VAULT PASSWORD BROWSER EXTENSION

IPA-Bericht

Hunter Wilson

Inhaltsverzeichnis

Teil 1: Umfeld und Ablauf

1 Aufgabenstellung

(Direkt aus die PkOrg.ch Webseite kopiert)

1.1 Ausgangslage

Die Wunderman Thompson Switzerland AG betreibt einen selbstentwickelten Passwort-Manager namens Vault, um private sowie geteilte Accounts und Zugänge zu verwalten. Dieses Tool besitzt sein eigenes Web-Interface. Vault kann nur über das firmeninterne Netzwerk erreicht werden.

Die vorliegende IPA soll die Usability dieses Tools verbessern. Hierzu wird eine Browser Extension für die Browser Google Chrome und Microsoft Edge entwickelt, welche die entsprechenden Accounts und dazugehörigen Passwörter im jeweiligen Kontext anzeigt.

1.2 Detaillierte Aufgabenstellung

Der zurzeit verwendete Passwortmanager wird um eine Browser Extension erweitert, damit die Lösung über die Browser Chrome und Edge hinweg nahtlos verwendet werden kann.

Folgende Elemente der Arbeit bilden den Pflichtteil:
Entwicklung der Browser Extension:
Die Erstellung beinhaltet ein vereinfachtes User Interface, welches den nachfolgenden Funktionsumfang erlaubt:
- Das Nutzen von Passwörtern, welche als Favoriten markiert sind
- Anbieten einer Auswahl nach URL, Name
- Einfaches Copy & Paste von Benutzername und Kennwort
- Unterstützung der Browser Edge (ab Version 110 aufwärts) und Chrome (ab Version 109 aufwärts). Die Versionen aus 2023 oder neuer werden damit abgedeckt.
- Aufruf des Vault-Eintrages im Web Interface für erweiterte Funktionalität

Die Umsetzung der Benutzerautorisierung via vorgegebener Benutzerrolle und Integrated Security via Kerberos:
Die Authentisierung wird in den Einstellungen der Extension verwaltet und soll mittels Standardfunktionalität des genutzten Technologiestacks umgesetzt werden (aka Integrated Security), Vorzugsweise mittels Kerberos Delegation der Windows-Benutzersession. (Das Web Frontend unterstützt dies bereits) Es soll nur die Zugangsberechtigungen des jeweiligen User berücksichtigt werden und nur die entsprechenden Einträge an die Browser Extension gesendet werden.

- Integrationskonzept ans existierende Backend (Vault)
Jegliche API Calls, welche für die Extension verwendet werden, werden im Integrationskonzept schriftlich festgehalten. Dies umfasst: API Call; Verwendungszweck des API Calls; Parameter, welche im Call mitgegeben werden; Antwort des API Calls.

- Umsetzung automatisierbares Testset, welches Integration und Funktionalitätstest umfasst, um die Stabilität der API und sonstige Abhängigkeiten zu gewährleisten. Happy Path Test Cases sind ausreichend. Dieses Testset umfasst mindestens einen Test der folgenden Testarten:
-Unit Test, welche einzelne Funktionalität der Browser Extension testen
-Integration Test, welche die Integration von der Browser Extension zu einem gemocketen Backend testen
-Systemtest, welche die komplette Lösung testen

Dies dient der Sicherstellung, dass die neue Lösung keine existierende Funktionalität bricht und zukünftige Changes nicht die neue Lösung brechen.

- Die Arbeit wurde entsprechend dem MVC design pattern implementiert und verletzt nicht die clean code Prinzipien.

- Die Zuverlässigkeit der Browserextension wird über eine lückenlose Fehlerbehandlung im ganzen Code konsistent gewährleistet.

1.3 Mittel und Methoden

Für die Entwicklerumgebung wird Visual Studio Code verwendet. Als Programmiersprache kommen HTML, CSS und JavaScript zum Einsatz.

Zur Verfügung stehen dem ausführenden Praktikanten sein Firmenlaptop sowie ein eigener Arbeitsplatz. Ein Austausch mit Fachverantwortlichen ist persönlich oder remote gegeben.

1.4 Vorkenntnisse

Der Praktikant verfügt bereits über Kenntnisse in den Programmiersprachen HTML, CSS und JavaScript.

1.5 Vorarbeiten

Der Praktikant analysiert den bestehenden Passwortmanager "Vault" im Zuge der Vorbereitung. Des weiteren arbeitet sich der Praktikant in die Thematik der Chrome- und Edge-Browserwerweiterungen ein.

1.6 Neue Lerninhalte

Die Thematik vpn Browsererweiterungen für Chrome und Edge ist für den Praktikanten grundsätzlich ein neues Feld. Hierzu gehört auch die gesamthafte Verwaltung inkl. Zuweisung der Passwörter. Dies beinhaltet auch die sichere Übermittlung der Passwörter.

Die Arbeit mit dem Authentifizierungsdienst Kerberos stellt ebenfalls einen neuen und bisher nicht bekannten Bereich dar.

Als eine Quelle wird dem Kandidaten die bereits bestehende Dokumentation zu "Vault" seitens Wunderman Thompson dienen. Ansonsten werden sich die bezogenen Informationen auf öffentlich zugängliche, web-basierte Quellen stützen.

1.7 Arbeiten in den letzten 6 Monaten

Der Lernende bzw. Praktikant hat in den letzten 6 Monaten tatkräftig an Kentico-basierten Kundenprojekten mit C# gearbeitet. Hierzu gehören Arbeiten am Front- sowie Backend und Code. Der Praktikant hat dabei vorrangig an Terminbuchungstools via React-App für verschiedene Kunden gearbeitet und dabei Änderungen am Buchungsprozess in der Test- sowie Liveumgebung durchgeführt. Der Praktikant hat hierbei nicht nur Kundenvorgaben umgesetzt, sondern auch eigene Lösungsvorschläge eingebracht.

Des weiteren hat der Praktikant während eines Monats mittels Godot-Engine eine Schach-Applikation programmiert.

2 Projektaufbauorganisation

(Die folgende aufgelistete Einträge sind aus die PkOrg.ch Webseite genommen)

Kandidat	
Name	Hunter James Wilson

Beschreibung	Führt die IPA aus.
Phone	+41 78 401 38 63
Email	hunterinswitzerland@gmail.com

Tabelle 1: Projektaufbauorganisation Kandidat

Verantwortliche Fachkraft	
Name	Marco Daniele
Beschreibung	Unterstützt die Ausführung der IPA, kann als Hilfeleistung arbeiten bei technischen Schwierigkeiten. Bewertet eventuell diese IPA
Phone	044 448 38 38
Email	Marco.daniele@wundermanthompson.com

Tabelle 2: Projektaufbauorganisation VF

Berufsbildner	
Name	Beda Riklin
Beschreibung	Unterstützt die Vorbereitung auf den IPA
Phone	044 266 57 57
Email	Beda.riklin@ksh.ch

Tabelle 3: Projektaufbauorganisation Berufsbildner

Hauptexpert	
Name	Janes Thomas
Beschreibung	Führt die Expertbesuche aus. Bewertet eventuell diese IPA
Phone	079 295 82 88
Email	janes@janesthomas.ch

Tabelle 4: Projektaufbauorganisation Hauptexpert

Nebenexpert	
Name	Valentin Marolf
Beschreibung	Bewertet eventuell diese IPA
Phone	+41 79 936 27 80
Email	v.marolf@rafisa.ch

Tabelle 5: Projektaufbauorganisation Nebenexpert

Betrieb (Durchführungsort)	
Name	Wunderman Thompson
Adresse	Hardturmstrasse 133 8005 Zürich
Phone	044 448 38 38
Email	Info.ch@wundermanthompson.com

Tabelle 6: Projektaufbauorganisation Betrieb

3 Vorbereitungsarbeit

Kandidat hat folgendes vor der IPA gemacht:

- Hat ein Worddokument-Template für die Dokumentation erstellt.
- Hat ein Exceldokument-Template für den Zeitplan erstellt.
- Hat eine GitHub-Repository für die Versionierung von der Dokumentation erstellt.
- Hat eine Kanban-Board mit den Kriterien aus dem Kriterienkatalog als Fortschritts-Tracker erstellt.
- Hat seine Arbeitsumgebung für die IPA aufbereitet.
- Hat auf developer.chrome.com/docs/extensions verschiedene Informationen über Extensions angeschaut.
- Hat die Wunderman Thompson Vault API und deren Schnittstellen angeschaut.

