Google Chrome: Proposed Feature

Thick Glitches

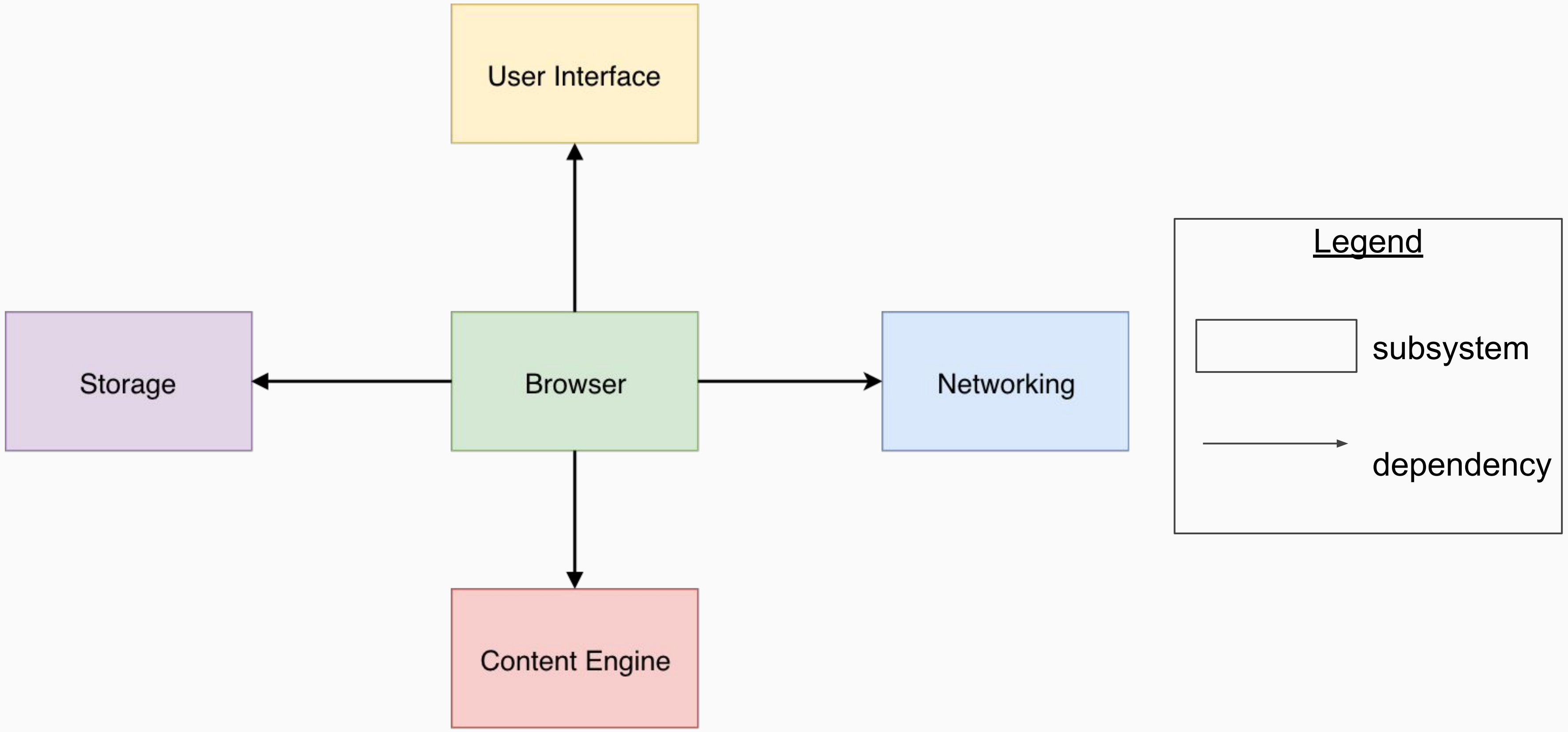
Tyler Mainguy, Liam Walsh, Andrea Perera-Ortega, Jessica Dassanayake, Alastair Lewis, Brendan Kolisnik

