Profiling

Studienarbeit

Studiengang Informatik
OST - Ostschweizer Fachhochschule

Herbstsemester 2021

Autoren: Bojan Dakic, Fabian Tiri

Betreuer: Prof. Dr. Nathalie Weiler

Danksagung

Wir möchten unserer Betreuungsperson Frau Prof. Dr. Weiler einen grossen Dank für die wertvollen Inputs und Hilfestellungen aussprechen.
Ausserdem möchten wir uns bei Dajana Dakic für das Korrekturlesen der Arbeit bedanken.
Zudem danken wir allen Personen, die uns bei unserer Studienarbeit in irgendeiner Art und Weise unterstützt haben.

Abstract

Seit die EU im Mai 2018 die Datenschutz-Grundverordnung (DSGVO) in Kraft gesetzt hat, ist die Anzahl der sogenannten Cookie-Banner im Web schlagartig angestiegen. Cookie-Banner sind Pop-ups, die auftauchen, wenn man eine Seite aufruft. Ihre Funktion ist es die Einwilligung des Benutzers einzuholen, um persönliche Daten über ihn verarbeiten zu dürfen. Allen Cookies zuzustimmen, ist dabei oft die Standardeinstellung. Die Banner ständig manuell auszufüllen ist mühselig und eine automatische Bearbeitung dieser ist wünschenswert.

Das Ziel dieser Arbeit ist es eine Lösung zu entwerfen, die zum einen die User Experience des Benutzers verbessert, indem es ihm die Arbeit des Ausfüllens der Banner abnimmt. Zum anderen soll die Lösung auch dazu dienen weniger Informationen an Webseiten zu übermitteln und somit dem Profiling entgegen zu wirken. Für diesen Entwurf soll dann ein Prototyp entwickelt werden.

Um dies zu bewerkstelligen, wurden zunächst Cookie-Banner von verschiedenen Webseiten analysiert. Anhand dieser Analyse wurden schliesslich Lösungsansätze erarbeitet. Daraufhin wurden die Vor- und Nachteile der Ansätze miteinander verglichen und schlussendlich einer ausgewählt, der umgesetzt wurde.

Das Ergebnis unserer Arbeit ist ein Plugin Prototyp, welcher aufzeigt, dass es grundsätzlich möglich ist, die Cookie-Banner automatisiert zu bearbeiten. Dabei kann der Benutzer festlegen, ob er nur die essentiellen Cookies akzeptieren will oder alle. Der Prototyp unterstützt sieben Consent Management Provider (CMPs) und einige Webseiten, die den Banner selbständig implementieren. Weiterhin kann das Plugin ohne viel Aufwand um zusätzliche CMPs erweitert werden, indem Regeln in einem JSON-File ergänzt werden. Benutzertests haben die Usability des Plugins bestätigt.

In unserer Arbeit haben wir die am weitverbreitetsten CMPs implementiert. CMPs, welche seltener vorkommen, können das Plugin vollenden. Allerdings ist es bereits jetzt in einem Stadium, in dem es der breiten Masse zugänglich gemacht werden kann und das Erlebnis der Benutzer verbessert.

Management Summary

Ausgangslage

Cookie-Banner sind Pop-ups, die auftauchen, wenn man eine Webseite aufruft. Ihre Funktion ist es, die Einwilligung des Benutzers einzuholen, um persönliche Daten über ihn verarbeiten zu dürfen. Diese Banner sind mittlerweile ein ständiger Begleiter unseres Lebens. In den letzten paar Jahren ist die Anzahl dieser stark angestiegen und wegen immer strikteren Datenschutzgesetzen werden es auch in Zukunft nicht weniger. Die meisten Benutzer empfinden diese als störend und verstehen nicht warum es sie überhaupt benötigt.

In dieser Studienarbeit werden wir eine Lösung vorstellen, die die User Experience wieder verbessern soll. Dies wird bewerkstelligt, indem wir einen Prototypen eines Plugins entwickeln, der automatisiert die Banner gemäss Benutzervorlieben bearbeitet.

Vorgehen

Bevor das Plugin verwirklicht wurde, musste zunächst geklärt werden was genau Cookies sind. Daraufhin wurden diverse Cookie-Banner analysiert. Auf Grundlage dieser Analyse wurden unterschiedliche Lösungsansätze evaluiert.

Mithilfe der Erkenntnisse unserer Vorstudie wurde dann ein Plugin entwickelt, welches die Banner automatisch ausfüllen kann.

Ergebnisse

Das Ergebnis dieser Arbeit ist ein Plugin, das in der Lage ist anhand Benutzervorlieben Cookie-Banner zu akzeptieren oder nur die Essentiellen auszuwählen und die restlichen abzulehnen. Dabei visualisiert es anhand des Icons, ob ein Banner auf der Webseite detektiert wurde und wie mit diesem verfahren wurde. Zusätzlich ist es in der Lage anhand eines Klicks auf einen Button die momentan aktiven Cookies anzuzeigen.

Das Plugin wurde für die Browser Google Chrome und Mozilla Firefox entwickelt und verwendet dabei für beide den selben Source Code.

Ausblick

Unsere Arbeit zeigt auf wie man Cookie-Banner automatisiert bearbeiten kann. Es werden bereits viele Banner vom Plugin erkannt und ausgefüllt, jedoch handelt es sich lediglich um einen Prototypen. Anhand eines JSON-Files können auf eine einfache Art und Weise neue Regeln für die Detektion hinzugefügt werden und damit mehr Banner abgedeckt. Um diesen Prozess zu vereinfachen, wäre es sinnvoll die nicht detektierten Webseiten an einem Server zu übermitteln, um eine Priorisierung der neuen Implementationen zu definieren. Die entsprechende Funktion, um

die Webseiten zu loggen gibt es bereits in unserer Lösung, welche weiterentwickelt werden kann.

Inhaltsverzeichnis

1 Einleitung

Wir schreiben den 25.11.2009 und die EU erlässt die Richtlinie 2009/136/EG. Diese Richtlinie besagt, dass für die Verwendung von Cookies die Einwilligung der Benutzer erforderlich ist.[1] Dies war die Geburtsstunde der Cookie-Banner. Da es sich aber lediglich um eine Richtlinie handelt, müssen die Mitgliedstaaten diese erst in nationales Gesetz umsetzen, bevor sie eine Wirkung entfalten kann. Dadurch dass die Umsetzung von Staat zu Staat sehr unterschiedlich ausfiel, waren die Banner nicht weit verbreitet, bis am 25.05.2018 die DSGVO in Kraft trat. Als Verordnung ist diese im Gegensatz zu einer Richtlinie sofort rechtskräftig und zwang somit Webseiten diese für EU-Bürger auch umzusetzen. Dadurch vermehrten sich die Cookie-Banner schlagartig.

Gemäss einer Umfrage von bitkom erachteten bereits vor der DSGVO 55% aller Benutzer die Banner als nervig.[2] Durch die in Kraft getretene DSGVO sind die Banner unterdessen omnipräsent und nicht mehr wegzudenken. Würde man die Umfrage jetzt nochmals durchführen, würden wahrscheinlich noch mehr Menschen angeben, dass sie die Banner als störend empfinden. Daher werden wir im folgenden Lösungsansätze präsentieren und einen Prototypen entwickeln und vorstellen, mit dessen Hilfe die User Experience verbessert und das Tracking von Benutzern reduziert werden soll. Die exakte Aufgabenstellung dieser Arbeit findet sich im Anhang.

Der Bericht beginnt mit der Vorstudie, in der wir Cookie-Banner analysieren und mögliche Ansätze für die Lösung des Problems vorstellen. Diese stellen wir dann gegenüber und wählen schliesslich einen aus, um ihn zu verwirklichen.

Im zweiten Teil gehen wir auf die konkrete Umsetzung unseres Prototypen ein und erläutern die Idee dahinter, sowie mögliche Risiken. Weiterhin gehen wir auf Herausforderungen ein, auf die wir während der Entwicklung stiessen, und analysieren unseren Testbericht. Zum Schluss gehen wir auf zwei Fehler ein, die im Plugin noch vorhanden sind, und schliessen den Abschnitt mit einem Fazit und Ausblick für die Zukunft ab.

2 Vorstudie

2.1 Lösungsansätze

In diesem Abschnitt werden die verschiedenen Lösungsansätze mit ihren Vor- und Nachteilen vorgestellt.

2.1.1 Lösungsansatz 1: Alle Cookies akzeptieren und im Nachhinein löschen

Für den ersten Lösungsansatz haben wir uns überlegt, dass wir zunächst alle Cookies akzeptieren. Damit stellen wir sicher, dass User nicht mit den lästigen Bannern interagieren müssen und die Seite wie vorgesehen funktioniert. Die Cookies sollen dann beim Schliessen der Seite gelöscht werden, um zu verhindern, dass diese beim erneuten Aufruf an den Server übermittelt werden.

Vorteile

- Theoretisch vollständige Anonymität möglich
- Jeden Banner kann man einfach akzeptieren.
- Der User muss nur das Plugin installieren und danach keine weiteren Einstellungen vornehmen.

Nachteile

- HTTP-only Cookies können nicht bearbeitet werden.
- Neben dem Namen des Cookies wird auch der Pfad benötigt zum bearbeiten.
- Jede Seite wird gleich behandelt, es gibt keine Ausnahmen.

2.1.2 Lösungsansatz 2 : Cookie nach Präferenzen des Benutzers wegklicken

Ein zweiter Ansatz wäre es ein Plugin zu entwickeln, in dem ein User seine persönlichen Vorlieben auswählen kann. Das Plugin soll dann entsprechend dieser Vorlieben die Banner automatisch ausfüllen.

Vorteile

- User können individuell anpassen, welche Informationen sie freigeben wollen.
- Funktionalität der Seite bleibt sicher vorhanden.
- Sollte der User für eine Seite andere Einstellungen wollen, kann er diese manuell im Privacy Center festlegen und sie werden nicht überschrieben.

Nachteile

- Cookie-Banner variieren von Seite zu Seite, was es schwierig machen könnte einzelne Cookies abzulehnen.
- Sind personenbezogene Daten essentielle Cookies, müssen diese akzeptiert werden.
- Der User weiss eventuell nicht, welche Optionen er auswählen soll.

