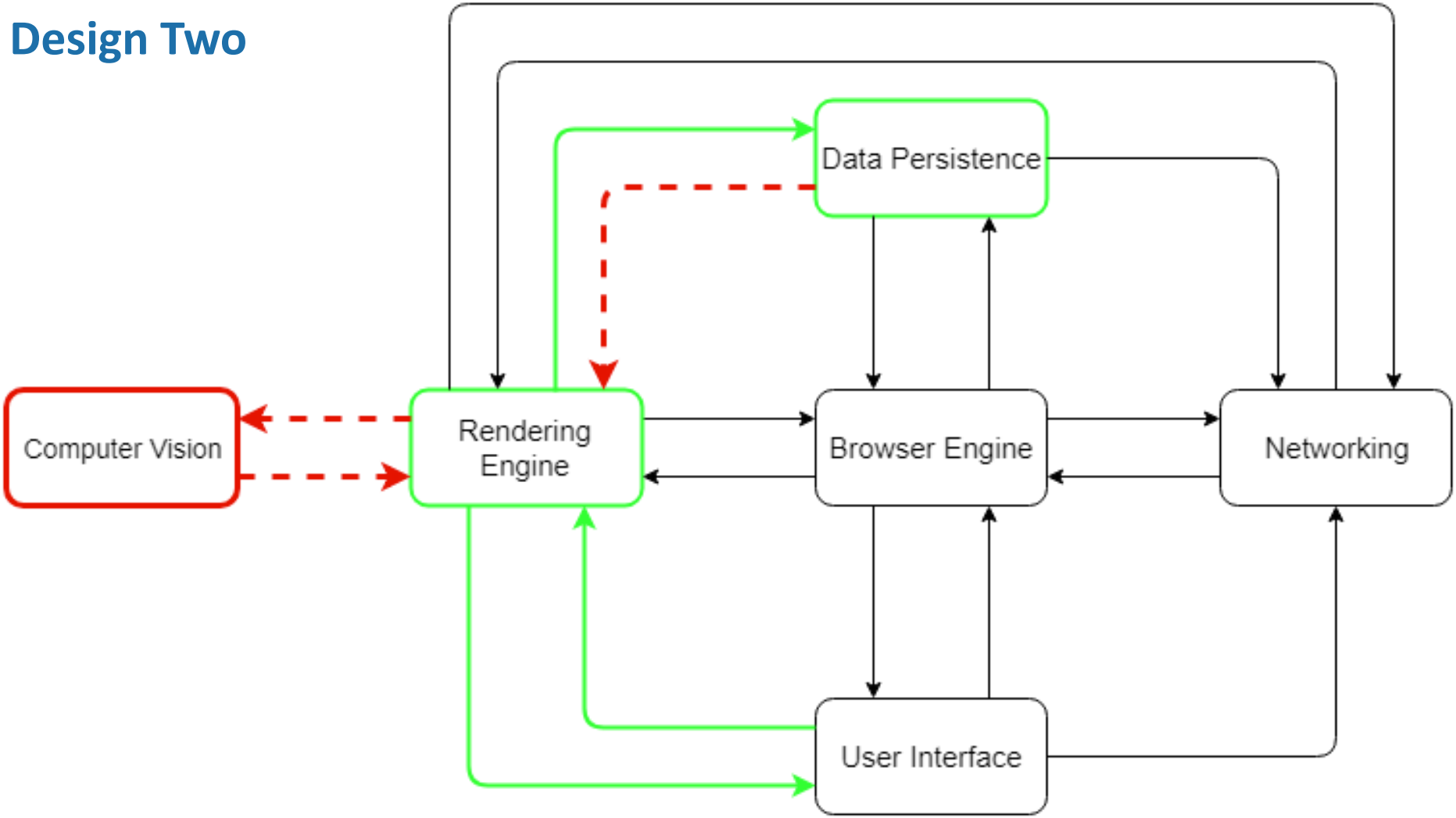
# SAAM Analysis

# Design One




Dependency Used by Feature → Subsystem Used by Feature

# Design Two




Dependency Used by Feature

Subsystem Used by Feature

New Dependency

New Subsystem

# Stakeholders & NFRs

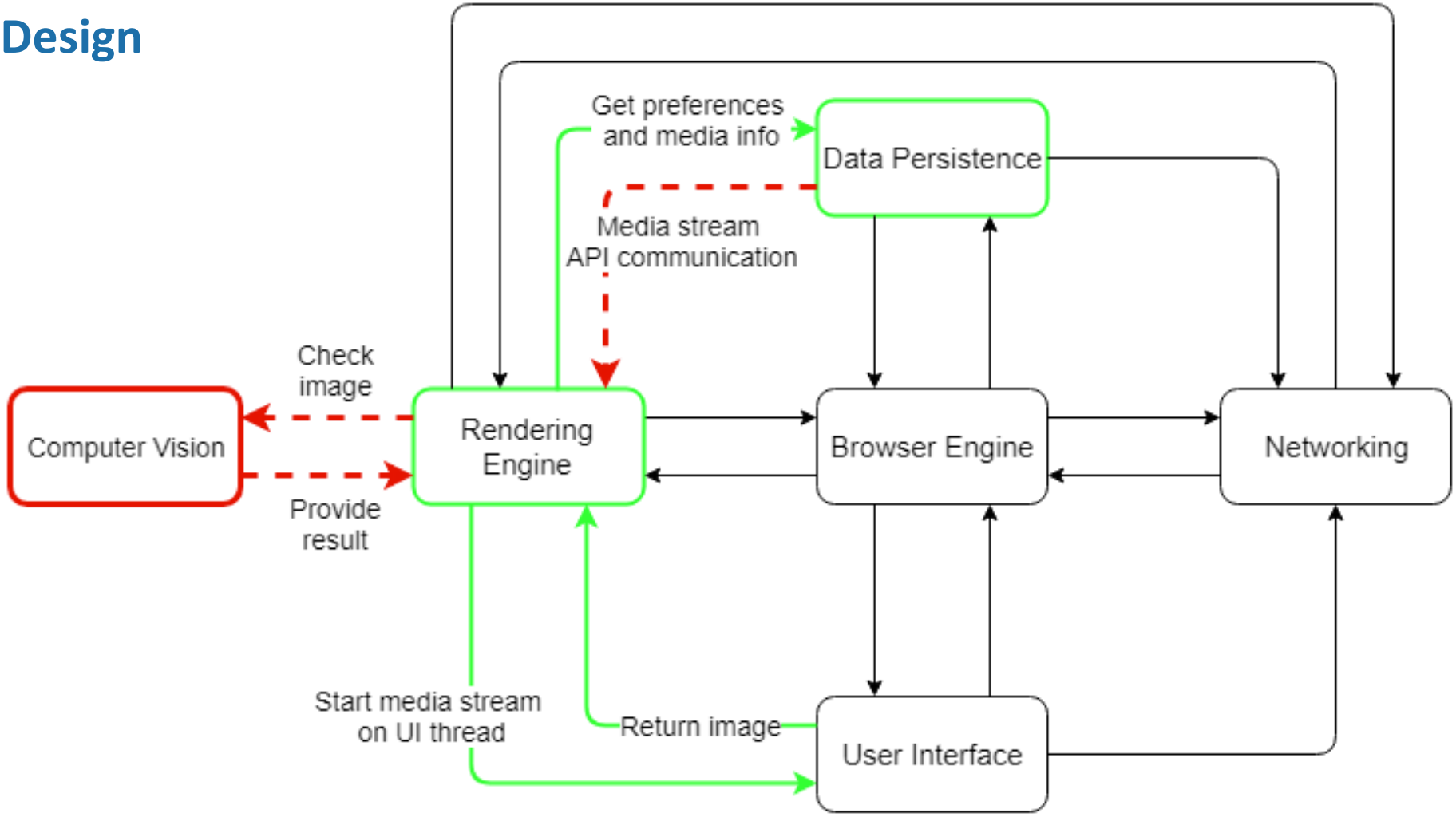
- Users: **performance** & **security**
- Google: **security**
- Developers: **evolvability, maintainability** & **testability**

# Advantages & Disadvantages

| | Design One | Design Two |
|---|---|---|
| Security | | • Preexisting security built into API<br>• Concurrency of using rendering engine |
| Maintainability | • Redundancy due to recreation of similar features clutters code | |
| Testability | • Decreased cohesion of subcomponents makes individual testing difficult | • High cohesion helps with testing<br>• Unit tests already exist for autofill and webcam API |
| Evolvability | • Decreased cohesion may reduce evolvability as changes in subsystems will be difficult to make | • Increased coupling can make evolution difficult<br>• But, spreading out functionality across more subsystems can ease isolated development |
| Performance | • Less subsystems and required connections, so less chance of bottlenecking | • API usage increases efficiency<br>• However, more subsystems and required connections can increase chance of bottlenecking |
| Other | • More work to develop | • Less work to develop |