4 Firmenstandards

4.1 Versionsverwaltung

Nach Firmenstandards wird ein Versionsverwaltungstool verwendet, um die Arbeitsergebnisse und die Dokumentation der IPA zu speichern und zu versionieren. Zwei Repositories werden erstellt, eins für die IPA-Dokumentation (schon vor dem IPA-Start erstellt), und die andere für den Code selbst (wird während der IPA erstellt). (Sehe 5.1 Dokumentation Verwaltung)

4.2 Confluence Dokumentation

Die Dokumentation zur Installation und Verwendung des Endprodukts von der IPA wird nach Firmenstandards auf die internen Dokumentationstools Confluence geschrieben. Diese Dokumentation soll unter der Vault Page gespeichert werden. Sämtliche Informationen, die zu dem Dienst vom Endprodukt unterstützen, die aber ausserhalb des Endprodukts selbst liegen, werden auch beschrieben. Ein Link/Kopie dieses IPA-Berichts wird auch auf dieser Seite zur Verfügung gestellt.

4.3 Arbeitsumgebung

Der Kandidat arbeitet während der Ausführung der IPA am Arbeitsplatz beim Betriebsort. Zur Verfügung gestellt wird: eine Windows Arbeitslaptop, die die Flexibilität des Arbeitsorts erleichtert. Eine Widescreen Bildschirm, und ein Tisch. Für Mittagspausen hat es eine Kantine im Betrieb, und pro Arbeitswoche werden 2 Tage vom Homeoffice angeboten. Für die Ausführung der IPA hat der Kandidat entschieden, die ganze Zeit in Office zu bleiben, um die Fokusse auf der Arbeit zu verbessern.

5 Organisation der Arbeitsergebnisse

5.1 Dokumentation Verwaltung

Der IPA-Bericht und alle verwandten Dateien werden mit dem Versionsverwaltungstool Git versioniert, und die Repository wird auf der Webseite «GitHub» gespeichert.

Diese Dokumentation wird auf der öffentlichen «HW-IPA-Dokumentation» Repository gespeichert (https://github.com/Hunter-1/HW-IPA-Dokumentation). Um die täglichen Speicherungen der Dokumentation leichter durchsuchbar zu machen, wird am Ende des Tages ein spezielles Release erstellt, mit den bis jetzt ausgefüllten Dokumentationen darin.

Als Backup wird die Dokumentation auch auf dem Betriebs-OneDrive gespeichert; dieses Backup hat leider keine Versionierungsfähigkeit.

5.2 Code Verwaltung

Gleich wie die Dokumentation wird der Code dieses Projekts mit Git versioniert und auf GitHub gespeichert. Für den Workflow wird Gitflow verwendet, dabei wird für jedes Arbeitspaket eine eigenes Branch erstellt, bevor es am Masterbranch gemergt wird.

Diese Dokumentation wird auf der öffentlichen «Vault-password-extension» Repository gespeichert (https://github.com/Hunter-1/vault-password-extension).

6 Projektmanagement

6.1 Anforderungen

Die Auswahl meiner Projektmanagementmethode hat zwei grosse Anforderungen, eins, die an diesem Projekt ausrichtet, die andere, die an den Kandidaten ausrichtet. Die Methode soll für eher kleine Projekte geeignet sein, da dieses Projekt allein ausgeführt wird und nur einen Zeitumfang von 10 Tage hat; und es soll genügend strukturiert sein, weil der Kandidat sich damit gemütlicher fühlt. Was für eine Projektmanagementmethode kann diese zwei Anforderungen erfüllen?

6.2 Gewählte Projektmanagementmethode

6.2.1 Wasserfallmethode

Die Wasserfallmethode wird für das Projekt gewählt. Diese Arbeitsmethode hat eine lange und bekannte Geschichte in der Arbeitswelt von Softwareentwicklern. Heutzutage wird es oft durch andere neue flexible Arbeitsmethoden ersetzt, aber für dieses Projekt soll es gut geeignet sein.

Diese Methode ist in 5 Phasen gegliedert:

1. Analysis-Phase
2. Design-Phase
3. Implementation-Phase
4. Testing-Phase
5. Maintenance-Phase

Diese Methode ist für dieses Projekt gut geeignet, für mehrere Gründe. Die Phasen kommen eins voreinander vor, sie haben eine sehr starke und klare Gliederung der Arbeit, und jede Phase hat eine klare Aufgabe und Zweck. Die Flexibilität von anderen Projektmanagementmethoden ist für dieses Projekt nicht geeignet, weil die Arbeitsanforderungen aus der IPA alle statisch sind.

Für dieses Projekt wird die letzte Phase nicht verwendet, weil Maintenance dieses Projekt ausserhalb des zehn Tage Umfangs dieses Projekt liegt. Daher wird diese Phase durch eine Reflexions-Phase ersetzt.

Diese alternative Methode neu gegliedert:

1. Analysis-Phase
2. Design-Phase
3. Implementation-Phase
4. Testing-Phase
5. Reflexions-Phase

6.3 Alternative Projektmanagementmethoden

6.3.1 Mini-Projectmanagement (mPM)

Mini Projectmanagement ist besonders gut für kleine Projekte geeignet, weil es in zwei sehr offene Arbeitsphasen unterteilt ist. Die Planungsphase und die Realisierungsphase. Diese Methode ist nicht gewählt worden, weil diese zwei Phasen zu offen für den Kandidaten waren; er möchte etwas mit mehr Struktur.

6.3.2 IPERKA

IPERKA ist eine oft vorkommende Projektmanagementmethode für Softwareentwicklung. Wie in dem Namen dieser Methode angedeutet, ist IPERKA in 6 Phasen unterteilt:

1. Informieren
2. Planen
3. Entscheiden
4. Realisieren
5. Kontrollieren
6. Auswerten

IPERKA und die ausgewählte Wasserfallmethode haben eine sehr ähnliches Phasen Struktur; diese Methode wird aber nicht ausgewählt, weil der Kandidat ein unklares Verständnis von einer Phase hat, nämlich die Entscheiden-Phase.

7 Arbeitspakete

A	**Umfeld und Ablauf**	
A.1	Allgemeine Inhalt	Allgemeine Information wird im Teil 1: Umfeld und Ablauf geschrieben. Inklusiv: Die Aufgabestellung aus PkOrg, Projektaufbauorganisation, Firmenstandards, Organisation der Arbeitsergebnisse, Projektmanagement
A.2	Zeitplan	Die Arbeitspakete werden definiert und der Zeitplan wird gefüllt.

Tabelle 7: Arbeitspakete Vorbereitung

1	**Analysis**	
1.1	Anforderungs-analyse	Die Anforderungen aus PkOrg werden angeschaut und eine genaue Liste von Anforderungen wird definiert.
2	**Design**	
2.1	Ablaufdiagramm	Zeigt den Ablauf einer normalem Usecase
2.2	Strukturdiagramm	Zeigt die Struktur vom Programm. Die verschiedenen Teile und ihre Verantwortungen werden auch beschrieben.
2.3	Integrationskonzept	Integrationskonzept ans existierende Backend
3	**Implementation**	
3.1	Funktion 1: Einträge speichern	Die Passworteinträge werden vom Vault API aufgerufen und im Model gespeichert.
3.2	Funktion 2: Einträge anzeigen	Die Passworteinträge werden in der Extension sichtbar sein. Die Passwörter können vom Extension copy and pasted werden.
3.3	Funktion 3: Aussehen	Das Aussehen der Extension wird entworfen, aufgezeichnet und implementiert.
4	**Testing**	
4.1	Testausführung	Das Testkonzept wird durchgeführt.
5	**Reflexion**	
5.1	Evaluation	Wie nah an die Planung ist das Programm implementiert. Sind alle Anforderungen erfüllt, wenn nicht, warum?
5.2	Kurzfassung	Die Kurzfassung am Anfang vom Teil 2 wird geschrieben.