2.2 Analyse der Lösungsansätze

In diesem Abschnitt werden die beiden Lösungsansätze genauer betrachtet und Probleme, die es bei diesen gibt, analysiert.

2.2.1 Lösungsansatz 1: Alle Cookies akzeptieren und im Nachhinein löschen

Im Gegensatz zu Lösungsansatz 2 müssen wir bei diesem direkt mit den Cookies interagieren. Hierfür ist es notwendig zu wissen, was genau Cookies sind und wie sie aufgebaut sind.

Cookies

Wenn man mit dem Browser eine Webseite aufruft, speichert in der Regel der Server Cookies auf die lokale Maschine. Diese sind eine Textdatei und enthalten Daten, anhand der Server den User identifizieren kann. Sobald man sich erneut mit der Webseite verbindet, schickt der Browser die Cookies an den Server und dieser kann daraufhin die Seite gemäss der übermittelten Informationen anpassen, indem man z.B. sich nicht mehr einloggen muss oder der Warenkorb vom letzten mal noch gespeichert ist. Allerdings haben Cookies nicht nur Vorteile. Ein Unternehmen, welches Werbung auf mehreren Seiten betreibt, kann bei jedem User der eine der Seiten aufruft ein Cookie speichern, auch wenn mit der Werbung nicht interagiert wird. Dadurch kann dann ein Profil von jemanden erstellt werden, indem man analysiert, welche Seiten der User besucht hat.[3]

Aufbau

Um den Aufbau eines Cookies verstehen zu können, haben wir uns diese zunächst in den DevTools von Google Chrome angesehen (Abb. 2.1).

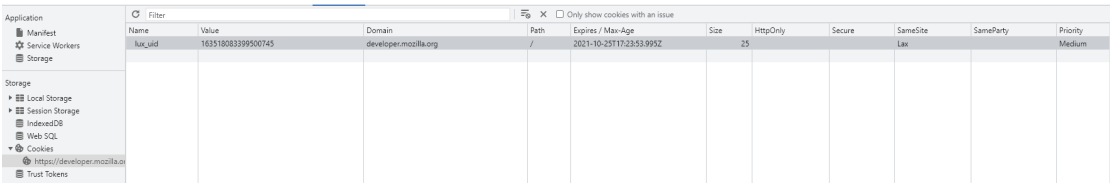


Abbildung 2.1: Cookies in Chromes DevTools

Für unsere Zwecke interessant sind die Attribute: Name, Path, Expires und HttpOnly.

Name

Anhand dieses Attributes kann man das Cookie identifizieren.

Path

Der Pfad bezieht sich nicht auf das lokale Filesystem sondern auf die URL. Das Cookie wird nur dann an den Server übermittelt, wenn der Pfad in der URL enthalten ist. Das bedeutet für den Pfad `/`, dass das Cookie immer mitgesendet wird, für den Pfad `/login` nur, wenn die URL mit `www.example.com/login` beginnt.

Expires

Mit diesem Attribut wird festgelegt wie lange das Cookie gültig sein soll. Sobald das festgelegte Datum erreicht ist, wird es automatisch gelöscht.

HttpOnly

HttpOnly ist ein Attribut, welches der Sicherheit dient. Ist dieses auf True gesetzt, verhindert es, dass man mittels JavaScript auf das Cookie zugreifen kann. Somit kann unter normalen Umständen nur der Server solche Cookies bearbeiten.[4]

Probleme

Um Cookies von einer Webseite auszulesen, kann man in der Konsole `document.cookie` eintippen und man erhält einen String, der alle Cookies enthält (Abb. 2.2). Das Format des Strings ist `cookiename=value`. Die einzelnen Cookies werden mit einem `;` voneinander getrennt.

Abbildung 2.2: document.cookie auf https://www.kicker.de

Die Cookies kann man bearbeiten, indem man eine Zuweisung an `document.cookie` macht, z.B. `document.cookie = 'test=test'` (Abb. 2.3).

Abbildung 2.3: Ein neues Cookie 'test' wird erstellt

Um nun ein Cookie zu löschen, muss man lediglich noch das Attribut `expires` definieren und auf ein Datum setzen, welches in der Vergangenheit liegt (Abb. 2.4).

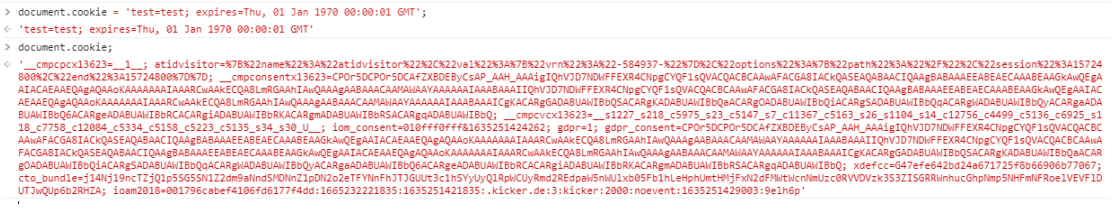


Abbildung 2.4: Das Cookie 'test' wird gelöscht

Bei allen Operationen mit `document.cookie` ist zu beachten, dass man den richtigen Pfad und die richtige Domain mit angeben muss. Will man ein Cookie bearbeiten und einer der Werte stimmt nicht überein, wird ein neues erstellt. Dementsprechend würde beim Versuch ein Cookie zu löschen, bei dem man den Pfad oder die Domain nicht kennt, einfach nichts passieren (Abb. 2.5).

Abbildung 2.5: Versuch das Cookie 'ioam2018' zu ändern und '__cmpcpcx13623' zu löschen ohne den richtigen Pfad bzw. Domain anzugeben

Diese Eigenschaft stellt uns vor ein Problem, da man mit simplen JavaScript nur den Namen und den Value eines Cookies auslesen kann. Allerdings kann man mit `window.location` auf den `hostname`(domain) und `pathname` von der aktuellen Seite zugreifen. Anhand dieser Informationen kann man dann den richtigen Pfad und die richtige Domain bruteforcen. Hierfür muss man lediglich alle möglichen Kombinationen ausprobieren. Dabei gilt als Trennzeichen bei dem hostname `.` und bei dem Pfad `/`. Für die Domain `www.test.ch` und den Pfad `/test` gebe es somit sechs verschiedene Möglichkeiten.

Ein weiteres Problem bleibt jedoch bestehen. Cookies, bei denen das Attribut HttpOnly auf True ist, lassen sich damit nicht löschen. Dieses Problem können wir mit einfachen JavaScript auch nicht lösen.

Für beide Probleme gibt es aber eine gemeinsame Lösung. Mit der `chrome.cookies` API bzw. `browser.cookies` für Firefox kann man auch HttpOnly Cookies bearbeiten und benötigt zudem nicht den Pfad und die Domain.

2.2.2 Lösungsansatz 2 : Cookie nach Präferenzen des Benutzers wegklicken

Bei diesem Ansatz geht es darum, die PopUps automatisch auszufüllen. Hierzu haben wir uns zunächst verschiedene PopUps angesehen. Dabei mussten wir feststellen, dass diese sich stark voneinander unterscheiden (Abb. 2.6, 2.7, 2.8).

Abbildung 2.6: Banner von https://www.google.ch

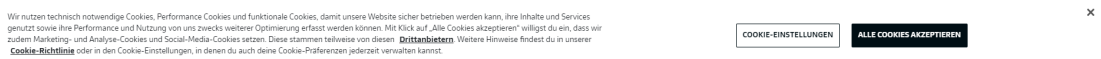


Abbildung 2.7: Banner von https://www.bundesliga.de

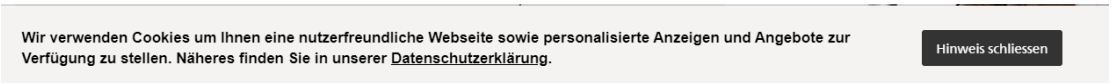


Abbildung 2.8: Banner von https://www.post.ch

Wie man an Abb. 2.8 sehen kann, gibt es sogar Banner, die man nur schliessen kann. In der Regel ist es möglich, daraufhin über Einstellungen auf der Seite die Vorlieben für die Cookies einzustellen. Aber für manche Seiten gibt es so eine Option nicht und man kann somit nur die Cookies akzeptieren. Dies stellt uns vor das Problem, wie wir mit solchen Bannern umgehen sollen. Um das Problem lösen zu können, müssten wir, nach dem Wegklicken des Banners, herausfinden, ob man Einstellungen vornehmen kann und falls ja, wo dies zu tun ist. In Anbetracht der Aufgabe, die PopUps effizient automatisch auszufüllen, erachten wir dieses Problem zwar als wichtig, jedoch würde es den Rahmen dieser Arbeit sprengen. Daher haben wir uns entschlossen, dass in diesem Fall lediglich der Banner weggeklickt wird.

Für die weitere Analyse haben wir uns das Document Object Model (DOM) genauer angeschaut (Abb. 2.9, 2.10, 2.11).