# Implementation

# Design



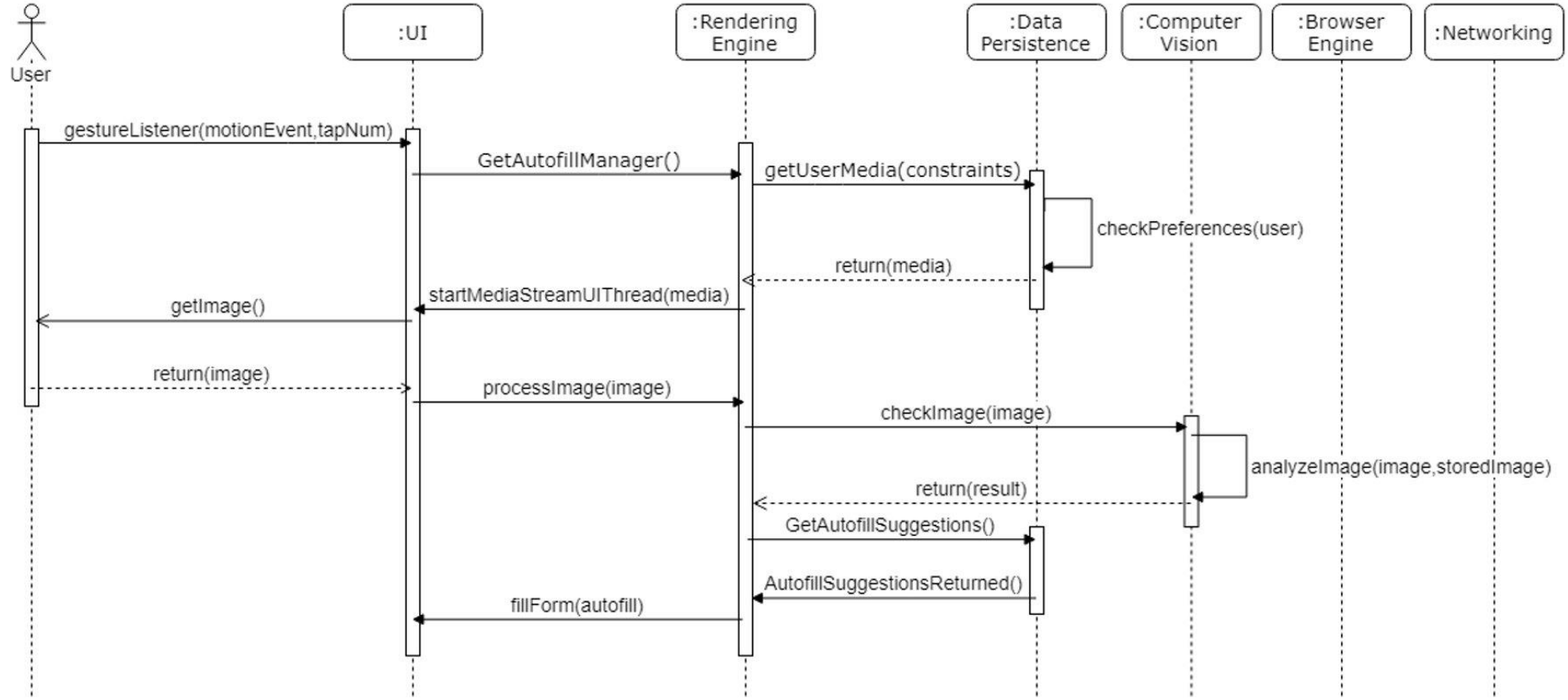

Dependency Used by Feature

Subsystem Used by Feature

New Dependency

New Subsystem

# Autofill Use-Case




Message → Response - - -> Lifeline | Object ▭ Actor  Activation ▮

# High-Level Impact

- Still object-oriented architecture
- No noticeable impact on performance, testability, maintainability
- Could decrease evolvability due to increased coupling

# Low-Level Impact

- No impact on most internal components
  - Mostly OO internally, with implicit style for UI and layering in rendering/browser system
- Data persistence and rendering engine subsystems need minor adjustment to allow new communication channels
  - Can be done easily using WebRTC, so won't impact component performance
- Following files in rendering engine must be adjusted:
  - *content/browser/renderer_host/media*
  - *content/renderer/media/*
  - *content/browser/webrtc/*
  - *content/public/renderer/*

# Concurrency

- Each Chrome tab runs its own rendering engine instance
  - Limits performance bottlenecks
  - Ensures errors in one tab won't impact login/autofill for another
  - Increases security (facial recognition must be re-done in each webpage)

# Risks

- Performance
  - Inefficient facial recognition algorithm could greatly reduce usage speed
- Security
  - Inaccurate facial recognition (false positives) is a huge security risk

# Testing

1. Facial recognition (performance)
2. Facial recognition (accuracy)
3. Correct UI & rendering engine reaction to changes in data persistence object
4. Correct response to system failures & errors during recognition process

# Design Process

# Team Issues

- Implementation involves multiple subsystems
  - Require cross-collaboration between multiple teams
- Suggestion: main development done by rendering engine team
  - Consultation with a few UI and data persistence team members to ensure correct feature functionality in those systems

# Limitations

- Difficult to piece together existing autofill and webcam functionality from Chromium documentation
- Multiple assumptions were probably incorrect, skewing accuracy of design

# Lessons Learned

- Even seemingly small changes can have a large impact on the entire system in unexpected ways
  - Really reinforces importance of low coupling and high cohesion

# Conclusion

- Many factors to consider when implementing a new feature
- Less intensive approach taken towards feature implementation
- Risks to performance and security with improper implementation
  - But benefits from successful implementation outway risks

Questions?