Tabelle 8: Arbeitspakete Hauptarbeit

B	**Allfällige Dokumentation**	
B.1	Dokumentation	Während des Projekts soll die Implementation gleichzeitig dokumentiert werden.
B.2	Journal	Am Schluss jedes Tages soll in dem Projekt Journal einen Eintrag geschrieben.
B.3	Expert-Visit	Der Hauptexpert besucht am Tag 2 (03.05.24) und am Tag 9 (16.05.24). Diese Besuche werden richtig protokolliert.

Tabelle 9: Arbeitspakete Dokumentation

8 Zeitplan

(Der Zeitplan wird innerhalb einer Excel-Datei erstellt und als Screenshot im Bericht dargestellt.)

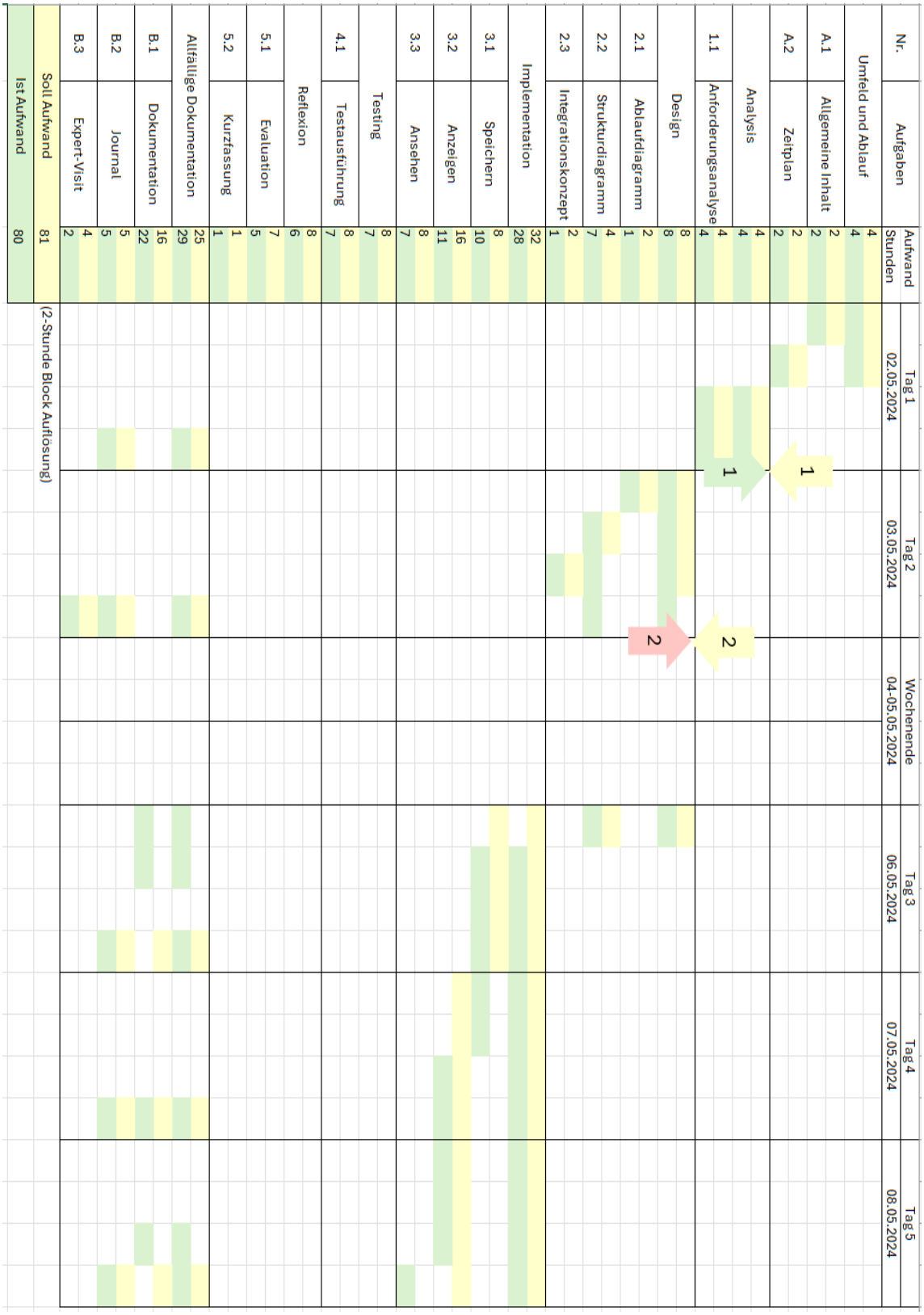

Abbildung 1: Zeitplan A

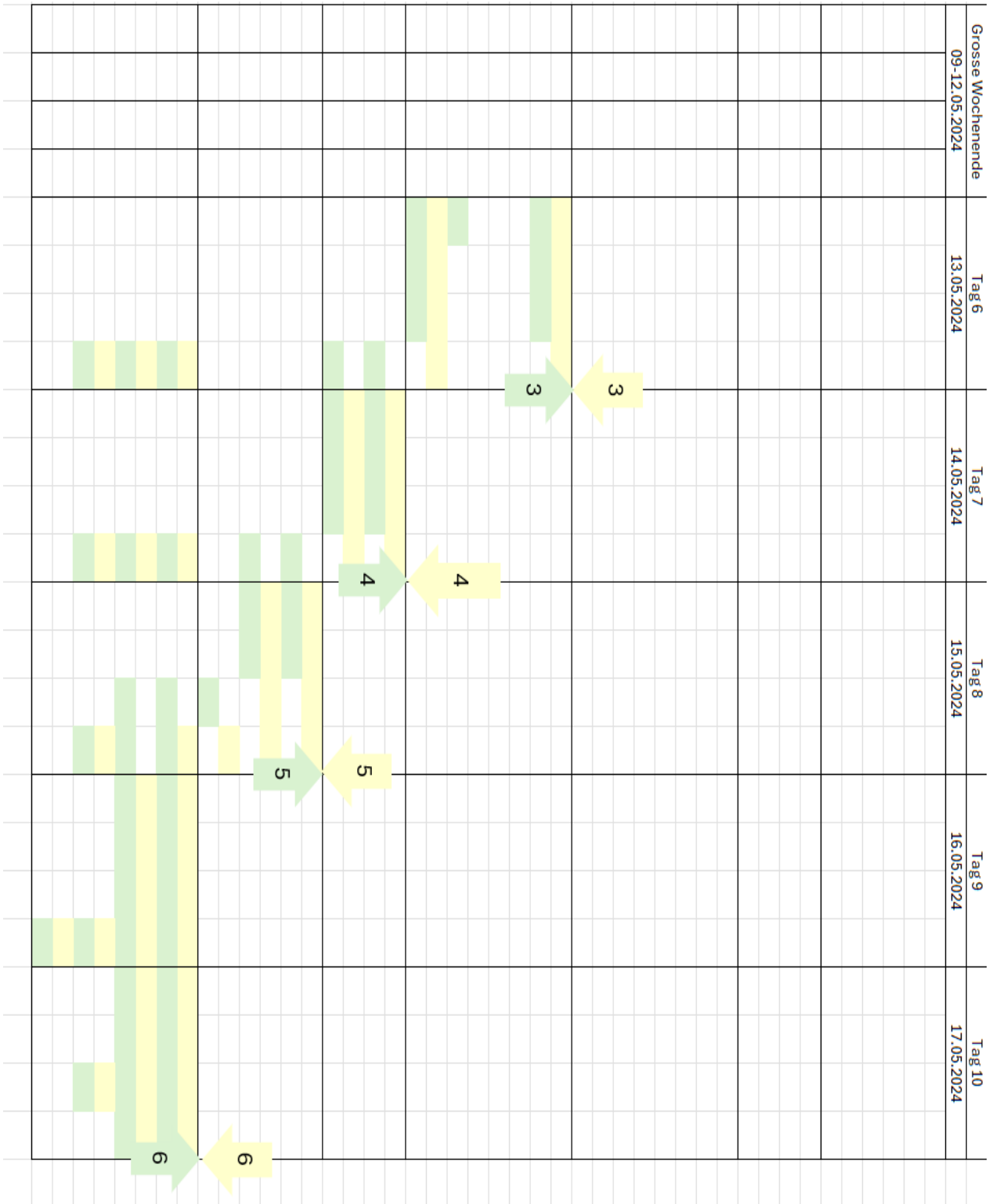

Abbildung 2: Zeitplan B

9 Arbeitsjournal

9.1 Tag 1

Tag 1	02.05.24
Geplante Arbeitspakete	A.1, A.2, 1.1
Erledigte Arbeitspakete	A.1, A.2, 1.1
Aufgetretene Probleme	-
Hilfsmittel	Als Hilfsmittel habe ich alte IPA-Berichte angeschaut und gelesen, um eine gute Gliederung der Texte zu erreichen.
Vergleich mit Zeitplan	Es gibt keine grossen Abweichungen vom Zeitplan.
Reflexion	Heute ist alles gut gegangen, morgen beginne ich mit der richtigen Planung.