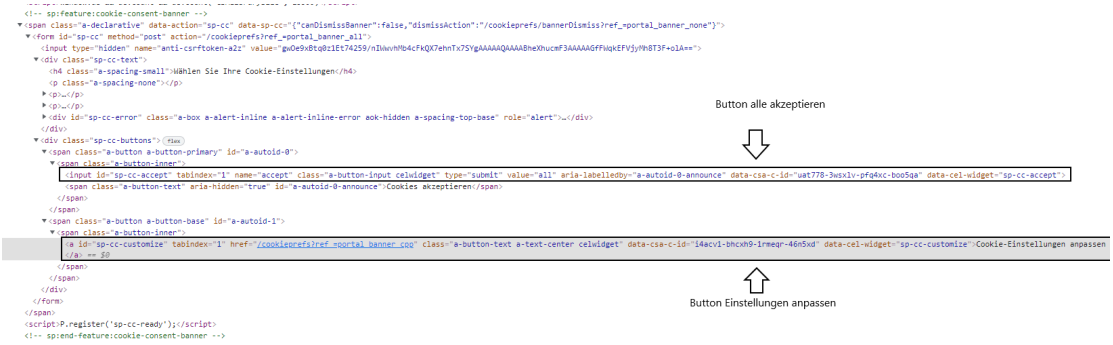


Abbildung 2.9: DOM von https://www.amazon.de

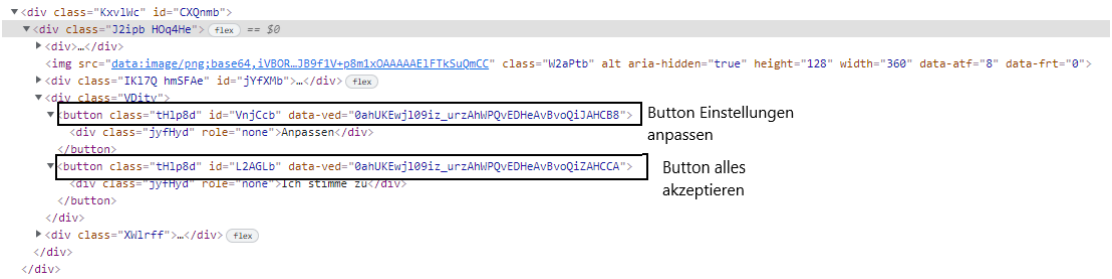


Abbildung 2.10: DOM von https://www.google.ch

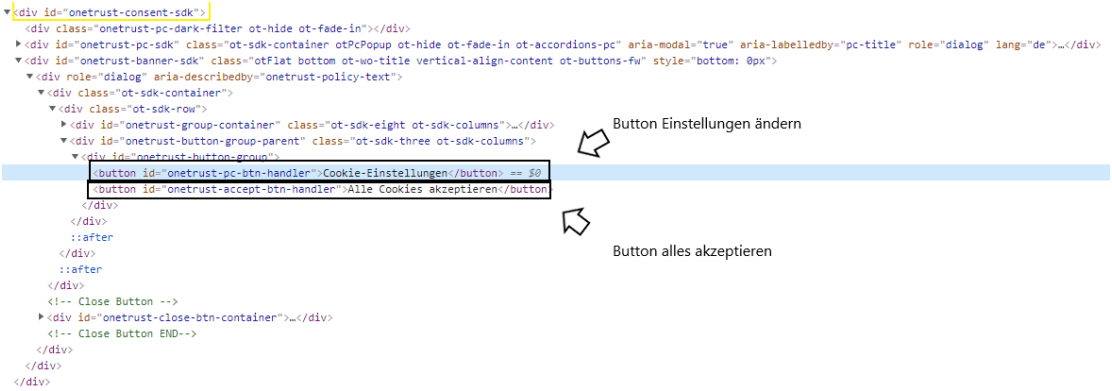


Abbildung 2.11: DOM von https://www.bundesliga.de

Auch hier mussten wir festellen, dass sich die Implementationen stark unterscheiden und keine allgemeine Struktur festzustellen ist. Allerdings haben wir in Abb. 2.11 die gelb markierte id entdeckt. Indem wir ein bisschen über OneTrust recherchierten, fanden wir heraus, dass OneTrust ein CMP ist und es noch weitere gibt. Vergleicht man noch andere Seiten, die OneTrust verwenden, mit Abb. 2.11 stellt man fest, dass sich das DOM sehr stark ähnelt (Abb. 2.12, 2.13).

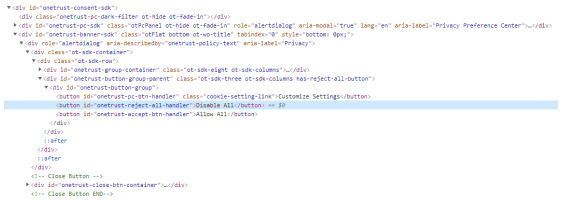


Abbildung 2.12: DOM von https://www.onetrust.com

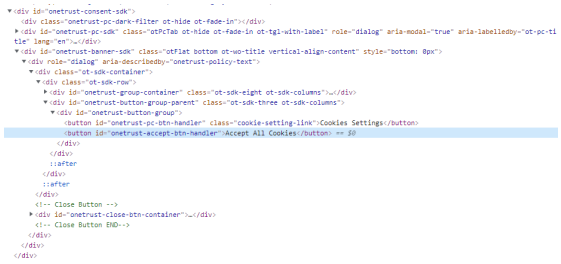


Abbildung 2.13: DOM von https://www.drax.com

Gemäss der Webseite von OneTrust wird ihre Lösung auf über 350'000 Seiten verwendet.[5] Iubenda, ein weiterer CMP, bietet seinen Service für 80'000 Seiten an.[6]

Letztendlich kamen wir zu dem Schluss, dass es nahezu unmöglich ist eine Lösung für jegliche Webseiten anzubieten. Aber in Anbetracht der Zahlen können wir, indem wir uns auf die CMPs fokussieren, eine Lösung anbieten, die auf den meisten seriösen Seiten funktionieren wird.

2.3 Lösungsvorschlag

In diesem Abschnitt wird beschrieben, für welchen Lösungsansatz wir uns entschieden haben und wieso wir genau diesen ausgewählt haben.

2.3.1 Entscheidung

Lösungsansatz 1 wäre an und für sich die bessere Alternative. Mit dieser Lösung wäre es möglich, komplett anonym Seiten zu besuchen, da die personenbezogenen Cookies gezielt gelöscht werden. Allerdings ist ein wichtiger Bestandteil der Arbeit, dass die Banner automatisch beantwortet werden. Unsere Analyse hat jedoch gezeigt, dass Banner nicht standardisiert sind und jede Webseite kann diese beliebig implementieren. Dementsprechend gestaltet sich die automatische Beantwortung der Banner als schwierig. Da für Lösungsansatz 1 es auch notwendig ist, einen Teil von Lösungsansatz 2 zu implementieren, haben wir uns dazu entschlossen uns voll auf Lösungsansatz 2 zu konzentrieren und diesen umzusetzen.

2.4 Anforderungen

Die Anforderungen werden durch die Use Cases definiert. Dementsprechend werden zuerst die Use Cases und im Nachhinein die nicht-funktionalen Anforderungen vorgestellt.

2.4.1 Use Cases

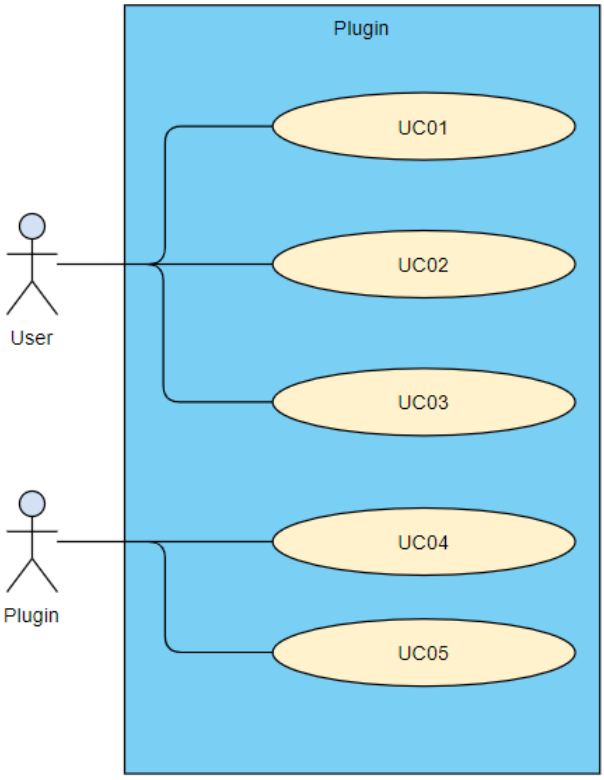


Abbildung 2.14: Übersicht der Use Cases

UC01 - Cookie Präferenzen festlegen

Primary Actor	Benutzer
Stakeholders and Interests	Der Benutzer kann seine Cookie-Präferenzen festlegen.
Preconditions	Das Plugin ist installiert.
Postconditions	Das Plugin bearbeitet die Cookie-Banner anhand der Präferenzen.
Main Success Scenario	1.) Der Benutzer klickt auf das Plugin. 2.) Das Popup wird geöffnet und bietet die Auswahlmöglichkeiten an. 3.) Der Benutzer wählt seine Präferenzen aus.
Frequency of Occurence	Wir gehen davon aus, dass der Benutzer diese Einstellung einmalig macht und danach seine Präferenzen nicht mehr oft ändert.

UC02 - Plugin kann angehalten werden

Primary Actor	Benutzer
Stakeholders and Interests	Der Benutzer kann das Plugin anhalten.
Preconditions	Das Plugin ist installiert.
Postconditions	Das Plugin wird angehalten und der Benutzer muss selber die Cookie-Banner auf den Webseiten wegklicken.
Main Success Scenario	1.) Der Benutzer klickt auf das Plugin. 2.) Das Popup wird geöffnet und der Button „Plugin pausieren“ wird angezeigt. 3.) Der Benutzer betätigt den Button.
Frequency of Occurence	Wir gehen davon aus, dass der Benutzer das Plugin nicht oft anhalten möchte.

UC03 - Aktive Cookies werden angezeigt

Primary Actor	Benutzer
Stakeholders and Interests	Der Benutzer kann die aktuell aktiven Cookies der Webseite anschauen.
Preconditions	1.) Das Plugin ist installiert. 2.) Der Benutzer besucht eine Webseite.
Postconditions	Die Cookies werden im Popup-Fenster angezeigt.
Main Success Scenario	1.) Der Benutzer klickt auf das Plugin. 2.) Das Popup wird geöffnet und die aktiven Cookies aufgelistet.
Frequency of Occurence	Wir gehen davon aus, dass der Benutzer wissen will, was für Cookies auf der aktuellen Seite aktiv sind, wodurch dieser UC oft auftreten wird.

