Redecorate

Shivam Kedia

Browser Extensions for Improved User Experiences

December 15, 2024

Contents

1. Background

1.1. Motivation

The digital landscape is rapidly evolving, with increased dependence on websites for utilizing various essential services. Many websites still suffer from poor design choices which hinder access to essential functionality often resulting in poor user experiences. As a growing number of non-technical users transition to online platforms for everyday tasks, the need for well-designed websites has become all the more critical.

Users lack control over the content they are shown on websites. The essential nature of such platforms allows scarcely any alternatives. Consequently, inefficient designs can lead to frustration in a significant proportion of people subjected to requiring their use.

Furthermore, some websites boast certain accessibility features to showcase inclusive design choices catering to diverse populations. However, it is not unusual to witness unsatisfactory implementations of said enhancements, thereby distracting users from the primary use cases of such online platforms.

We recognize the immense value in addressing the differences between the sophisticated experiences offered by websites designed for elective purposes and the deplorable experiences found on essential websites.

Much of the work done within this project's scope discusses possible interventions in that part of the experience *within user control*, i.e., the client side within a browser context. We thus propose a browser extension that enables creating and sharing modifications to webpages in an intuitive manner, requiring minimal involvement on the everyday user's part.

1.2. Examples

Widely used Indian government websites like IRCTC[1], Parivahan Sewa[2], UIDAI[3] and the Income Tax Portal[4] are prototypical examples of websites with these problems. An internal survey conducted by Nishtha Das[5] sought to substantiate these general observations of poor experiences when using some of these government websites. While details of the survey contents are omitted here, participants complained of *slow load times, nonintuitive design* and *navigation difficulties* when using IRCTC and UIDAI websites. Suggestions for improvements included *reducing clutter*, improvement in *language* and *nomenclature* and *including tooltips*.

Screenshots in Figure 1 and Figure 2 illustrate the simplest design flaws through coloured borders with the following indications:

- Red: Advertisements and related distractions
- Orange: Unrelated to tertiary features; distractions from the primary functionality
- Yellow: Visually ambiguous information sections; unclear intentions
- Green: Space where primary functionality is housed

[1] Indian Railways' official website; primarily used for railway e-ticket booking: https://irctc.co.in/
[2] Ministry of Road Transport's online portal for availing license related services: https://parivahan.gov.in/
[3] Aadhar Portal (India's Biometric Identity System) https://uidai.gov.in/
[4] Official Website for e-filing of tax returns: https://incometax.gov.in/
[5] Ashoka University, Computer Science, ASP '24; her report was inaccessible, results from her project log