Tabelle 10: Arbeitsjournal Tag 1

9.2 Tag 2

Tag 2	03.05.24
Geplante Arbeitspakete	2.1, 2.2, Testkonzept
Erledigte Arbeitspakete	2.1, 2.2, 2.3 (Integrationskonzept)
Aufgetretene Probleme	Obwohl ich heute das Testkonzept machen wollte, ohne konkretes Code Beispiel, war das nicht möglich. Stattdessen habe ich auf das Integrationskonzept gearbeitet, weil ich beim Erstellen des Zeitplans diesen wichtigen Teil vergessen habe. Ein anderes aufgetauchtes Problem ist, dass ich Schwierigkeiten bei der Design-Phase habe. Ich wusste nicht, was genau ich in dieser Phase schreiben möchte.
Hilfsmittel	Ich habe mit dem Hauptexpert über meine Schwierigkeiten mit der Design-Phase Dokumentation besprochen. Eine mögliche Lösung dafür ist, mit dem MVC-Pattern jedem Teil einige Verantwortungen und Funktionen zu geben.
Vergleich mit Zeitplan	Das Testkonzept im Zeitplan ist mit dem Integrationskonzept ersetzt. Bei der Implementation-Phase habe ich drei Aufgaben aufgeschrieben und im Zeitplan aufgeteilt. Der Meilenstein für die Design-Phase ist leider nicht erreicht worden und muss daher nächste Woche fertig sein.
Reflexion	Heute hat es mehrere Abweichungen und Schwierigkeiten gegeben, aber mithilfe des Hauptexperten haben wir meinen Fokus verbessert und nächste Woche soll ich diese Dokumentation fertig machen.

Tabelle 11: Arbeitsjournal Tag 2

9.3 Tag 3

Tag 3	06.05.24
Geplante Arbeitspakete	2.2, 3.1
Erledigte Arbeitspakete	2.2, 3.1 Ich habe auch in der Dokumentation über Authentication geschrieben.
Aufgetretene Probleme	Wegen des späten Erledigens vom Design-Phase, die extra Dokumentation über Authentication, kleine Schwierigkeiten, während das Setup vom Projekt, andere Schwierigkeiten mit Typescript in React, und eine eher grössere Schwierigkeit mit

	async initialisieren von den API-Call-Daten, wird die Speicherfunktion sicher länger brauchen als gedacht.
Hilfsmittel	Genauer in dem Quellenverzeichnis aufgeschrieben: ein Tutorial für eine Chrome-Extension mit React und Informationen über Integrated Windows Authentication.
Vergleich mit Zeitplan	Die Entwicklung vom Speichern der Daten wird länger dauern als gedacht. Die Entwicklung von den Anzeigen der Daten soll weniger Zeit brauchen.
Reflexion	Ich habe Schwächen mit Typescript und ich habe vergessen dafür zu planen; andererseits bin ich sicher, dass die View zu erstellen wird weniger Zeit als gedacht brauchen, weil ich mehr Erfahrung mit dem Erstellen von HTML-Komponenten als Datenspeicherung habe. Während der Zeitplanungsphase sollte ich daran mehr denken.

Tabelle 12: Arbeitsjournal Tag 3

9.4 Tag 4

Tag 4	07.05.24
Geplante Arbeitspakete	3.1, 3.2
Erledigte Arbeitspakete	3.1, 3.2
Aufgetretene Probleme	Es gab keine grossen Verzögerungen während der Implementation, das Speichern und das Anzeigen, aber nichts zu Grossen. Ich hatte aber Probleme mit Langweile.
Hilfsmittel	Genauer in dem Quellenverzeichnis aufgeschrieben: Ein Artikel über Async Constructors in Javascript und wie man Props zu einer React-Komponente gibt. Ich habe auch eine Idee über React Context bekommen, nachdem ich an andere Firmenprojekte für Inspiration geschaut habe.
Vergleich mit Zeitplan	Die fertige Implementation der Speicherung ist in kurzer Zeit fertig geworden; das Anzeigen macht allmählich Fortschritt. Ich habe heute weniger dokumentiert als geplant, aber ich denke, es soll keine Probleme verursachen.
Reflexion	Ich hatte heute weniger Energie als die vorherigen Tage; das eigentliche Programmieren ist mir mehr Routine als Dokumentation. Hoffentlich werde ich mehr Energie nach dem grossen Wochenende bekommen.

Tabelle 13: Arbeitsjournal Tag 4

9.5 Tag 5

Tag 5	08.05.24
Geplante Arbeitspakete	3.2, 3.3 (Aussehen entwerfen)
Erledigte Arbeitspakete	3.2, Copy and Paste, 3.3
Aufgetretene Probleme	Ein Funktional Requirement war unklar geschrieben, diese wurde umgeschrieben. Das Copy-and-paste-Funktionalität ist heute zufällig implementiert, da es sehr wenig zum Entwickeln gebraucht hat; wird das Arbeitspaket 3.3 (Copy and Paste) mit Arbeitspaket 3.3 (Aussehen entwerfen) ersetzt. Heute war es ein Umzugstag für die Firma, viel Möbel ist umgezogen worden und ich müsste innerhalb eines separaten Büroraums arbeiten. Dies hat mein Arbeitstempo verlangsamt.

Hilfsmittel	Genauer in dem Quellenverzeichnis aufgeschrieben: ein Artikel über das React Context API. Wie sortiert man ein Array von Javascript-Objekten mit leeren Strings.
Vergleich mit Zeitplan	Das Anzeigen zu entwickeln hat eher weniger Zeit gebraucht, wie gestern vorgesehen ist.
Reflexion	Obwohl heute ein bisschen hektisch und laut war, bin ich trotzdem vor dem Zeitplan. Ich bin sehr bequem mit meinem aktuellen Fortschritt, und ich werde mein 4-Tage-Wochenende geniessen.

Tabelle 14: Arbeitsjournal Tag 5

9.6 Tag 6

Tag 6	13.05.24
Geplante Arbeitspakete	3.3
Erledigte Arbeitspakete	Verbesserungen zum API-Call Fehlerbehandlung, 3.3, Testkonzept planen.
Aufgetretene Probleme	-
Hilfsmittel	Ich habe alte Testkonzepte von alten IPA-Berichten angeschaut.
Vergleich mit Zeitplan	Ich müsste eine Stunde verwenden, um das Error Handling vom Controller zu verbessern, aber die Arbeit am Aussehen der Extensions ist schneller als geplant gegangen, und ich konnte ein bisschen für das Testkonzept planen.
Reflexion	Heute ist gut gegangen, es gab keine grossen Abweichungen oder Probleme, und ich bin vor dem Zeitplan.

Tabelle 15: Arbeitsjournal Tag 6

9.7 Tag 7

Tag 7	14.05.24
Geplante Arbeitspakete	4.1
Erledigte Arbeitspakete	4.1, Anfang von der Reflexion
Aufgetretene Probleme	-
Hilfsmittel	Ich habe die Dokumentation für das Javascript-Testing Framework Jest und Mock Server angelesen.
Vergleich mit Zeitplan	Ich bin immer noch ein bisschen vor dem Zeitplan.
Reflexion	Heute ist auch gut gegangen, ich bin aber ein paar Sorgen über die Dokumentation, ich habe fast gar nichts in der Implementation-Dokumentation geschrieben, und ich bin keine schnellen Schreiber. Hoffentlich gehen die Evaluation und die Dokumentation schnell, jetzt, wo ich nichts mehr programmieren muss.