UC04 - Webseiten ohne bekanntem Banner loggen

Primary Actor	Plugin
Stakeholders and Interests	Das Plugin loggt, wenn auf einer Webseite kein Banner erkannt wird.
Preconditions	1.) Das Plugin ist installiert. 2.) Der Benutzer besucht eine Webseite. 3.) Eine Netzwerkverbindung.
Postconditions	Ein Logfile, in dem die Webadressen, auf denen kein Banner detektiert wurde, gespeichert sind.
Main Success Scenario	1.) Der Benutzer klickt auf das Plugin. 2.) Der Benutzer wählt aus, dass Daten geloggt werden können. 3.) Das Plugin schickt die Daten an das Entwicklungsteam.
Frequency of Occurence	Sofern der Benutzer die Option erlaubt, dass das Plugin Webseiten loggen darf, wird der UC oft angewendet. Wenn der Benutzer die Option nicht aktiviert hat, wird der UC nie ausgeführt.

UC05 - Der Benutzer kann anhand des Plugin-Icons erkennen, ob und wie die Banner bearbeitet wurden

Primary Actor	Plugin
Stakeholders and Interests	Das Plugin-Icon ändert seine Farbe je nachdem, wie der Banner bearbeitet wurde (grün: gemäss Präferenzen bearbeitet, gelb: kein Banner detektiert, rot: Banner bearbeitet, jedoch nicht gemäss Präferenzen, blau: CMP detektiert, jedoch kein Banner).
Preconditions	1.) Das Plugin ist installiert. 2.) Der Benutzer besucht eine Webseite.
Postconditions	Angepasste Farbe des Plugin-Icons
Main Success Scenario	1.) Der Benutzer besucht eine Webseite. 2.) Das Plugin bearbeitet den Banner und aktualisiert das Icon. 3.) Der Benutzer sieht, wie der Banner bearbeitet wurde.
Frequency of Occurence	Bei jedem neuen Seitenaufruf wird der UC ausgeführt.

2.4.2 Nicht-funktionale Anforderungen

Scope

- Top 5 der CMPs sollen funktionieren.[7]
- Top 20 der meistbesuchten Webseiten in der Schweiz sollen funktionieren.[8]

Compatibility

- Das Plugin soll mit Google Chrome und Mozilla Firefox funktionieren, da diese die zwei beliebtesten Browser in der Schweiz sind.[9] Der Browser Safari wäre eigentlich auf Platz zwei, jedoch wird dieser nur für Apple Geräte supportet und wir möchten eine plattformunabhängige Lösung anbieten.[10]

Performance

- Sobald die Webseite komplett geladen ist, soll der Banner innerhalb von 2 Sekunden verschwinden.

Maintainability

- Es soll möglich sein, weitere CMPs im Plugin zu integrieren.
- Sollte kein Banner erkannt werden, wird die Webseite lokal in einem Logfile geschrieben.

Usability

- Die Erweiterung läuft, ohne dass der Benutzer mit dieser interagieren muss.

2.4.3 Wireframes

Das nachfolgende Wireframe veranschaulicht, wie das Popup des Plugins aussehen soll. Hierbei handelt es sich jedoch nur um eine Skizze des Popup Fensters. Das Endprodukt kann anders aussehen.

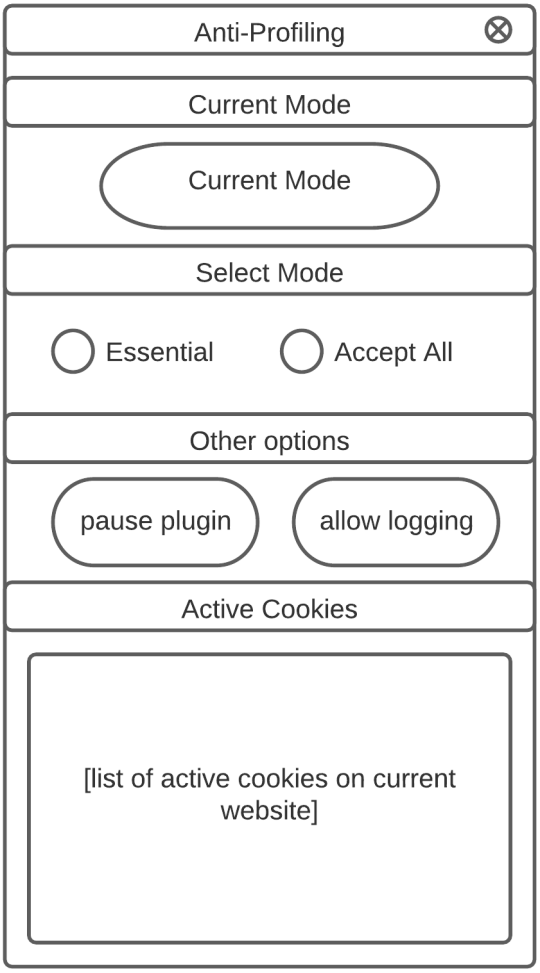


Abbildung 2.15: Wireframe

3 Umsetzung

3.1 Architektur

In diesem Abschnitt gehen wir darauf ein, wie unser Plugin aufgebaut ist. Dafür geben wir zunächst eine Übersicht von den einzelnen Komponenten mit einer Kurzbeschreibung und gehen danach, sofern erforderlich, detaillierter auf sie ein und begründen diverse Design-Entscheidungen.

3.1.1 Übersicht

Popup.js/Popup.html	Bietet ein einfaches GUI, dass dem User ermöglicht, Einstellungen für das Plugin vorzunehmen.
Background.js	Läuft im Hintergrund und injiziert unser Script in die Seite. Ausserdem ist es für die Änderung des Icons zuständig.
Script.js	Dient als Einstiegspunkt in das Content-Script.
Detector.js	Ist dafür zuständig, ein Cookie-Banner auf der Webseite zu detektieren.
Observer.js	Für den Fall, dass kein Banner detektiert werden konnte, observiert diese Klasse das DOM, um bei einer Änderung das Script erneut auszuführen, da der Banner eventuell nachgeladen wurde.
Handler.js	Diese Klasse ist für die Bearbeitung des Cookie-Banners zuständig.
Messenger.js	Schickt Nachrichten an Background.js, damit es weiss, wie das Icon gefärbt werden soll. Weiterhin hört es auf Nachrichten von Popup.js, um diesem Informationen über die Cookies liefern zu können.
Rules.json	Legt die Regeln fest, anhand derer Detector.js die Banner erkennt und Handler.js diese bearbeitet.

3.1.2 Allgemeine Entscheidungen

Eines unserer Ziele ist es, dass das Plugin in den Browsern Firefox und Chrome läuft. Dabei wollen wir soweit möglich den selben Code für beide Browser verwenden. Um dies zu bewerkstelligen, benutzen wir den Chrome Namespace, da Firefox diesen weitestgehend unterstützt. Weiterhin verwenden wir für das Manifest.json die Version 2, weil Firefox noch nicht die Version 3 umgesetzt hat.[11]

Damit wir unseren Code problemlos in Klassen unterteilen können, wird Webpack eingesetzt, dass den Code schlussendlich in eine Datei zusammenfügt.

3.1.3 Popup

Bei der Gestaltung der grafischen Benutzeroberfläche haben wir uns an Abbildung 2.15 gehalten. Das Ziel war es, die Benutzeroberfläche möglichst einfach und verständlich zu halten, jedoch sollte das GUI trotzdem ansprechend aussehen. Mit diesen Anforderungen haben wir uns für das Framework Bootstrap entschieden. Während dem Studium ist uns dieses Framework immer wieder über den Weg gelaufen, nur hatten wir nie die Gelegenheit es selbst auszuprobieren. Das Studium ist unter anderem auch dazu da, neue Technologien auszutesten und, da unser Hauptmerkmal auf der Funktionalität lag, kamen uns die vordefinierten CSS-Vorlagen von Bootstrap gerade gelegen.

Das Endresultat des GUIs kann man Abbildung 3.1 entnehmen.

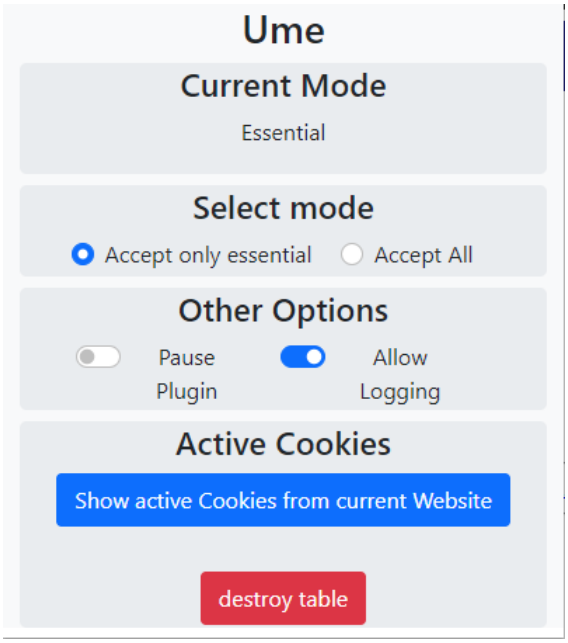


Abbildung 3.1: GUI des Plugins

3.1.4 Background

Background.js wird verwendet, um unser Content-Script zu injizieren, da wir damit mehr Kontrolle über den Zeitpunkt der Ausführung haben. So können wir sicherstellen, dass wenn das Plugin pausiert ist, das Script gar nicht erst in die Seite eingefügt wird.

3.1.5 Rules

Das Ziel des Plugins ist es, dass der Code generisch ist und möglichst gleich bleibt bei Erweiterungen. Um dies erreichen zu können, existiert Rules.json.

Rules.json ist prinzipiell ein Objekt, dessen Properties die einzelnen CMPs mit den jeweiligen Regeln für diesen sind. In einem CMP ist die erste Property „looking_for“, welche beschreibt, nach was für einem Attribut gesucht wird, z.B. „class“. Die zweite Property ist „detectors“ und beschreibt den Value vom Attribut, z.B. „uc-btn-more-info-banner“. Beide Properties sind Arrays, da es möglich ist, dass es mehrere Möglichkeiten für einen CMP gibt. Anhand diesen beiden Properties kann dann detector.js erkennen, ob ein CMP auf der Seite vorhanden ist, indem es das DOM nach einem Element mit dem Attribut von „looking_for“ und dem Value von „detectors“ durchsucht.