Introduction

<u>Proposed Feature: Safe Mode</u>

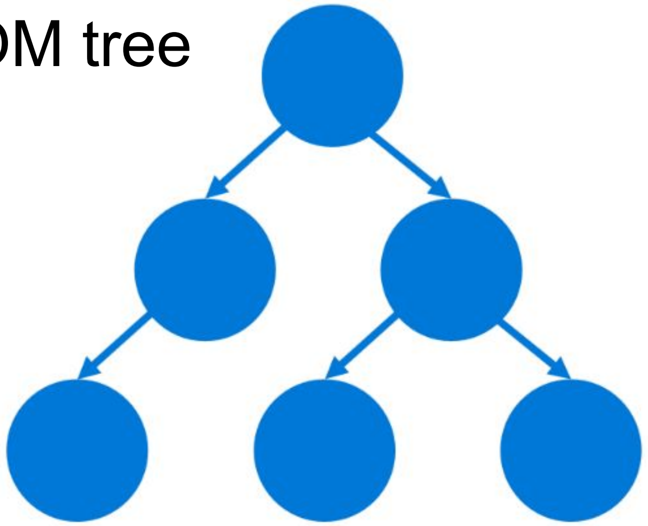
- Ideal for kids or people at work
- Allows user to censor instances of pre-set blacklisted words on a webpage deemed as inappropriate
- Blocks out media with inappropriate file names
- Can be activated/deactivated with a user entered password
- Utilizes all of the subsystems

Recall: Conceptual Architecture



Alternative Approach

1. Filter activated through Chrome settings
2. Check password against value in storage module
3. User loads the webpage
4. Content engine begins to build the DOM tree
5. Fetch blacklisted from content engine
6. Parses tag content as DOM tree built
7. Return filtered page back to user

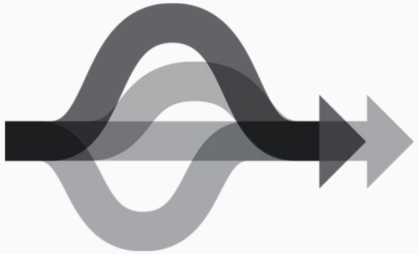


Proposed Approach

1. Activates filter using UI button beside the URL bar and enters the username and password

2. Checks username and password against storage module

3. Content engine builds the DOM tree for the site



4. Feature searches the tree for illicit content using the Boyer-Moore algorithm for the text of each HTML tag in the DOM tree.