Tabelle 16: Arbeitsjournal Tag 7

9.8 Tag 8

Tag 8	15.05.24
Geplante Arbeitspakete	5.1, 5.2
Erledigte Arbeitspakete	5.1, 5.2, Confluence Dokumentation
Aufgetretene Probleme	-
Hilfsmittel	Ich habe alte IPA-Berichte angeschaut, um anzuschauen, was für Themen in der Reflexion vorkommen.
Vergleich mit Zeitplan	Ich bin immer noch vor dem Zeitplan.

Reflexion	Die Dokumentation für die Implementation habe ich noch nicht begonnen; das muss ich morgen fertig schreiben, besonders wegen des zweiten Expertenbesuchs. Es macht mich nervös, aber wenn es mehr Zeit als gedacht braucht, hat es immer noch Tag 10 zur Verfügung.

Tabelle 17: Arbeitsjournal Tag 8

9.9 Tag 9

Tag 9	16.05.24
Geplante Arbeitspakete	B.1, B.3
Erledigte Arbeitspakete	B.1, B.3
Aufgetretene Probleme	Bei der Dokumentation von der View hatte ich einige Schwierigkeiten, weil ich nicht genau wusste, was ich darüberschreiben möchte.
Hilfsmittel	-
Vergleich mit Zeitplan	Es gibt keine erkennbaren Abweichungen vom Zeitplan.
Reflexion	Heute ist ein bisschen langsam gewesen. Ich bin nach 2 Wochen ein bisschen müde geworden. Die Dokumentation für die Implementation ist fertig geschrieben (ausser der View, sie braucht ein bisschen mehr Arbeit) und der Hauptteil des Berichts ist jetzt komplett. Bei dem Treffen mit dem Hauptexperten hat er einige Tipps gegeben, was ich am letzten Tag daran arbeiten und verbessern kann.

Tabelle 18: Arbeitsjournal Tag 9

9.10 Tag 10

Tag 10	17.05.24
Geplante Arbeitspakete	B.1
Erledigte Arbeitspakete	B.1, hat mit dem Grammatiktool «LanguageTool» (https://languagetool.org/de) Grammatikfehler gefunden und korrigiert
Aufgetretene Probleme	-
Hilfsmittel	-
Vergleich mit Zeitplan	Heute kommen wir zum Ende des Projekts, das Zeitplan ist komplett.
Reflexion	Heute bin ich mit dem Projekt fertig geworden. Ich habe heute die Dokumentation durch eine Rechtschreibprüfung durchgegangen und geprüft, und es hat ein bisschen länger als gedacht gedauert. Ich habe auch einige fehlende Tabellen Captions gefunden und zugefügt. Ich bin müde, ich werde meine 3 Tage Wochenende geniessen.

Tabelle 19: Arbeitsjournal Tag 10

Teil 2: Projekt

10 Kurzfassung des Berichts

10.1 Ausgangssituation

Um die verschiedenen Logins für Kundenprojekte zu speichern, hat Wunderman Thompson einen Passwortspeicherungsservice mit dem Namen «Vault» erstellt. Vault speichert den Username und Passwort unter einem beschreibenden Titel für eine spezifische URL, und man kann das Login mit anderen Mitarbeitern von Wunderman Thompson durch ein kräftiges Berechtigungssystem. Logins können auch favorisiert werden, um das Verwenden günstiger zu machen. Die Funktionalität dieses Service wird mit diesem Projekt erweitert durch die Entwicklung einer Browser-Extension für Google Chrome und Microsoft Edge. Diese Extension soll die Liste von favorisierten Passwörtern aus der Vault Webseite durch ihre API nehmen. Die Passwörter und Usernames sollen zum Clipboard kopierbar sein.

10.2 Umsetzung

Dieses Projekt wird gemäss eine teilweise geänderte Wasserfallmethode ausgeführt.

- Die Anforderungen vom PkOrg werden angeschaut und genaue Requirements werden während der Analyse-Phase festgelegt.
- Während der Design-Phase wird die Architektur und der allgemeine Ablauf der Entwicklung geplant. Für dieses Projekt wird auch die Integration zur schon existierenden Vault API durch die verwendeten API-Calls dokumentiert.
- Die Implementation-Phase beschreibt die eigentliche Entwicklung und Programmierung vom Projekt. Für die Umsetzung wird eine Typescript React Projekt erstellt und durch Webpack wird das Projekt als Extension gebaut.
- In der Testing-Phase wird das Projekt getestet. Für dieses Projekt werden 3 Testarten geplant: Unittests, Integrationstests und Systemtests.
- Und am Schluss im Reflexion-Phase werden mögliche Verbesserungen und generelle Gedanken zum Projekt erläutert.

10.3 Ergebnis

Das Ergebnis ist eine Extension, die die Favoritenpasswörter vom Vault API aufruft und in der Extension anzeigt. Es erfüllt die angeforderten Grundfunktionen und berücksichtigt die Sicherheit, die Passwörter. Diese Extension wird für die Mitarbeiter von Wunderman Thompson zur Verfügung gestellt und erleichtert das Leben von Leuten, die oft Vault Passwörter brauchen.

11 Analyse

11.1 Requirement Analysis

Diese Extension soll die Arbeiter bei Wunderman Thompson die Arbeit erleichtern durch das schnelle Aufrufen der gespeicherten Passwörter. Um diese Aufgabe gut zu erfüllen, soll es diese Requirements erfüllen.

11.1.1 Functional Requirements

- Das Extension soll eine Liste von allen Passwörtern anbieten, die in Vault als «Favoriten» markiert ist. (Diese Passwörter müssen vorher auf die Vault.wundermanthompson.com Webseite manuell markiert sein).
- Diese Liste soll nach URL oder Name ordnen lassen, diese Ordnung soll wechselbar sein.
- Es soll möglich sein, einen erweiterten Passworteintrag in der Extension anzuzeigen, der mehr Information als die Listenansicht hat.
- Aus diesem Eintrag soll es möglich sein, das Password und das Username an den Clipboard zu kopieren (Copy and Paste Fähigkeiten)
- Der Aufruf von Passwörtern soll die Berechtigungen des Benutzers und seiner Benutzerrollen beachten. Unberechtigte Passwörter sollen nicht angezeigt werden.

11.1.2 Non-Functional Requirements

- Die Extension soll auf Microsoft Edge (ab Version 110) und auf Chrome (ab Version 109) lauffähig sein.
- Die Security der Extensions soll durch Windows Authentication verstärkt werden.
- Es soll eine umfassende Fehlerbehandlungsmethodik in der Extension eingesetzt.
- Da diese Extension für technisch unbewusste Leute ist, soll die Extension so Benutzer freundlich wie möglich sein.

12 Design

12.1 Architektur

Die Extension wird nach dem MVC-Pattern entworfen und gebaut, daher wird die Extension in 3 Teile organisiert.

- Das Model: verantwortlich für die Speicherung der Daten.
- Die View: verantwortlich für das Anzeigen von den Daten aus dem Model und das Interpretieren von Daten und Signalen vom User.
- Der Controller: verantwortlich für die Verbindung zwischen das Model, die View und irgendwelche anderen verwandten Systeme.

Auch ein Teil dieser Architektur ist die Vault API, woher die Passwörter stammen. Diese API ist Teil vom Arbeitsintranet und funktioniert nur bei einer Verbindung damit. Ein einfaches Diagramm von der geplanten Architektur wird hier angezeigt:

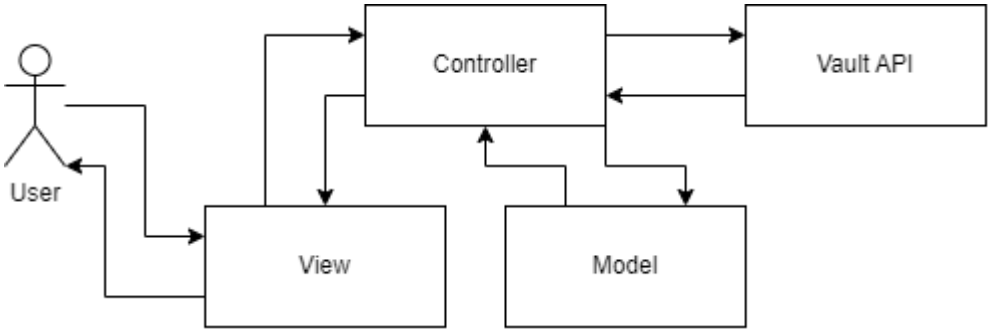

Abbildung 3: Architekturdiagramm

Da eine Extension entwickelt wird, werden die verschiedenen Websprachen verwendet für die Programmierung (Html, Css, Javascript). Zusätzlich wird für dieses Projekt React angewendet.