Alternativ kann man bei „looking_for“ auch „query“ definieren und in „detectors“ einen String angeben, der an einen QuerySelector übergeben werden soll, z.B. „[data-testid='uc-customize-anchor']“. Ebenfalls ist es möglich, nach einer Funktion zu suchen, die dann ausgeführt werden soll. Hierfür trägt man bei bei „looking_for“ „function“ ein und bei „detectors“ den vollständigen Namen der Funktion z.B. „window.parent.sp.openTrustWidget()“

Ein CMP hat standardmässig noch vier weitere Properties („options“, „reject_all“, „accept_all“, „close“), die für die Buttons in einem Banner stehen und wiederum die Properties „looking_for“ und „detectors“ enthalten.

Ein Beispiel eines CMPs ist in Abbildung 3.2 zu finden.

```
"OneTrust": {
  "looking_for": ["id"],
  "detectors": ["onetrust-consent-sdk"],
  "options": {
    "looking_for": ["id"],
    "detectors": ["onetrust-pc-btn-handler"],
    "checkboxes": {
      "looking_for": ["class"],
      "detectors": ["category-switch-handler"]
    },
    "confirm": {
      "looking_for": ["id", "class"],
      "detectors": ["confirm-choices-handler", "save-preference-btn-handler"]
    }
  },
  "reject_all": {
    "looking_for": ["id"],
    "detectors": ["onetrust-reject-all-handler"]
  },
  "accept_all": {
    "looking_for": ["id"],
    "detectors": ["onetrust-accept-btn-handler"]
  },
  "close": {
    "looking_for": ["class"],
    "detectors": ["banner-close-button"]
  }
},
```

Abbildung 3.2: CMP „OneTrust“ von Rules.json

Sollte man das Plugin um einen weiteren CMP oder eine Webseite erweitern wollen, genügt es im Normalfall ein neues Objekt in Rules.json zu erfassen.

3.2 Methodik

In diesem Abschnitt wird beschrieben, wie wir vorgegangen sind, um einen neuen CMP oder eine neue Webseite im Plugin einzubinden. Weiterhin gehen wir auf Schwierigkeiten ein, auf die wir während der Entwicklung trafen.

3.2.1 CMPs

Um die Regeln für einen CMP festzulegen, haben wir zunächst auf der Webseite nach einer Dokumentation gesucht, die eventuell den Aufbau des Banners beschreibt. Viele CMPs stellten auch eine Demo-Version zur Verfügung, anhand derer wir testen konnten, wie ihr Banner in eine Webseite integriert wird.

3.2.2 Webseiten

Bei den Top 20 Webseiten haben wir zunächst getestet, ob das Plugin mit den bereits implementierten CMPs funktioniert. War dies nicht der Fall, versuchten wir herauszufinden, ob ein anderer CMP verwendet wird und dann diesen umzusetzen. Bei den meisten ist es allerdings eine Eigenimplementation. Daher haben wir das DOM analysiert und anhand diesem die Regeln definiert, indem wir geschaut haben, wie wir die Elemente möglichst eindeutig identifizieren können.

3.2.3 Herausforderungen

Eine grosse Herausforderung war der Zeitpunkt, zu welchem das Content-Script injiziert werden soll. Zunächst wurde es injiziert, sobald der Status der Webseite auf „complete“ war. Dies führte jedoch dazu, dass es auf gewissen Seiten nicht funktionierte, da der Body des HTML-Dokumentes erst nachgeladen werden musste. Somit war unser Content-Script fertig, bevor der Cookie-Banner angezeigt wurde. Dieses Problem wurde gelöst, indem das Plugin, falls kein Banner detektiert wurde, den Body des HTML-Dokumentes observiert und bei einer Änderung erneut die Suche startet.

Ein weiteres Problem ist, dass die einzelnen CMPs sich teilweise stark voneinander in der Implementierung unterscheiden. OneTrust versieht die Buttons in ihrem Banner mit IDs und für die Checkboxen wird eine Klasse verwendet. Dadurch ist es sehr einfach, die einzelnen Elemente zu detektieren. Viele CMPs verwenden ein ähnliches Schema, wodurch es ausreichend war Rules.json anzupassen.

Eine Ausnahme hiervon bildet z.B. Secure Privacy. Secure Privacy verwendet für die Buttons keine Klassen, Ids oder andere Attribute, anhand derer man sie identifizieren könnte. Stattdessen hinterlegen sie eine Funktion, die beim Klick ausgeführt wird. Da wir mit unserem Content-Script allerdings keinen Zugriff auf Funktionen im Dokument haben, mussten wir eine Hilfsfunktion schreiben. Diese Funktion injiziert in das DOM ein Script-Tag, dessen Inhalt die Funktion ist, welche bei den Buttons registriert ist.

Mit steigender Anzahl der implementierten CMPs und Webseiten wurden die nötigen Code Änderungen jedoch immer weniger und mittlerweile reicht es in der Regel aus, nur Rules.json zu bearbeiten, um neue CMPs und Webseiten hinzuzufügen.

3.3 Testing

Das Testing stellt sich in unserem Fall als besonders schwierig da. Das Plugin an sich enthält relativ wenig Logik, für die es sich lohnt, Unit Tests durchzuführen. Der grösste Teil des Plugins besteht daraus, Elemente aus Webseiten herauszusuchen und Klicks auf diesen auszuführen. Grundsätzlich könnte man die Seitenaufrufe mit einem Tool wie Puppeteer automatisieren. Die Erstellung dieser Tests ist jedoch mühselig und zeitaufwendig, weshalb sie für uns wenig praktikabel sind. Weiterhin müssen wir, um die Funktionalität automatisiert testen zu können, nach dem Aufruf einer Webseite mittels Puppeteer überprüfen, ob der Banner tatsächlich entfernt wurde. Hierfür bräuchte es ein Tool, welches die Banner detektieren kann. Ein Teil unserer Arbeit ist genau dieser Aspekt, da wir zunächst Banner detektieren müssen, bevor wir sie bearbeiten können. Dadurch erachten wir automatische Tests als nicht möglich.

Um trotzdem die Funktionalität, abgesehen von manuellen Tests, testen zu können, haben wir uns dazu entschieden, einen Field Test zu machen, in dem Freiwillige das Plugin ausprobieren. Die Zusammenfassung der einzelnen Tests ist im Anhang zu finden.

3.3.1 Auswertung Field Test

Das Ziel des Field Tests war es unser Plugin in einem realen Umfeld zu testen und ein Feedback von Benutzern zu erhalten. Da jeder das Internet anders verwendet und auf unterschiedlichen Seiten unterwegs ist, können wir so möglichst objektiv die Funktionalität überprüfen. Wir als Entwickler können nur erahnen, ob wir unser Plugin auch wirklich einen Nutzen für Benutzer bringt.

Vorgehen

Das Testszenario besteht aus zwei Teilen. Zuerst sollen die Testpersonen zwei Tage lang im Inkognito-Modus normal surfen, ohne, dass das Plugin aktiviert ist. Die Tester sollen in dieser Zeit zählen, wie oft sie einen Cookie-Banner wegklicken mussten. Nach diesen zwei Tagen wird das Plugin aktiviert und man zählt wieder, wie viele Banner man wegklicken musste.

Ergebnisse

Vor der Installation des Plugins mussten unsere Testpersonen im Schnitt 22 Banner manuell wegklicken. Nach der Installation reduzierte sich die Anzahl der manuell weggeklickten Banner auf 11. Dies entspricht einer Reduktion von 50%. Da es hunderte an unterschiedlichen CMPs gibt und wir lediglich sieben davon implementiert haben (zum Testzeitpunkt waren es vier), sowie sichergestellt haben, dass es bei den Top 20 Webseiten funktioniert[8], sind wir mit diesem Ergebnis sehr zufrieden. Unsere Testpersonen gaben auch an, dass sie nach der Installation einen Unterschied im Vergleich zum normalen Surfen festgestellt haben.

Performance

Gemäss unseren Testern gibt es keinerlei Performance Einbusse, wenn man mit dem Plugin surft. Weiterhin sind bei ihnen auch keinerlei Probleme im Bezug auf das Plugin aufgetreten.

Kompatibilität

Die Probanden gaben an, dass auf den meisten Webseiten, die sie häufig besuchen, die Banner weggeklickt wurden. Dies zeigt, dass wir die am meisten relevanten Seiten abgedeckt haben. Ausserdem wird das Plugin nicht als störend empfunden.

Auf die Frage, ob es noch Webseiten gibt, die von unserem Plugin nicht abgedeckt sind und von denen man es gerne hätte, wurden lediglich die Seiten „Mediamarkt“ und „Saturn“ genannt. Darüber hinaus ist niemandem aufgefallen, dass eine Seite nicht mehr korrekt funktioniert, nachdem das Plugin den Cookie-Banner bearbeitet hat.

Usability

Die Bedienung empfanden die Testpersonen als einfach und verständlich. Ausserdem gefiel ihnen die optische Oberfläche des Popups.

Leider merkten die Probanden teilweise, dass die Banner vom Plugin weggeklickt werden. Unser Ansatz bei der Lösung ist, dass wir den Banner nach der Detektion unsichtbar schalten und das Ausfüllen somit vor dem Benutzer verborgen bleibt. Bei einigen CMPs funktioniert das hervorragend, da der komplette Banner von Beginn an im DOM enthalten ist. Bei anderen CMPs werden beim Klick auf „Optionen“ Elemente nachgeladen, teilweise findet sogar ein Redirect auf eine andere Webseite statt. Dies können wir selbstverständlich nicht vor dem Benutzer verbergen.