5. Replace each instance of illicit content with censor.

6. Blacklisted words are stored in an array in the storage system

7. Applies the filtering to URL searches

Risks and Limitations of Proposed Approach

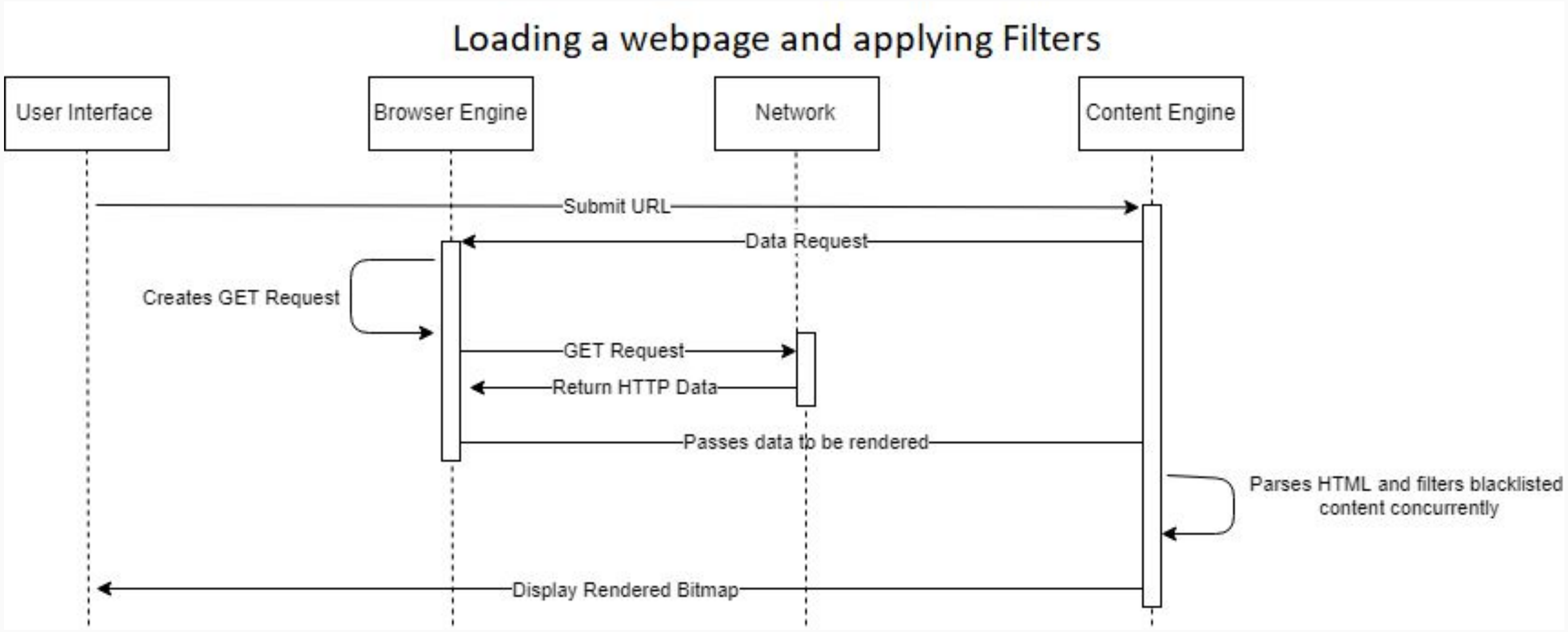
Risks

- Account management / password recovery system
- Man in the middle attacks for server communication (needs to be encrypted).
- Risk to children