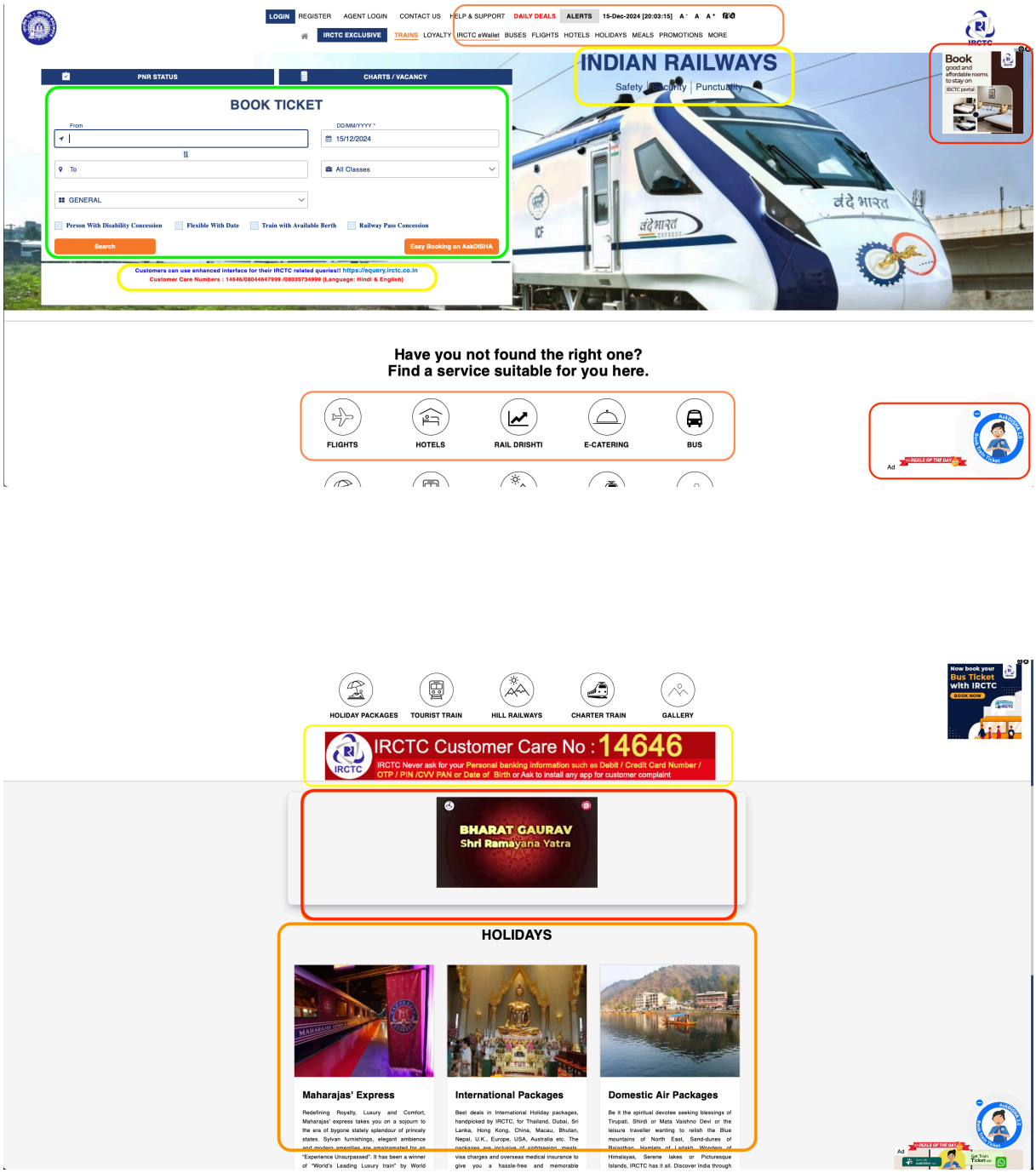


Figure 1: IRCTC (Indian Railways) Landing Page

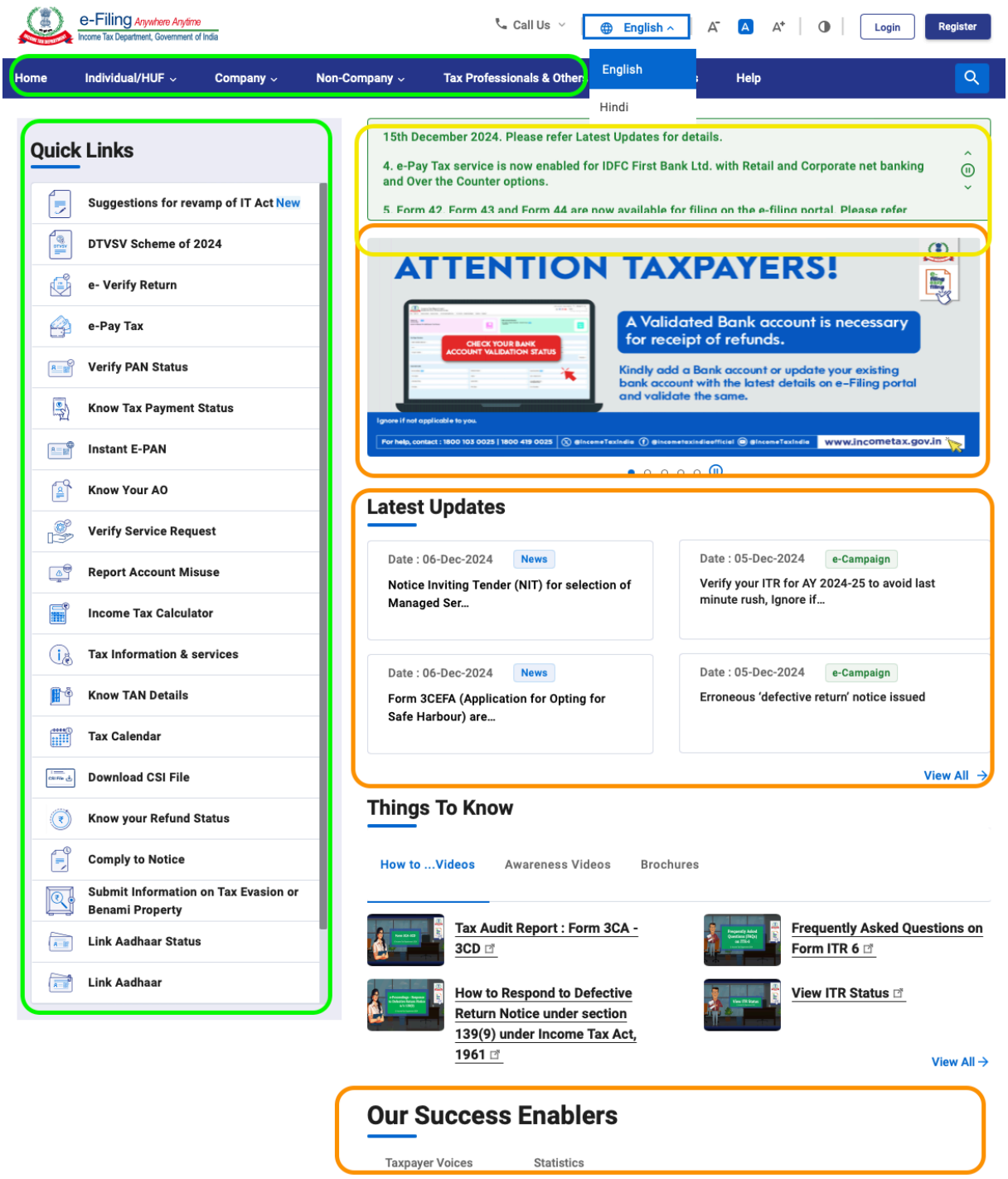


Figure 2: Income Tax Portal's Landing Page

1.3. Existing Work

It is important to acknowledge that no amount of supplementary enhancements can match the level of sophistication in user experience achieved through a well-designed website. Notwithstanding, below we outline few key ideas that address UI/UX problems in web browsing that have influenced this project's scope. There is an emphasis on creating options for the user to modify their view of the page in a variety of ways – no change whatsoever can be made to the original source.

1.3.1. Reading Mode

Also known as immersive reader and reader/reading view, this feature is available across popular browsers including Apple's Safari[6], Mozilla Firefox[7], Microsoft Edge and Google Chrome[8]. Essentially, reading mode provides a clutter-free experience by presenting a web article's content in a centered layout, typically allowing customisable fonts and colours for comfortable reading while hiding unrelated elements to eliminate distractions.

1.3.2. Browser Extensions for UI/UX Enhancements

Easy to install browser extensions provide a plethora of add-on functionality. For our purposes we were inspired by extensions that provide two broad categories of functionality: *style customisation* and *blocking*.

1. **Dark Reader**[9]: This popular free and open-source extension provides viewers a dark mode regardless of whether a particular website natively provides one. The extension can be easily toggled on and off. It allows customization options for brightness, contrast and colour schemes in addition to specific readability enhancements like custom fonts on a per-website basis.

2. **Stylus**[10]: Allows creation of user-styles per webpage through by *writing* one's own stylesheets. Other users of the extension can search for and apply publicly shared modifications through an application-store like interface. Commonly used for creating *skins* for platforms like YouTube and Discord.