Weitere Bemerkungen

Auf die Frage, ob sie das Plugin installieren würden, wenn es im Store verfügbar wäre antworteten nahezu alle mit Ja. Nur eine Person sagte, dass sie sich unsicher ist, da sie die Banner nicht so störend findet. Da unsere Testpersonen hauptsächlich Schweizer sind, gehen wir davon aus, dass sie primär auf Schweizer Webseiten unterwegs waren. Das Schweizer Datenschutzgesetz schreibt zum jetzigen Zeitpunkt noch nicht vor, dass die Betreiber von Webseiten die Benutzer auf den Datenschutz hinweisen müssen. Die Schweiz hat jedoch eine Totalrevision des Datenschutzgesetzes verabschiedet, welches voraussichtlich in der zweiten Jahreshälfte 2022 in Kraft treten soll. Wir vermuten, dass mit dem neuen Datenschutzgesetz die Anzahl der Banner auf Schweizer Seiten ansteigen wird und somit das Interesse auch für Schweizer Benutzer steigt.[12]

Ausserdem gab es noch eine Bemerkung, dass das Icon anspruchsvoller aussehen könnte. Der Fokus des Arbeit liegt jedoch darauf, die Funktionalität aufzuzeigen, weshalb dieser Punkt für uns vernachlässigbar ist.

Fazit

Zusammenfassend kann man sagen, dass der Test erfolgreich war. Unsere Testpersonen sind alles in allem sehr zufrieden mit dem Plugin. Wir haben neue Erkenntnisse gewonnen und nützliches Feedback für weitere Verbesserungen erhalten. Anhand der Testergebnisse kann nun für einen Release geplant werden.

3.4 Limitation der Anforderungen

In den nicht-funktionalen Anforderungen haben wir definiert, dass wir, unter anderem, die Top 5 CMPs[7] einbinden würden. Dies ist uns leider nicht ganz gelungen. Den CMP Segment konnten wir nicht implementieren. Die Herausforderung hier ist, die Elemente zu detektieren, damit wir die Aktionen ausführen können. Das beginnt bereits beim Root Element.

Unsere Lösung basiert darauf, dass die Banner von jedem Provider gleich aufgebaut sind und wir über Attribute der Elemente auf diese zugreifen können. Bei diesem CMP ist dies jedoch nicht der Fall.

Nachfolgend sieht man einen Ausschnitt des DOMs von den Webseiten `https://landbot.io` und `https://segment.com`. Hier ist schön zu erkennen, dass die Struktur an sich sehr ähnlich ist, aber abgesehen davon wenige Gemeinsamkeiten bei den einzelnen Elementen vorhanden sind.

Abbildung 3.3: Vergleich RootNode von `https://landbot.io` (links) und `https://segment.com` (rechts)

Um eventuell eine Lösung zu finden, haben wir den Source Code von Segment angeschaut. Die Lösung von Segment basiert auf React. Beim Lesen der Dokumentation bestätigte sich unsere Vermutung, dass Segment dem Benutzer die Freiheit gibt, selbst die Banner zu gestalten. Dabei fängt es schon damit an, dass noch nicht einmal die Root-Node gleich ist, da die Benutzer diese selbständig festlegen müssen.

Schlussendlich sind wir auf einen Eintrag auf der Webseite gestossen [13]. Hier wird beschrieben, dass das UI vom Benutzer beliebig angepasst werden kann. Dies hat unsere Vermutung definitiv bestätigt, dass die Banner nicht standardisiert sind und der Benutzer den Banner beliebig gestalten kann. Mit dieser Info haben wir feststellen müssen, dass es zeitlich sehr aufwändig wäre zu prüfen, ob eine Webseite Segment als CMP im Einsatz hat oder nicht.
Aus diesen Gründen haben wir uns entschieden, Segment nicht zu integrieren. Dafür haben wir drei weitere CMPs integriert, welche nicht in den TOP5 aufgelistet waren. Mit insgesamt sieben CMP Implementationen haben wir aufgezeigt,

dass unser Ansatz funktioniert und es grundsätzlich möglich ist, die Cookie-Banner automatisch zu bearbeiten.

3.5 Known Issues

Das Plugin ist funktionstüchtig und läuft stabil, jedoch sind uns zwei Fehler aufgefallen, auf die wir eingehen wollen.

3.5.1 Farbänderung des Icons

Der erste Fehler bezieht sich auf die Farbänderung des Icons. Hier gibt es insgesamt vier Modi, wie in UC05 beschrieben. Die Farbe Blau zeigt an, dass wir einen CMP detektiert haben, jedoch kein Banner angezeigt wird. Dies impliziert, dass der Banner bereits einmal von uns oder von dem Benutzer bearbeitet wurde.

Fälschlicherweise passiert es, dass das Icon sich gelb färbt, obwohl ein Cookie-Banner auf dieser Seite bearbeitet wurde. Eine Seite, bei der dieses Verhalten zu beobachten ist wäre z.B. https://www.google.ch/. Der Fehler tritt auf, da, nachdem der Banner bearbeitet wurde, dieser komplett aus dem DOM verschwindet und bei einem erneuten Aufruf der Webseite auch nicht mehr vorhanden ist. Dadurch detektiert das Plugin auch keinen Banner und zeigt „korrekterweise“ das gelbe Icon an.

Wir haben uns zwei Lösungsansätze überlegt, mit denen wir diesen Fehler beheben könnten. Der erste Ansatz ist, dass wir die URL von jeder Webseite, auf die wir einen Cookie-Banner bearbeitet haben, speichern und bei einem Seitenaufruf zuerst diese Liste mit der URL abgleichen, bevor wir versuchen einen Banner zu detektieren. Dies würde allerdings zu einem weiteren Problem führen. Wenn ein Benutzer entscheidet, die Cookies aus seinem Browser zu löschen, wären die URLs immer noch in unserem Plugin gespeichert und somit müsste der User die Banner auf diesen Seiten manuell wegklicken. Da es deutlich wichtiger ist, dass das Plugin die Banner richtig bearbeitet, als das die Farbe korrekt angezeigt wird, haben wir diesen Ansatz verworfen.

Beim zweiten Ansatz haben wir uns überlegt, dass die Banner, sobald sie einmal akzeptiert oder abgelehnt wurden, nicht mehr angezeigt werden. Wir stellten uns also die Frage, woher die Webseiten wissen, ob sie einen Banner anzeigen sollen oder nicht. Die naheliegende Antwort darauf ist, dass Cookies dafür verwendet werden. Also haben wir versucht unsere Theorie zu bestätigen, indem wir Cookies von Seiten analysiert haben, nachdem der Banner bearbeitet wurde.

Abbildung 3.4: Cookie für akzeptierten Banner bei https://www.bundesliga.de/ (links) und Cookie für akzeptierten Banner bei https://www.google.ch/ (rechts)

In Abbildung 3.4 sieht man, welche Cookies dafür sorgen, dass der Banner bei https://www.bundesliga.de/ bzw. https://www.google.ch/ nicht mehr angezeigt wird. Es ist anzunehmen, dass andere Webseiten es ähnlich handhaben.

Indem wir im Voraus überprüfen, ob diese Cookies gesetzt sind, könnten wir unser Problem lösen. Allerdings würde das bedeuten, dass wir für jede Seite analysieren müssten, welches Cookie dafür verantwortlich ist. Dies würde den Rahmen unserer Arbeit sprengen. Da der Fehler nicht schwerwiegend ist, haben wir uns dazu entschlossen, diesen nicht weiter zu behandeln.

3.5.2 Iframe Injection

Der zweite Fehler bezieht sich auf die CMP Sourcepoint und TrustArc. Bei diesen beiden ist der Cookie-Banner in einem Iframe-Element. Dies führt dazu, dass, wenn man versucht auf die Elemente im Banner zuzugreifen, ein Cross-Origin Fehler geworfen wird. Man kann allerdings im Background.js festlegen, dass das Content-Script auch in die Iframes injiziert wird, wodurch der Fehler umgangen wird. Wir haben das im Background.js umgesetzt, jedoch wird das Script auf manchen Webseiten nicht ausgeführt.

Hier liegt wieder das Problem mit dem Nachladen von Inhalt vor. Prinzipiell kommt es vor, dass die Iframes noch nicht im DOM enthalten sind, wenn unser Content-Script injiziert wird, wodurch es auch nicht in den Iframes ausgeführt wird. Man kann den Fehler beheben, indem detector.js zunächst das Iframe detektiert und danach das Content-Script in dieses injiziert wird. Da die beiden CMPs die letzten waren, die wir implementierten, hat es zeitlich nicht mehr gereicht, diesen Fehler zu beheben.

3.6 Risiken

Die Grundidee unseres Plugins ist es, die Banner durch Klicks auf die einzelnen Buttons automatisch auszufüllen und zu bestätigen. Hierfür werden die Elemente im DOM anhand von Regeln gesucht und danach die `click()` Methode aufgerufen bzw. die Funktion, welche im `onclick listener` registriert ist. Die Regeln sind dabei z.B. Ids oder Klassen. Da es jedem freisteht, die Attribute nach Belieben zu benennen und es keinen Standard hierfür gibt, könnte es vorkommen, dass zufällig jemand die selben IDs verwendet, wie ein CMP. Dadurch könnte es zu unerwünschten Nebeneffekten kommen, wenn das Plugin auf einmal auf dieser Seite Klicks ausführt. Sollte das Plugin sich grosser Beliebtheit erfreuen, wäre es nicht auszuschliessen, dass Verbrecher bewusst Elemente wie ein CMP benennen um z.B. automatisch Viren herunterzuladen beim Aufruf der Seite.

3.7 Fazit und Ausblick

3.7.1 Fazit

Wir haben verschiedene Lösungsansätze eruiert und einen Prototypen entwickelt, welcher die Cookie-Banner automatisiert bearbeitet und somit das Ziel der Aufgabenstellung erreicht. Das Plugin funktioniert in den Browsern Google Chrome und Mozilla Firefox und der Benutzer muss nicht mit dem Plugin interagieren, da standardmässig nur die essentiellen Cookies akzeptiert werden. Sofern die Funktion vom User aktiviert wird, kann das Plugin Webseiten loggen, auf denen kein Banner detektiert wird. Es ist möglich, mit wenig Aufwand die Funktionalität um weitere CMPs zu erweitern, indem die entsprechenden Regeln in Rules.json definiert werden.