Jedes Teil des MVC-Patterns wird für bestimmte Funktionen des Ablaufs verantwortlich sein. Die Funktionen sind Folgendes:

- Model:
  - Erhaltet die rohen Daten vom Vault API. Diese Daten werden innerhalb der Model gelesen und für die View besser formatiert. Die formatierten Daten werden das lokal gespeichert und zu der View durch den Controller geschickt.
- View:
  - Erhaltet die formatierten Daten vom Model durch den Controller. Zeigt die Daten zum User in einer für Menschen lesbaren Weise. Passwörter und Usernames können vom gezeigten Passworteinträge durch einen Knopfdruck zu Clipboard kopiert werden. Erhaltet Inputs vom User und schickt diese Inputs zum Controller.
- Controller:
  - Kontrolliert die Kommunikation zwischen die View, das Model und der Vault API. Schickt API Calls zur Vault API, interpretiert die Inputs des Users.

12.2 Ablauf

Um das Verständnis zum Ablauf dieses Programms zu verbessern, wird ein Ablaufdiagramm dafür erstellt:

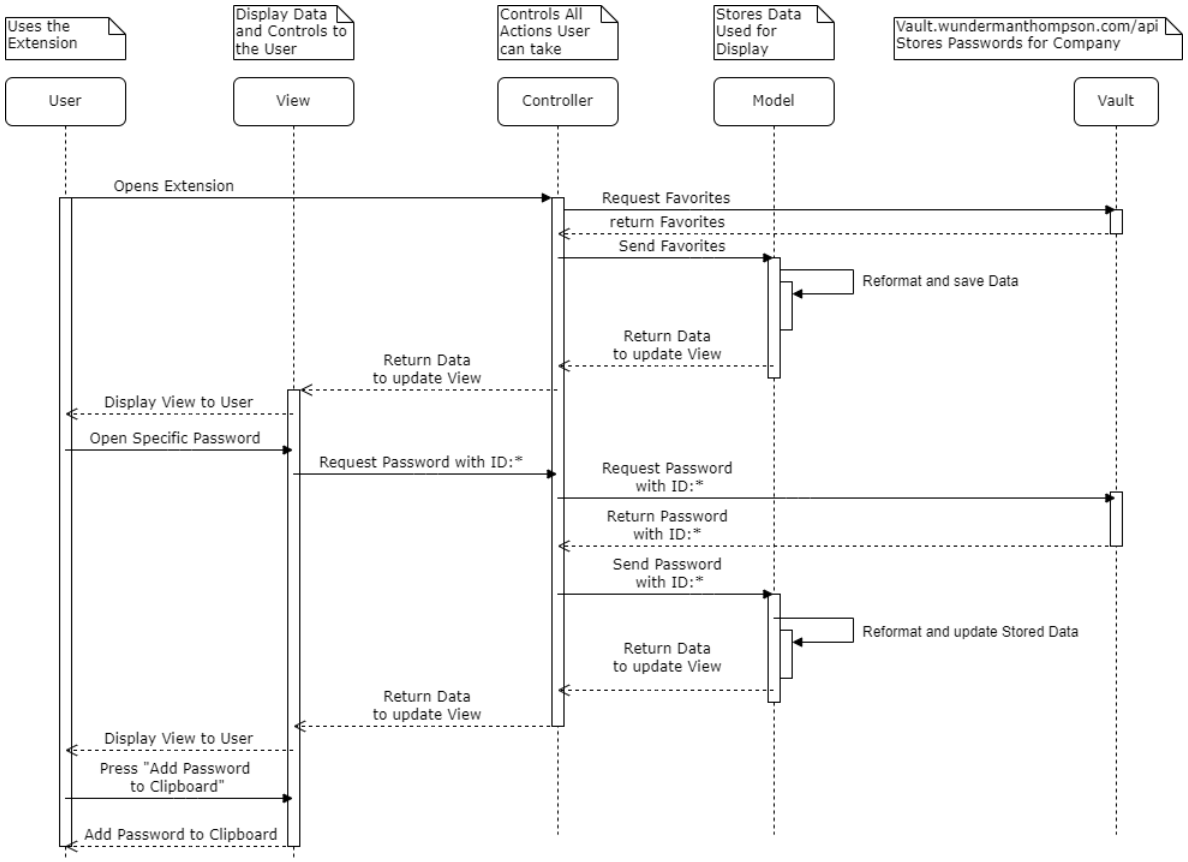

Abbildung 4: Ablaufdiagramm

Die wichtigsten Aktivitäten im Ablauf sind das Folgende:

- Die Extension wird geöffnet und eine initiale Request für die Favoriten wird geschickt. Inklusiv ist auch die Authentication vom User. Die Response hat nur die Liste von Einträgen und keine Passwörter selbst.
- Die Response wird zum Model geschickt und die Response wird neu formatiert und gespeichert. Jedes Mal, dass die Erweiterung gestartet wird, müssen die Favoriten neu gespeichert werden, um immer so aktuell wie möglich zu bleiben.
- Die gespeicherten Passworteinträge werden zu der View geschickt und zum User angezeigt.
- Nachdem das View einen Input vom User bekommt, einen Passworteintrag detailliert anzuzeigen, wird dieser Input zum Controller geschickt.
- Der Controller schickt eine neue Request zur Vault API und erhält den Passworteintrag mit dem Passwort.
- Dieser Eintrag wird zum schon existierenden Model integriert und gespeichert.
- Das neue Modell wird zu der View geschickt und die aktualisierte View wird zum User gezeigt.
- Wenn der User ein Passwort kopieren möchte, kann er das direkt im View machen.

12.3 Integrationskonzept

Da die Arbeit mit Passwörtern grosse Sorgfalt erfordert, sollen API-Calls, die Passwörter enthalten können, stark kontrolliert werden. Die verwendeten API-Calls, sowie der Zweck und der genaue Parameter, sollen dokumentiert werden.

API Call	https://vault.wundermanthompson.ch/api/favorites ?Page={pageNumber}&ClientID=
Verwendungs-zweck	Diese API-Call enthält die Liste alle favorisierten Passworteinträge des Users
Parameter	• pageNumber: Nur 20 Passworteinträge werden auf einmal geschickt, wenn es mehr als 20 Passworteinträge im Favoritenliste gibt, werden sie in mehrere Pages geteilt. Nimmt eine Zahl als Input.
Verwendete Response Parameter: • ItemsCount: Zeigt Anzahl Passworteinträge. • CurrentPage: Zeigt die aktuelle Seite. • UrlPreviousPage, UrlNextPage: Wird verwendet, um auf andere Seiten zu bewegen. (Diese Parameter werden im Konzept geplant, aber in der Implementation ist aufgedeckt worden, dass die Pagezahl nur einen Eintrag inkrementiert, anstatt einer ganzen Seite; die Parameter werden deshalb entfernt. • FoundItems: Hat die Liste von Passworteinträgen, die für das Anzeigen verwendet wird. (Diese Liste enthaltet die Passwörter selbst nicht.) Verwendete Parameter aus die Passworteinträge: • Id: Identifiziert der Passworteintrag. • Title: Name der Passworteintrag; wird für das Ordnen der Einträge verwendet. • Url: Url, wo das Passwort verwendet wird; wird für das Ordnen der Einträge verwendet. • ReadPermissions: Zeigt, ob den User dieses Passwort anschauen darf. • RequestUrl: Der API-Call für den Passworteintrag selbst, im Gegensatz zur Favoritenliste hat dieser API-Call das Passwort.	

Tabelle 20: API Call 1

API Call	https://vault.wundermanthompson.ch/api/credentials/ {Id}
Verwendungs-zweck	Diese API Call enthält die Liste alle favorisierten Passworteinträge des Users
Parameter	• Id: Die URL braucht das Id für den Passworteintrag; diesen Parameter muss man selbst nicht ausfüllen, weil die ganze RequestUrl in den vorherigen API-Call gespeichert wird.
Verwendete Response Parameter: • Comment: Beschreibung der Passworteintrag. • Username: Wird zur Verfügung in der Extension gestellt. • Password: Wird zur Verfügung in der Extension gestellt.	