3. **uBlock Origin**[11]: A content-blocker used to block ads and other irrelevant / distracting content on webpages. Greatly enhances user-experience by eliminating clutter. An additional benefit is the reduction of network requests to fetch unnecessary content.

4. **DFTube**[12]: Distraction-Free for YouTube is an extension used to hide various sections of the YouTube web-app including the feed, recommended videos, shorts, comments and notifications.

[6] https://support.apple.com/en-in/guide/iphone/iphdc30e3b86/ios
[7] https://support.mozilla.org/en-US/kb/firefox-reader-view-clutter-free-web-pages
[8] https://support.google.com/chrome/answer/14218344?hl=en
[9] https://darkreader.org
[10] GitHub: https://github.com/openstyles/stylus
[11] https://ublockorigin.com
[12] https://chromewebstore.google.com/detail/df-tube-distraction-free/mjdepdfccjgcndkmemponafgioodelna

1.3.3. Browser Included Enhancement Capabilities

1. **Arc Boosts**: Arc[13] is a new browser designed with an intent to make web-browsing less overwhelming and keep users "focused", "organized" and "in control". It ships with a significantly re-imagined browser layout alongside many other features that contribute to the impression of a cleaner interface compared to a typical modern browser.

   This project draws strong inspiration from Arc's unique feature called Boosts[14], which allows users to edit websites by:

   - Removing Elements (known as Zap)
   - Changing Background Colours
   - Modifying Fonts (by choosing from a limited selection)
   - Adding their own CSS and JavaScript to the page (known as Code)

   While some of these modifications like Code are manual changes, others like Zap provide an intuitive mouse-based way of usage. The GUI neatly supplements the Code feature with a web-inspector like interface to identify elements by hovering over them.

   Arc Boosts can be shared with other users if they don't contain custom JavaScript. Users obtain links to boosts they would like to share, which can then be distributed. Boosts for popular websites are available on Arc's Boost Gallery which provides an interface familiar to Stylus.

2. **Safari's Distraction Control**: Similar to Arc's Zap, the 2024 update to Apple's Safari adds a feature known as distraction control which allows users to hide elements that disrupt user experience. The interface is simple – users simply tap on the elements they would like not to see.

[13] https://thebrowser.company/

[14] https://resources.arc.net/hc/en-us/articles/19212718608151-Boosts-Customize-Any-Website

2. Project Overview

2.1. Scope

A review of the existing work has inspired us to create a browser extension to address some problems mentioned in Section 1.1. We consider the previous work done in this area to substantiate the value created for users. To illustrate this, we note the case of Dark Reader which claims 8 million active users and is now available in 20+ languages.

We propose a tool that allows users to **create** and **share modifications** to websites in the form of a chrome extension in Section 2.2. Our idea considers usability of enhancements, and a potential for reaching a wide audience. With regard to these considerations, we now detail the choices made and the rationale behind them.

2.1.1. Assumptions

Having acknowledged that there is no substitute for the levels of user satisfaction obtained from thoughtfully crafted websites, we make certain reasonable assumptions when defining the scope of our idea:

- **Site Compatibility**: We also note that we accept that implementations may not work well on every single website given the variety of factors determining user experience, including but not limited to browser versions, static / dynamic nature of sites and developer choices.
- **Low Overhead**: We would like to minimize external dependencies to keep the project low maintenance for both future developers and users. For example, we use JSON to contain modifications to websites, since it is a highly-interoperable low-dependency text-based format.
- **Functionality vs. Aesthetics**: Although extensions like Stylus and features like Arc Boost's Code allow aesthetic modifications to website, we instead prioritize functional improvements to the user experience. Omitting custom CSS-based modifications keeps the friction at a minimum for non-technical users.

2.1.2. Potential Reach

We implement the project as a Chrome Browser Extension due to its dominance in the browser market[15] as of 2024. We also note that our high-level design facilitates consistency across production-grade implementations on major browser platforms.

2.1.3. Ease of Use

Our design philosophy gives utmost importance to the tool's ease of use. The extension should not only be accessible to non-technical users but also allow them to obtain maximal functionality through minimal interaction with it. This core idea is reflected in the tool's simplistic interface.