Bis auf die Umsetzung eines CMPs wurden alle nicht-funktionalen Anforderungen erfüllt. Allerdings kann das Plugin in der jetzigen Form bereits verwendet werden und bietet dem Benutzer einen Mehrwert.

In einem Field Test konnten wir das Plugin unter realen Bedingungen testen. Dadurch wurde sichergestellt, dass die Funktionalität gegeben ist und die Benutzererfahrung verbessert wird.

3.7.2 Ausblick

Das Plugin wird im Anschluss dieser Arbeit nicht mehr von uns weiterentwickelt werden. Der Code wird allerdings an das INS der OST übergeben und es steht ihnen frei das Plugin zu erweitern.

Hierfür wäre zunächst die Implementierung von weiteren CMPs wichtig um mehr Webseiten abdecken zu können.

Die Logging-Funktion sollte erweitert werden, sodass die Webseitenadressen nicht nur lokal gespeichert, sondern auch an einen Server übermittelt werden. Dadurch kann man priorisieren, welche CMPs man als nächstes implementieren will, indem man die Logs auswertet.

Zu guter Letzt, wäre die Darstellung eines kurzen Infotextes im Pop-up des Plugins, was die Funktion der einzelnen Cookies ist, interessant.

4 Anhang

4.1 Zeitplan

4.1.1 Phasen

Inception

Diese Phase läuft vom 30.09.2021 bis zum 06.10.2021. In dieser Phase überlegen wir uns unterschiedliche Möglichkeiten, mit der die Aufgabenstellung gelöst werden kann.

Elaboration

Diese Phase läuft vom 07.10.2021 bis zum 27.10.2021. In dieser Phase erarbeiten wir die Vor- und Nachteile der einzelnen Möglichkeiten und vergleichen sie untereinander, damit wir uns am Ende der Elaboration für eine entscheiden können, mit der wir die SA angehen.

Construction

Diese Phase läuft vom 28.10.2021 bis zum 09.12.2021. Während der Konstruktion wird die Software-Komponente der SA entwickelt.

Transition

Diese Phase läuft vom 10.12.20212 bis zum 23.12.2021. In der letzten Phase sollen keine Eweiterungen mehr an der Software durchgeführt werden, sondern nur noch ein Feinschliff. Weiterhin soll die Dokumentation fertiggestellt und für die Abgabe bereit gemacht werden.

4.1.2 Meilensteine

M1 Brainstorming

Datum: 06.10.2021
Für den ersten Meilenstein sollen Ideen gefunden werden, mit denen man die Aufgabenstellung der SA lösen kann.

Arbeitsprodukte

- Liste der Lösungsvorschläge mit deren Beschreibung

M2 End of Elaboration

Datum: 27.10.2021
Für die End of Elaboration soll die Analyse der Lösungsvorschläge und die Entscheidung auf einen Lösungsvorschlag fallen.

Arbeitsprodukte

- Angenommener Lösungsvorschlag mit Begründung

M3 Prototyp

Datum: 10.11.2021
Für diesen Meilenstein soll ein Prototyp fertiggestellt sein, der die grundlegende Funktionalität der Software beherrscht.

Arbeitsprodukte

- Prototyp
- Demonstration der Funktionalität an einem Use Case

M4 End of Construction

Datum: 09.12.2021
Nach diesem Meilenstein soll keine weitere Funktionalität hinzugefügt werden.

Arbeitsprodukte

- Ansatz eines Softwareproduktes, das in einer weiteren Arbeit weiterentwickelt werden kann

M5 Abstract Submission

Datum: 21.12.2021
Mit diesem Meilenstein wird das Abstract im Abgabetool abgegeben.

Arbeitsprodukte

- Abstract

M6 Final Submission

Datum: 23.12.2021
Mit diesem Meilenstein ist die SA abgeschlossen.

Arbeitsprodukte

- Projektbericht

4.2 Aufgabenstellung

INSTITUTE FOR NETWORKED SOLUTIONS

OST Ostschweizer Fachhochschule

HS 2021

Studien- und Bachelorarbeiten

Prof. Dr. Nathalie Weiler

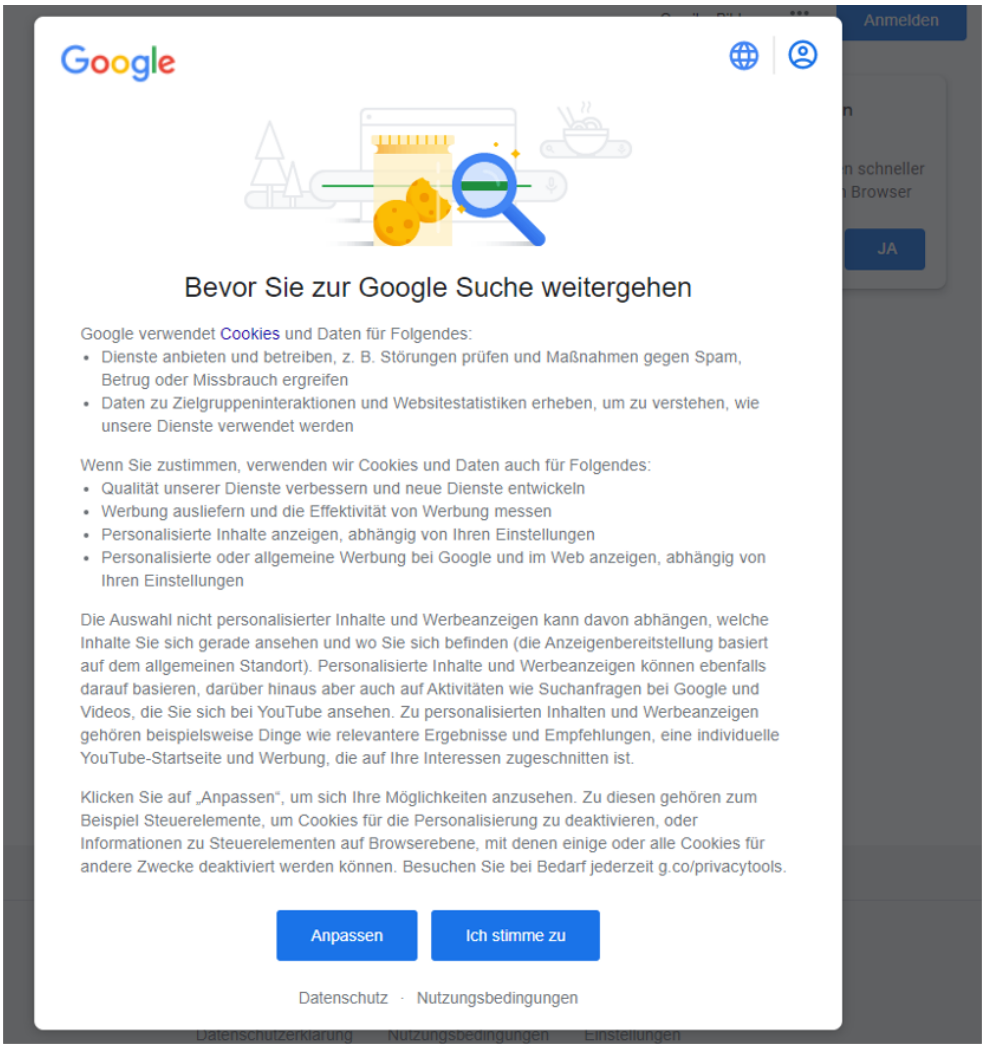
Version 1.0
30. September 2021
Institute for Networked Solutions

Teil II.

Bojan Dakic und Fabian Tiri HS2021

1. Aufgabenstellung SA Anti-Profiling

1.1. Einführung



Wer kennt die lästigen Pop-ups bei der Google-Suche z.B. nicht. Bei jeder Suche wird man genötigt sich in sein Profil einzuloggen, damit das Verhalten besser indexiert werden kann oder eben die Pop-up Fenster wegzublicken.

In dieser Arbeit sollen jetzt Ideen entwickelt werden wie man mittels on-the-fly generierten User Profilen diese lästigen Popups effizient automatisiert zu beantworten und dem Profiling entgegen zu wirken.

1.2. Aufgabe

Im ersten Teil der Arbeit werden die Ansätze zur generischen Profilerstellung on the fly entwickelt, im zweiten Teil wird ein passender Prototyp entwickelt.

Als Minimalziel soll in einem Browser die Funktionalität zeigen.

1.3. Erwartete Resultate

- Entwicklung von Lösungsansätzen und deren Beurteilung
- Prototypartige Umsetzung eines Ansatzes
- Dokumentation der Arbeit (inklusive Vorgehen und kritische Bewertung der getroffenen Entscheide)

2. Organisatorisches

2.1. Bewertung

Die Bewertung der Arbeit erfolgt nach dem für Studienarbeiten vorgegebenen Bewertungsschema (Gewichtung in Klammern):

1. (1/5) Organisation und Durchführung: Projektplanung und Nachführung der Arbeit gemäss Projektplan, Zusammenarbeit mit dem allfälligen Auftraggeber und der Betreuerin
2. (1/5) Bericht: Inhalt des Projektschlussberichtes, Gliederung, Darstellung und Sprache der Dokumentation
3. (3/5) Resultat der Arbeit:
   a) Problemanalyse: Vorstudie, Literaturstudium, Anforderungsspezifikation, Anforderungsanalyse
   b) Lösungsentwurf: Lösungsvarianten und deren Beurteilung, Variantenentscheid, Konzept und Entwurf
   c) Realisierung und Test

2.2. Verfassen der Arbeit

An der OST hat Prof. Dr. Zimmermann hat eine Zusammenstellung erstellt mit Hinweisen zum Schreiben einer wissenschaftlichen Arbeit.

Weitere Angaben zur Studienarbeit im allgemeinen befinden sich auf dem Teams Channel zum Studiengang.

2.3. Termine

Die offiziellen Termine für diese Studienarbeit sind soweit nicht anders verbindlich zwischen Studierenden und Betreuerin geregelt:

- Die Studienarbeit beginnt am 20. September 2021.
- Der Abgabetermin für den Abstract im Abgabetool ist am 21. Dezember 2021.
- Der Abgabetermin der Studienarbeit ist am 23. Dezember 2021 bis 17 Uhr.