Tabelle 21: API Call 2

13 Implementation

13.1 Fehlerbehandlung

Um Fehler im ganzen Code konsistent zu vermeiden, wird für dieses Projekt Typescript verwendet.

Typescript ist eine Programmiersprache, die auf Javascript basiert. Es ist eine Obermenge von Javascript durch die Hinzufügung von mehreren Features wie statische Typisierung und erweiterte Fehlererkennung. Dies erlaubt man, Syntaxfehler früher und zu erwischen.

Typescript wird für alle selbst geschriebenen Teile des Projekts verwendet.

13.2 Projekt Setup

Um eine Chrome / Edge Extension mit React zu bauen, muss man das Projekt speziell aufbauen. Hier sind die genommenen Schritte zum Erstellen des Projekts:(Eine online Tutorial wird für dieses Teil verwendet.) (1)

- Eine Create React App Befehl ausführen:
  `npx create-react-app vault-password-extension --template typescript`
- Webpack wird für diese Projekt installiert:
  `npm install --save-dev webpack webpack-cli copy-webpack-plugin css-loader ts-loader html-webpack-plugin ts-node`
- Eine Webpack Config File wird erstellt, um eine Build für das Extension zu verwenden. Diese Config File wird vom Tutorial kopiert.
- Zwei Scripts werden werden zum package.json file zugefügt:
  ```
  "build": "webpack --config webpack.config.js",
  "watch": "webpack -w --config webpack.config.js"
  ```
- Eine Manifest File wird erstellt, um das Code als Chrome Extension ausführbar zu machen:
  ```
  {
      "version": "1.0.0",
      "manifest_version": 3,
      "name": "Vault Password Extension",
      "description": "This Extension will display Passwords from the Wunderman Thompson Vault Service",
      "action": {
          "default_popup": "js/index.html",
          "default_title": "Vault Password Extension"
      }
  }
  ```

Um dieses Projekt zu testen, habe ich einen Build gebaut und auf Chrome getestet. Leider hat dieses Tutorial für mich nicht funktioniert und ich müsste ein bisschen Troubleshooting machen. Eventuell habe ich das Problem durch die folgenden Veränderungen gelöst.

- Im App.tsx habe ich das SVG Referenz und das SVG File selbst gelöscht.
- App.css habe ich gelöscht.
- Im index.tsx file habe ich das Code mit dem folgendes ersetzt:
  ```
  const root = document.createElement("div");
  root.className = "container";
  document.body.appendChild(root);
  const rootDiv = ReactDOM.createRoot(root);
  rootDiv.render(
    <React.StrictMode>
      <App />
    </React.StrictMode>
  );
  ```

Am Schluss vom Projekt Setup wird folgendes im Chrome Extension angezeigt. Der aufgetauchte Text zeigt, dass alles jetzt funktioniert und für die echtee Programmierung bereit ist. Eine Repository wird für diesen Code auf GitHub erstellt.

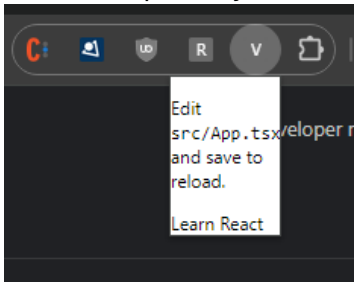

Abbildung 5: Extension im Browser nach Projektsetup

13.3 Authentication

Um Zugriff auf die Vault API zu erhalten, muss man zum Arbeitsintranet verbunden sein, und man muss sich zuerst authentisieren. Für diese Aufgaben gibt es zwei Vorgehensweisen dafür.

13.3.1 Manual Authentication

Normalerweise bei einer Request mit einer Authentifizierungsanforderung gibt es eine Login Popup für das vault.wundermanthompson.com Domain. Der User muss diesen Popup mit seinem Username und Passwort ausfüllen; dieser Login wird gecacht und der User muss sich nicht mehr für eine gewisse Zeitlänge einloggen. Diese Weise ist das Defaultverhalten vom Browser und von der Webseite.

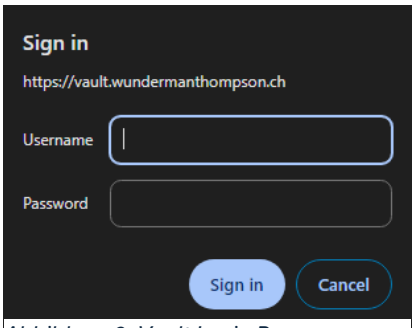

Abbildung 6: Vault Login Popup

13.3.2 Automatic Authentication

Weil diese Extension auf Windows entwickelt wird, kann ich das Windows-Feature: Integrated Windows Authentication verwenden. Dieses Feature erlaubt, man auf die Vault Webseite automatisch einzuloggen. Das automatische Login ist nur möglich für zwei Gründe, die Vault Webseite unterstützt Kerberos und NTLM Login, und das Login für die Webseite ist das gleiche Login wie das Windows Login vom User.

Um Integrated Windows Authentication zu aktivieren, muss man die folgenden Schritte folgen: (2)

- In das Windows Control Panel > Network and Internet > Internet Options > Advanced Tab > Enable Integrated Windows Authentication soll aktiviert sein.
- Im gleichen Ortc aber im Security tab > Local Intranet Sites > Advanced soll die Vault Webseite (https://vault.wundermanthompson.ch) auf dieser Liste hinzugefügt.
- Jetzt immer noch im Security tab > Local Intranet Site > Custom level > User Authentication soll das Logon setting auf entweder Automatic logon only in Intranet zone, oder Automatic logon with current username and password eingestellt.

Da diese Einstellungen tief in den Windows Settings sind und daher nicht möglich sind, wegen Sicherheitsbedenken, mit der Extension sie zu verändern, wird diese Authentifizierungsweise nicht gewählt werden. Möchte aber Wunderman Thompson die Einstellungen firmenweit verändern möchte, werden diese Protokolle in dem IPA-Bericht und auf Confluence dokumentiert werden.

13.4 Model

Für die Speicherung der Daten ist die «Model» Klasse verantwortlich. Die Daten selbst werden durch diese zwei Klassen gespeichert:

- PasswordEntries: speichert die Anzahl von total Passwörter§n, und die aktuelle Seite von Passwörtern.
- «passwordEntry»: speichert die Passworteintrag ID, die Name, die URL, wo das Passwort verwendet wird, ob man das Passwort lesen kann, und die URL, um das Passwort und Username aufzurufen, da die initiale Liste Aufruf die Passwörter selbst nicht enthält. Nachdem man den erweiterten Passworteintrag erhalten hat, wird der Username, das Passwort und irgendwelche Kommentare gespeichert. Um zu zeigen, ob dieser Eintrag das Passwort und Username hat, wird dies in einem Credentials-Boolean gespeichert.

```
You, last week | 1 author (You)
export class passwordEntries {
  itemsCount: number = 0;
  currentPage: number = 0;
  entries: Array<passwordEntry> = new Array();
}
```

Abbildung 7: PasswordEntries Klasse

```
You, last week | 1 author (You)
export class passwordEntry {
  id: string = "";
  title: string = "";
  url: string = "";
  readPermissions: boolean = false;
  requestUrl: string = "";
  credentials: boolean = false;
  comment: string = "";
  username: string = "";
  password: string = "";
}
```

Abbildung 8: PasswordEntry Klasse

Bei der Initialisierung der Model Klasse werden die erhaltenen Daten in einem PasswordEntries und in einem Array von PasswordEntry gespeichert. Das Model kann das ItemCount und das CurrentPage und spezifische Einträge aufrufen. Die Einträge können mit der richtigen ID und den Daten aktualisiert werden; und die Einträge können nach die Eintrag Name oder die Eintrag URL organisiert aufgerufen werden. Beim Sortieren von URLs gab es ein unerwartetes Verhalten, die leeren URLs werden am Anfang organisiert, anstatt am Schluss. Um dies zu lösen habe ich auf dem Internet gesucht (3). Bei einer leeren URL wird der Eintrag am Schluss organisiert.