2.1.4. Our Contribution

Lastly, we explain how our idea attempts to integrate the positive aspects of various work reviewed in Section 1.3.

[15] https://gs.statcounter.com/browser-market-share/desktop/worldwide

- Dark Reader offers focuses on enhancing **functionality** through an **uncomplicated** UI and consistent UX although for a niche use-case.
- Stylus inspires a **marketplace-like** UI to share customizations **per-website** with a minimum assumption of users' technical skills.
- uBlock Origin, DFTube and Distraction Control highlight the immense value in **hiding unwanted elements** for specific implementations.
- Arc features an **integrated way to create modifications** and share them. However, the browser is proprietary and Arc Boosts remain a closed-source feature.

Keeping these in mind our tool aspires to be open-source and available more generally across both – populations and websites.

2.2. High Level Design

Webpages are based on the client-server architecture. A browser extension adds supplementary functionality to the user's window (on the client side).

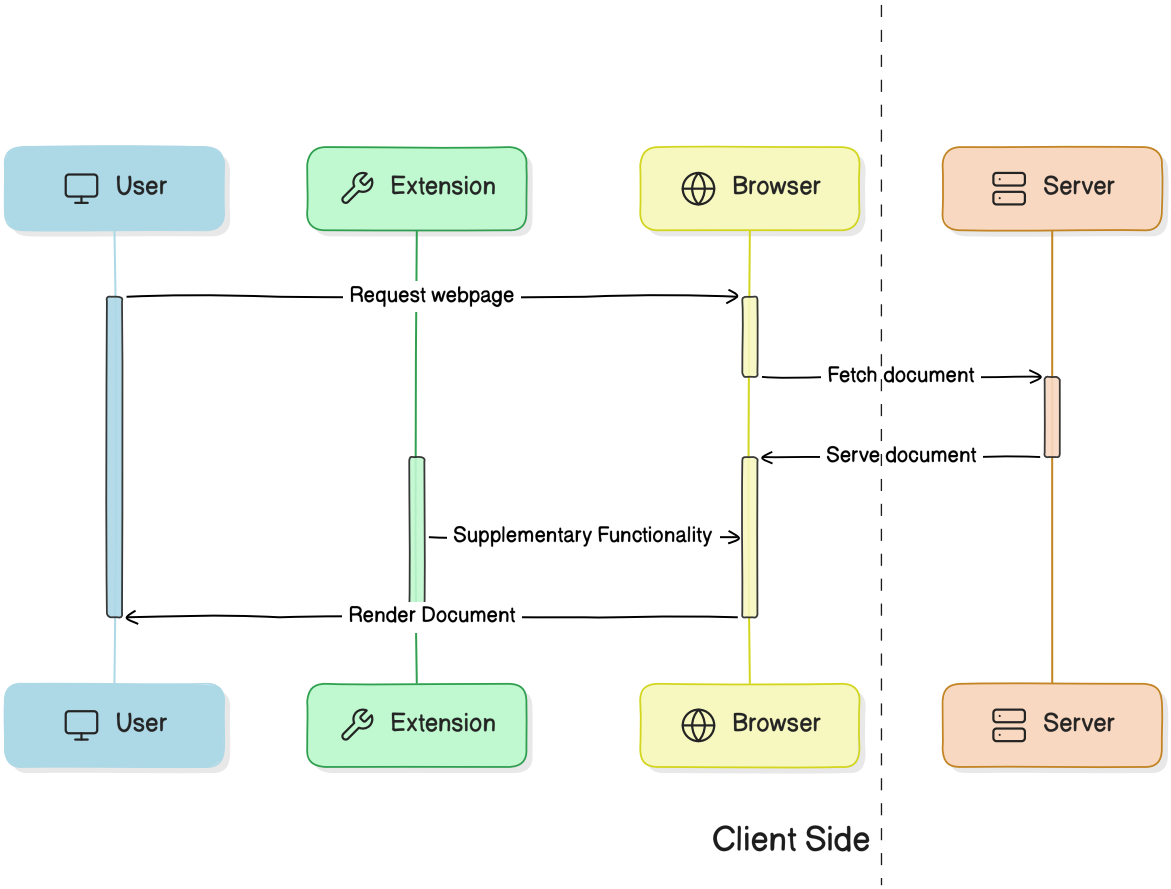


Figure 3: Extensions allow modifications to what users see on a webpage

2.2.1. Use Cases

Anyone with the extension can select edit the webpage as follows:

1. Select the modification / edit that they would like to apply.
2. Hover over and click elements to modify.
3. Input supplementary information (like text for creating tooltips), if any.
4. Save the modifications.

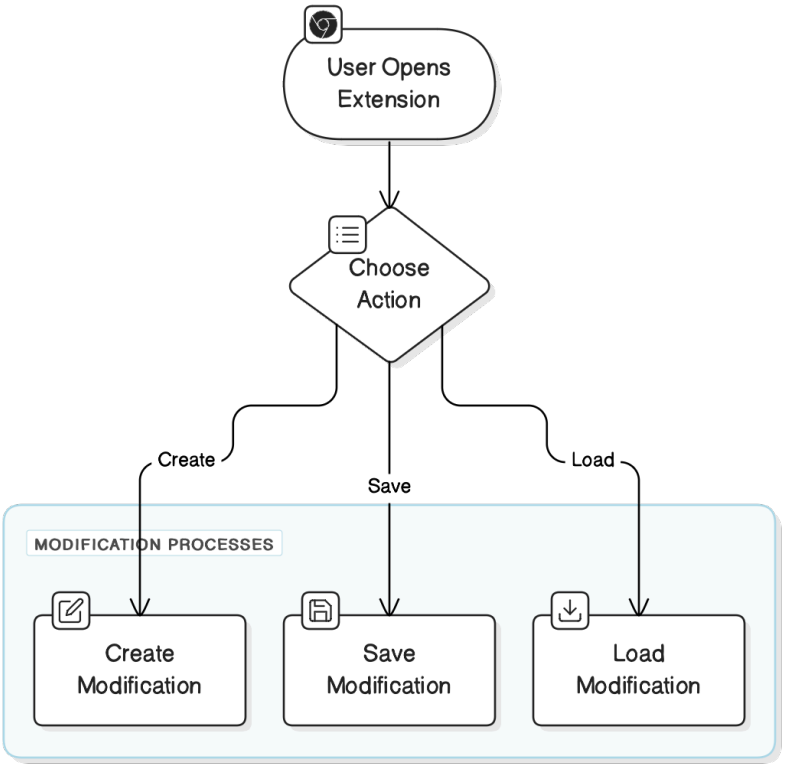
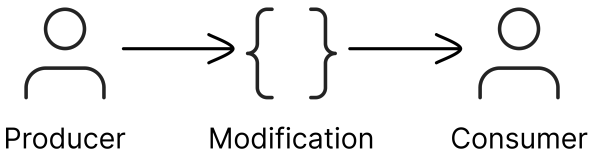


Figure 4: High-level overview of the tool

2.2.2. Structure of Modifications

Care has to be taken when designing the schema of modifications. We would like to ensure that once a modification has been created and saved, the result of applying the same modification is invariant across different machines (and browsers, if applicable).



We envision that a subset of users (**Producers**) create these modifications and the majority of the users (**Consumers**) simply apply others' modifications. Note that all producers are also consumers themselves. Thus, the invariance of results is key to a consistent user-experience.

A closely related feature of the above requirement is that modifications are *stateless*, thus becoming easily distributable. Regardless of when or how the modification is produced, the same instance of a modification object can now be used to produce consistent changes across computers.

2.2.3. Persistence

By virtue of their statelessness, each modification can now be stored either locally or on a hosted service, depending on the implementation requirements.

3. Proof of Concept

We illustrate the ideas presented in this document using a toy-implementation available on GitHub[16]. We mention important implementation details below.