2.4. Betreuung

Die Studienarbeit wird durch Prof. Dr. Nathalie Weiler betreut.

Eine regelmässige, wöchentliche Besprechung wird zwischen Studierenden und Betreuerin am Anfang der Arbeit festgelegt: Donnerstag, 9-10 Uhr

Ausgabe: Rapperswil, 25. September 2021

Nathalie Weiler

Prof. Dr. Nathalie Weiler
Institutspartnerin INS

4.3 Field Test

4.3.1 Wissensziele, Fragestellungen und Hypothesen

- Ist das Plugin gut bedienbar?
- Sind die Begriffe verständlich?
- Was für Probleme treten auf?
- Gibt es Webseiten, welche häufig besucht werden, jedoch vom Plugin nicht behandelt werden?

4.3.2 Testform und Testinhalt

- Testform: Field Test.
- Funktionalität des Plugins.

4.3.3 Geeignete Testpersonen

- Eigentlich nicht so eng. Alle Personen, die mit Google Chrome oder Firefox im Internet surfen.
- Demographie: ab 16 Jahren.

4.3.4 Anleitung

Für den Test ist es zunächst erforderlich für zwei Tage im Inkognito-Modus zu surfen und festzuhalten, wie oft man Cookie-Banner wegklicken muss. Nach den zwei Tagen muss man das Plugin installieren und weitere zwei Tage im Inkognito-Modus surfen und ebenso festhalten, wie oft man Cookie-Banner manuell wegklicken musste.

4.3.5 Testdurchführung

Vorbereitung

Installation Plugin

Google Chrome

1. Die ZIP-Datei entpacken.
2. chrome://extensions/ öffnen.
3. Oben rechts den Entwicklermodus aktivieren.
4. **Entpackte Erweiterung laden** anklicken.

5. Zu dem entpackten Ordner vom Plugin navigieren und mit dem Button **Ordner auswählen** bestätigen.
6. Oben rechts auf das Puzzle Symbol klicken und das Plugin anpinnen.
7. Beim Plugin **Details** anklicken und auswählen, dass es im Inkognito-Modus verwendet werden soll.

Mozilla Firefox

1. about:debugging#/runtime/this-firefox öffnen.
2. **Temporäres Add-On laden** anklicken.
3. Die ZIP Datei auswählen und auf **Öffnen** klicken.
4. about:addons öffnen.
5. Das Plugin anklicken und **In privaten Fenster ausführen erlauben** auswählen.

Anmerkung: Bei Firefox muss das Plugin nach jedem Neustart des Browsers wieder hinzugefügt werden.

Einführung

Es geht darum etwas neues auszuprobieren. Deshalb werfen wir dich ins „kalte“ Wasser. Das wäre auch in der Realität der Fall. Man wird vermutlich damit konfrontiert, ohne es zuvor zu ahnen.
Du kannst allerdings nichts falsch machen. Wir testen nicht dich, sondern unser Plugin. Wir möchten deine ehrliche Meinung.

Angaben zur Person

- Alter:
- Geschlecht:

Legende

	Legende		
++	trifft in hohem Mass zu	-	trifft eher nicht zu
+	trifft eher zu	--	trifft gar nicht zu
0	trifft mehr oder weniger zu	kinb	kann ich nicht beurteilen

Vor der Installation des Plugins

Szenario	**Anzahl**
Wie oft musste man Cookie-Banner wegklicken (ungefähre Zahl reicht)	22

Nach der Installation des Plugins

Szenario	Anzahl
Wie oft musste man Cookie-Banner wegklicken (ungefähre Zahl reicht)	11

Hast du einen Unterschied gemerkt nach der Installation?

– Ich muss keine Cookie-Banner wegklicken.
– Ja.
– Ja, man musste weniger Banner ausfüllen.

Performance

Szenario	++	+	0	-	--	kinb
Ich merke keine Performance Einbusse beim Surfen, wenn ich das Plugin benutze	X					

Falls Probleme auftraten, was waren das für Probleme?

– Es sind keine Probleme aufgetreten.

Kompatibilität

Szenario	++	+	0	-	--	kinb
Das Plugin klickt die meisten Banner auf Webseiten weg, die ich besuche		X				
Das Plugin ist nicht störend beim Surfen		X				

Auf folgenden Webseiten hätte ich es gerne, dass die Banner weggeklickt werden.

– mediamarkt.de
– saturn.de
– Meine meist besuchten Webseiten hatten entweder schon vorher keinen Banner bzw. Banner wurden mit dem Plugin weggeklickt.

Funktionieren gewisse Webseiten nicht mehr ordnungsgemäss, wenn das Plugin den Cookie-Banner weggeklickt hat, sprich, wenn das Plugin Icon nicht Gelb ist? Falls ja, welche Webseiten?

– Ist mir nicht aufgefallen.
– Bei mir haben alle Seiten normal funktioniert.

Usability

Szenario	++	+	**0**	**-**	**--**	**kinb**
Die Auswahlmöglichkeiten im Plugin sind verständlich.		X				
Ich merke nicht, dass das Plugin die Banner wegklickt.			X			
Die Oberfläche des Plugins gefällt mir.		X				
Die Bedienung des Plugins ist einfach	X					
Die Begriffe im Plugin sind verständlich		X				

Weitere Bemerkungen

Würdest du das Plugin nutzen, wenn du es im Store herunterladen könntest?

– Ja.
– Ja, würde ich.
– Weiß nicht. Eigentlich sind die Banner nicht so störend, man füllt die einmal aus und damit hat es sich.

Hier hast du die Möglichkeit weiteres Feedback zu geben.

– Das Icon könnte etwas anspruchsvoller sein.
– Bei den meisten Seiten ist kurz ein Banner aufgepopt und sofort wieder verschwunden (ab und zu war noch nicht einmal der Banner zu sehen). Allerdings sieht es bei manchen Seiten recht komisch aus z.B. Google oder Amazon.

Vielen Dank, dass du dir Zeit genommen hast, unser Plugin zu installieren und zu testen.

4.4 Literaturverzeichnis

[1] URL: https://eur-lex.europa.eu/legal-content/DE/TXT/?uri=CELEX:32009L0136. Zugegriffen am: 10.12.2021.

[2] URL: https://www.bitkom.org/Presse/Presseinformation/Cookie-Banner-stoeren-Internetnutzer.html. Zugegriffen am: 10.12.2021.

[3] URL: https://www.kaspersky.com/resource-center/definitions/cookies. Zugegriffen am: 25.10.2021.

[4] URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies. Zugegriffen am: 25.10.2021.

[5] URL: https://www.onetrust.com/products/cookie-consent/. Zugegriffen am: 27.10.2021.

[6] URL: https://www.iubenda.com/de/?utm_source=adwords&utm_medium=ppc&utm_campaign=aw_brand_germany_exact&utm_term=iubenda&utm_content=494571748765&gclid=Cj0KCQjw8eOLBhC1ARIsAOzx5cHgCqdrm5UcIwGGI2etfOrtYhzwcB. Zugegriffen am: 27.10.2021.

[7] URL: https://www.g2.com/categories/consent-management-platform-cmp?__cf_chl_captcha_tk__=pmd_NsbP9kp1iSZrR_kdHmSSTRW.QXqYuHPzsBL0h7gaejg-1634561770-0-gqNtZGzNA1CjcnBszQi9. Zugegriffen am: 18.10.2021.

[8] URL: https://www.similarweb.com/de/top-websites/switzerland/. Zugegriffen am: 27.10.2021.

[9] URL: https://de.statista.com/statistik/daten/studie/431513/umfrage/marktanteile-der-browser-bei-der-internetnutzung-in-der-schweiz/. Zugegriffen am: 23.11.2021.

[10] URL: https://support.apple.com/de-ch/HT204416/. Zugegriffen am: 23.11.2021.

[11] URL: https://blog.mozilla.org/addons/2021/05/27/manifest-v3-update/. Zugegriffen am: 11.12.2021.

[12] URL: https://www.bj.admin.ch/bj/de/home/aktuell/mm.msg-id-84103.html. Zugegriffen am: 10.12.2021.

[13] URL: https://segment.com/blog/how-to-build-consent-management-into-your-site-in-less-than-a-week/. Zugegriffen am: 08.12.2021.

4.5 Abbildungsverzeichnis

4.6 Glossar

Bootstrap	Bootstrap ist ein open source Frontend-Framework. Die Dokumentation ist hier zu finden
CMP	*Consent Management Provider:* Ein CMP bietet eine generische Lösung für die Cookiebanner an, damit Seiten diese nicht selbständig implementieren müssen.
Content-Script	Ein Script, welches Zugriff auf das DOM einer Webseite hat.
Cookie	Eine kleine Textdatei, die Informationen beim Browsen an den Server übermittelt.
CSS	*Cascading Style Sheets:* Ist eine Stylesheet-Sprache, um das Aussehen von HTML Elementen festzulegen.
DOM	*Document Object Model:* Ein DOM stellt ein HTML oder XML Dokument als einen Baum dar.
DSGVO	*Datenschutz-Grundverordnung:* Die DSGVO ist eine Verordnung der Europäischen Union, die den Datenschutz von EU-Bürgern verschärft.
Field Test	Ein Test bei dem die Funktionalität einer Software unter Realbedingungen getestet wird.
Framework	Ein Framework ist ein Programmiergerüst, dass z.B. Vorlagen oder Standardmodule zur Verfügung stellt.
GUI	*Graphical User Interface:* Die grafische Benutzeroberfläche
JSON	*JavaScript Object Notation:* ein Format um Daten strukturiert darstellen zu können.
Plugin	Eine Software, welche bestehende Software erweitert.
Puppeteer	Eine Node Library mit der man Chrome mittels Code steuern kann.
Unit Test	Tests, die die korrekte Ausführung von einzelnen Funktionen sicherstellen sollen.
Webpack	Ein Tool, welches in der Lage ist mehrere Dateien in eine zusammenzufügen. Weitere Informationen finden sich auf der Homepage.

ZIP Ein Dateiformat um Daten zu komprimieren bzw. um mehrere Dateien in eine zu bündeln.