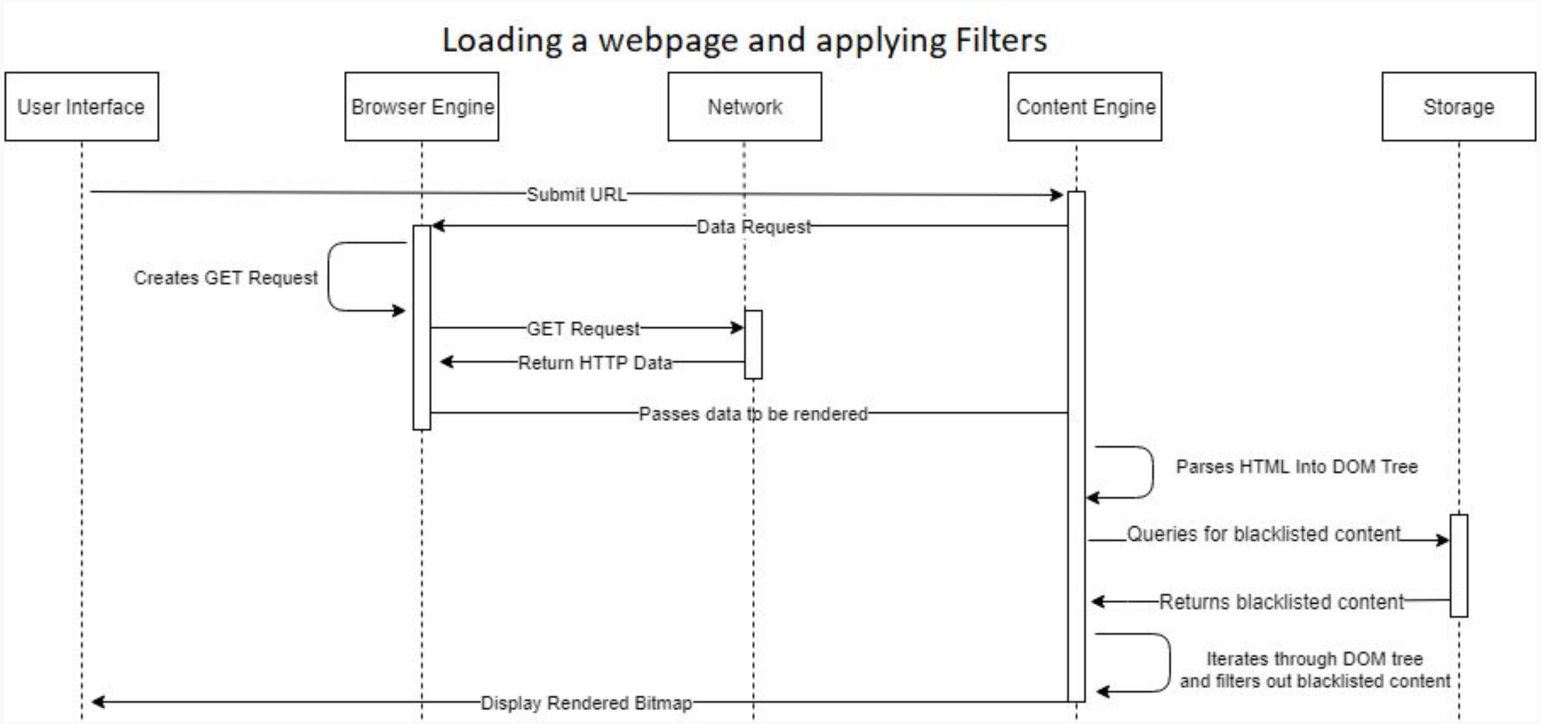
Limitations

- Requires an internet connection to recover password (need internet for browsing anyways)
- Requires an email address
- Not a perfect form of censorship some content may not get censored (media with no indication of their content in their filepath)

SAAM Analysis First Approach



SAAM Analysis Second Approach


Loading a webpage and applying Filters
User Interface
Browser Engine
Network
Content Engine
Storage
Submit URL
Data Request
Creates GET Request
GET Request
Return HTTP Data
Passes data to be rendered
Parses HTML Into DOM Tree
Queries for blacklisted content
Returns blacklisted content
Iterates through DOM tree and filters out blacklisted content
Display Rendered Bitmap

SAAM Analysis Comparison

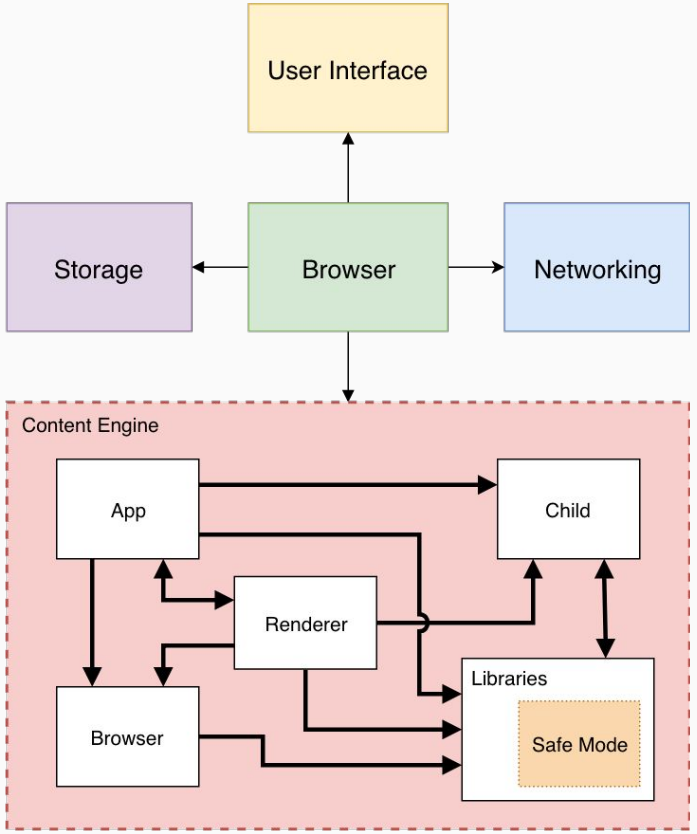
NFRs	Implementation 1:	Implementation 2:
Maintainability	● Only one area of code to be maintained in the system	● Several different sections of code to maintain that exist in seperate parts of the architecture
Performance	● Single pass of the DOM Tree improves run time efficiency of filtering out blacklisted content	● Poor performance as many subsystems are interacted with on the retrieval of the blacklist and then two passes over the dom tree takes more time then one
Reusability	● Poor reusability as the code is added into the actual rendering process and is tightly coupled to the parsing process	● The actual filtering implementation is seperate from the rest of implementation of the addition so that aspect can be easily reused