3.0.1. Modifications

Similar to Arc Boosts' Zap, we provide a mouse-based interface to **hide** elements. These elements continue to take up space on the page, only their `opacity` is set to 0.

Due to time limitations, other potential modifications are discussed in Section 4.

3.0.2. Structure of Modifications

Modifications are saved as JSON files. The structure is as follows:

```
{
  url: example.org,
  nodeModifications: [
    {
      node: <HTMLElement-descriptor>,
      modifications: [
        {
          variant: <type of modification>,
          data: <associated data>
        },
        ...
      ]
    },
    ...
  ]
}
```

Each object has the URL of the page for which it contains the modifications and a list of edits. Each list item specifies the element which is being modified and the nature (`variant`) of modification.

Since the same element can have multiple modifications as long as it remains visible, we present the modifications to each node as a list. Care has been taken to ensure that undefined behaviour is minimised, hence hidden elements do not allow other modifications.

3.0.3. Robust Element Selectors

It was imperative to look for ways to find robust selectors for modified nodes in the DOM. For this purpose, a dependency-free library known as Finder[17] was used. It guarantees unique selectors for DOM elements. In our testing, changes persisted across reloads.

3.0.4. Persisting Modifications

Modifications can be saved in a JSON format (see Section 3.0.2). By default, the JSON object is saved to the extension's offline storage via the `chrome.storage` API. Modifications for every URL visited are loaded from storage and applied if they exist. They can also be manually loaded if needed.

[16] https://github.com/shivamkedia17/capstone°, also reachable by clicking the GitHub icon on the cover page
[17] https://github.com/antonmedv/finder°

To illustrate that the idea works across devices, a serverless cloud platform is used to host a database with key-value pairs. `GET` and `POST` methods allow `get(key)` and `set(key, value)` respectively. The `key` is simply a SHA-256 hash of a webpage's URL, the `value` is the saved JSON structure.

We successfully managed to save, retrieve and apply modifications using the Cloud Storage platform.