SAAM Analysis Decision: Second Approach

NFRs	Stakeholders	
	Google Development Team	**Users**
Maintainability	● Easier to maintain the actual filtering process as it is separate from other Chrome functionality	● A more maintainable architecture for the addition makes it less likely for the code to have bugs or errors and display blacklisted content as a result ● The blacklist is kept in storage which means that the user can easily add or remove desired content from the blacklist
Reusability	● Retrieving and filtering out blacklisted content are two distinct sections of code that can be extracted and reused in another environment if desired	● The process of retrieving blacklists from the storage is its own process or code so it could be reused to also blacklist pictures, ads or anything other than text.

Potential Impacted Architecture

Content Engine	➢ Traverses DOM tree after creation
Browser	➢ Logic behind enabling/disabling safe mode
Storage	➢ Fetching blacklisted words, ➢ Account info storage
User Interface	➢ Buttons for enabling/disabling safe mode ➢ Entering password
Networking	➢ Resetting password through email (Google SMTP)



Testing Interactions with Current System

Test Case	Input	Output	Interactions
Blacklisted word in the web page content	“Go to h*ll”	“Go to heck”	Safe mode will change how the content engine renders the finished product bitmap for any given web page containing prohibited words
Blacklisted word in filename of media	explicit-image.jpg	[redacted]	Safe mode will prevent the content engine from rendering any media with a prohibited word in its file name
Turn off filter temporarily with password	********	Filter temporarily turned off	Safe mode will interact with the browser engine to enable its filter via the chrome settings, using a password, which is stored in the Storage subsystem
Reset password with email	Click: Reset Password	Password reset link has been sent to your email	Safe mode will interact with the networking subsystem, and use Gmail’s SMTP server to send an email to reset your password

The Effects of Concurrency

- **What does it allow Chrome to do?**
  - **Sandboxing processes**
    - Async requests confirm failed processes don’t block browser I/O thread
    - Restrict processes network requests and system access by facilitating requests through single access point
  - **Execution speed increases**
    - Requests to access data made by processes independent of one another
- **Concurrency of our Implementation**
  - Each filter running concurrently
  - Fetching of blacklisted sites concurrent while DOM tree built
  - Otherwise concurrency preserved with implementation

Team Issues

- Not an independent process due to coupling

- Implementing a new feature will affect multiple subsystems and the development teams assigned to them

Limitations & Lessons Learned

Limitations

- **Last week of the semester**
  - Busy schedules
- **Different from A1 & A2**
  - New approach and different material to discuss

Lessons Learned

- **Practical and tangible experience**
  - Implementing our proposed feature
- **Easier to get into smooth workflow**
  - Developed solid working habits from A1 & A2
  - Know the strengths and weaknesses of each group member, delegate content accordingly

Conclusion

- The motivation behind the feature is to control the content that is accessible to certain users
- Utilized SAAM analysis to determine the best approach
- All five subsystems are impacted by the realization of Chrome Safe Mode
- The object-oriented architecture style allows for modularity in implementing the new feature

Questions?