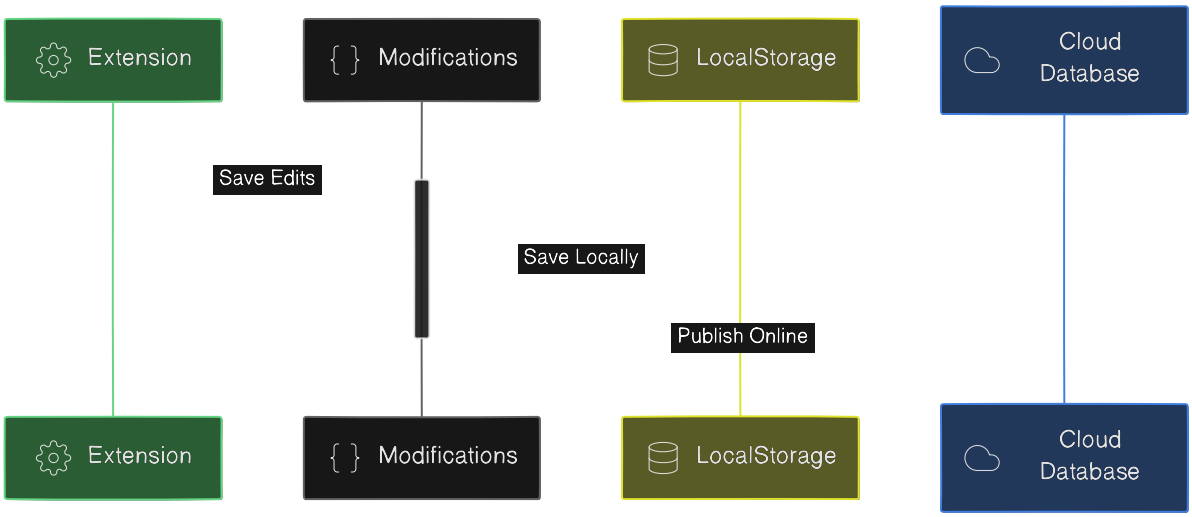


Figure 6: Saving Modifications

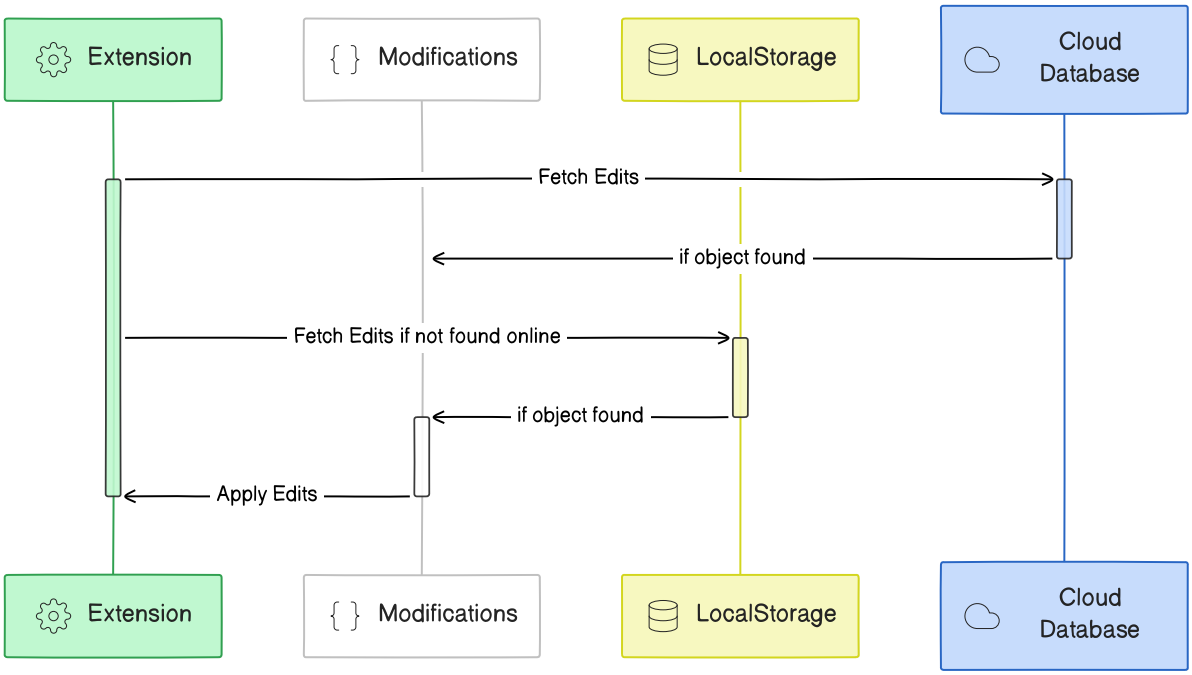


Figure 7: Loading Modifications

Sufficient error handling was put in place to prevent disruptions if modifications applied to dynamically generated elements are trying to be applied after a reload and these elements have not been found.

4. Future Work

The implementation serves as a proof-of-concept. We now conclude this by discussing potential improvements to the project's implementation. There is significant room for addition as discussed below.

4.1. Modifications

Our suite of modifications is presently limited to hiding elements. We considered allowing the following potential changes:

1. **Delete**: Elements may also be removed from the DOM instead of continuing to take up space.
2. **Annotations / Tooltips**: Helpful messages to allow users to better navigate pages with ambiguous layouts.
3. **Text Changes**: Rewriting parts of the page's text for enhanced communication clarity.

4.2. Persistence and Management

The Cloud Storage platform demonstrates immense potential for sharing extensions. Changes applied to a particular website are the same across multiple clients because they are fetched from the cloud server by default. However, our JSON structure can easily be used to offer a library of modifications per-website, optionally containing author information and temporal versioning.

Since we imagine much fewer users to be **producers** of modifications, such a library like interface could also be extended to allow multiple authors to jointly create modifications by merging their work.

4.3. User Feedback on Interface

The simplistic extension UI can be enhanced to provide some feedback to the user about changes in connection status and the current active state of the extension. The interface to hide elements can be make significantly more ergonomic.