13.4.1 Speichern Lösungsvariante

Zum Speichern von den Daten gab es zwei Möglichkeiten.

```
compareURLs(a: passwordEntry, b: passwordEntry) {
  if (a.url == "") {
    return 1;
  } else if (b.url == "") {
    return -1;
  } else {
    return a.url.localeCompare(b.url);
  }
```

Abbildung 9: CompareURLS Funktion

- Local Storage: Die Daten werden im Browser gespeichert. Local Storage wird nicht für dieses Projekt nicht gewählt, weil die Daten im Local Storage öffentlich zu anderen Programmen sind. Sensitive Daten müssen gespeichert werden, und die Daten müssen bei jeden API-Call neu gespeichert werden, um aktuell zu bleiben. Local Storage wird beim Browserabschalten nicht gelöscht.
- Variablen: Variablen werden im RAM gespeichert, dies ist für uns besser, denn die Daten beim Browser und Extension abschalten gelöscht werden. Nur die Extension hat auf diese Daten Zugriff.

13.5 Controller

```
class Controller {
  model: Model;
  ok: boolean = true;
  constructor(model: Model) {
    this.model = model;
  }
  static async init() {
    const [status, data] = await callFavorites(0);
    if (status == 200) {
      return new Controller(new Model(data));
    } else {
      return status;
    }
  }
```

Abbildung 10: Controller Constructor

Für die Kommunikation zwischen der View, dem Model, und die Vault API ist der Controller verantwortlich. Bei der Initialisierung wird eine erste API Call zur Vault API geschickt. Leider kann man Constructors nicht async machen, um herauszufinden wie man eine Constructor async auszuführen habe, ich eine Internet-Artikel gelesen (4). Um den API-Call auszuführen, muss man nur eine statische async Funktion innerhalb des Constructor auszuführen

Bei einem erfolgreichen API-Call wird die Statuszahl und die Json Datei zurückgeschickt. Bei einem gescheiterten API-Call wird die Zahl -1 zurückgeschickt.

Die Favoriten-Liste wird durch die callFavorites Funktion, was das callAPI verwendet, aufgerufen. Der Controller kann auch detaillierte Passworteinträge aufrufen und die gewünschte Passworteintragseite verändern. Diese Informationen werden zum Model geschickt und gespeichert.

```
You, 3 days ago | 1 author (You)
async function callApi(url: string) {
  try {
    const response = await fetch(url);
    //console.log(response);
    var data;
    if (response.status == 200) {
      data = await response.json();
    }
    //console.log(data);
    return [response.status, data];
  } catch {
    return [-1, null];
  }
}
```

Abbildung 12: CallApi Funktion

```
async changePage(changeAmount: number) {
  const currentPage = this.model.getCurrentPage();
  if (!this.checkIfTurnable(changeAmount)) {
    return;
  } else {
    const [status, data] = await callFavorites(currentPage + changeAmount);
    if (status == 200) {
      this.model = new Model(data);
    }
    this.setOk(status);
    return status;
  }
}
```

Abbildung 11: ChangePage Funktion

13.6 View

Die View zeigt die Controls vom Controller und die Daten vom Model. Bei der Initialisierung der Extension wird der Controller erstellt, bei der erfolgreichen Erstellung vom Controller wird es zu der View als Prop angegeben (5), und die View zum User angezeigt. Bei einem gescheiterten Controller, einer Fehleranmeldung.

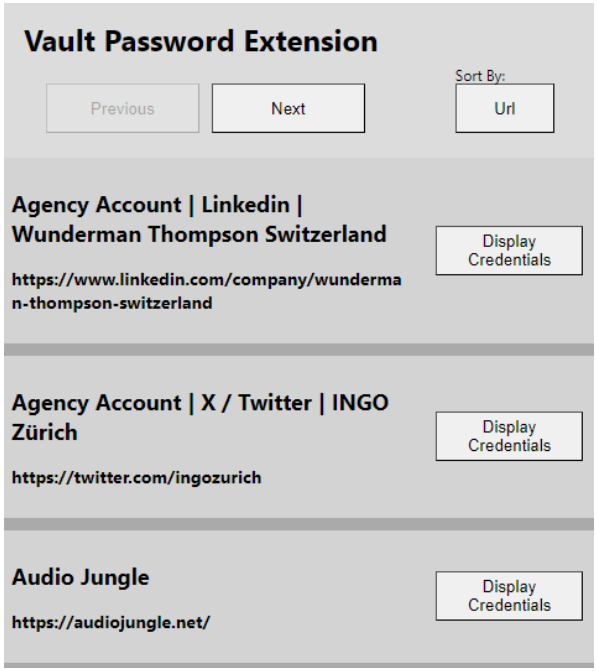

Abbildung 13: View GUI

Die Passworteinträge werden innerhalb einer Liste angezeigt (Link dargestellt). Die Liste kann mit dem Knopf oben rechts nach ihrem Namen oder URL sortiert werden. Beim Drucken vom «Display Credentials» Button, wird eine Api-Call aufgerufen, um die Credentials zu holen.

Um das API-Call beim Öffnen von Passworteintrag aufzurufen, musste ich der Controller als Prop eingeben. Ich dachte bei der zukünftigen Entwicklung von der Extension wäre es mühsam der Controller immer als Prop einzugeben, und ich suchte nach einer anderen Lösung. Dies habe ich im Code einer Kundenprojekts von Wunderman Thompson gefunden, und später richtig recherchiert (6).

Man kann den Controller zu jeder Child-Komponente geben durch einen Context Provider. Dies funktioniert, ohne eine explizite Prop abzugeben.

```
{controller ? (
  <ViewProvider controller={controller}>
    <View controller={controller} />
  </ViewProvider>
```

Abbildung 14: View mit View Provider

Bei einem Eintrag ohne richtige Berechtigungen wird stattdessen der Text im Knopf «Permission denied» gezeigt und es wird deaktiviert sein.

Auf dieser Sicht kann man den Username und Passwort anschauen und zum Clipboard kopieren.

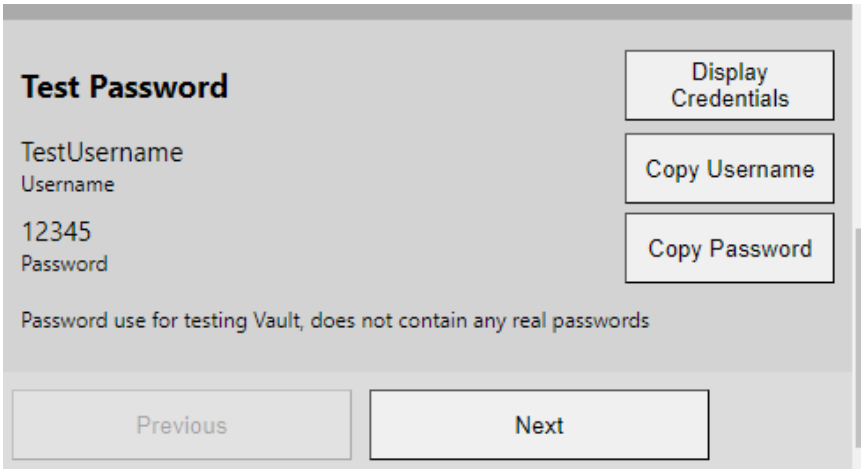

Abbildung 15: Passworteintrag mit Credentials GUI

14 Testing

14.1 Testkonzept

Für diese Projekt werden 3 Arten von Tests verwendet.

- **Unittests**: Getestet werden kleine und individuelle Teile vom Code; diese Teile sind auf andere Teile des Codes nicht abhängig und können sehr einfach und oft automatisiert getestet werden. Beispiel Daten vom Vault API werden vorher gespeichert und können mehrmals verwendet werden. Um dies zu zeigen, werden alle Unittests im Projekt geschrieben und automatisch ausgeführt. Das Javascript-Testing Framework «Jest» wird verwendet. (6) Getestet werden die Funktionen des Models.
- **Integrationtests**: Getestet wird die Integration zur API. Ähnlich zu den Unittests werden Beispiel-Daten vom Vault API vorher gespeichert, aber anstatt der direkten Daten müssen API-Calls verwendet. Für die gemocketen API wird die Extension «Mock Server» verwendet. Die API-Calls werden für diese Tests zum MockServer API verändert anstatt der echten Vault API. Für diese Tests wird keine Authentication getestet. Getestet werden die Funktionen des Controllers.
- **Systemtests**: Getestet wird die Funktionalität des Extensions im Chrome und Edge Browser und die Integration zur echten Vault API. Getestet wird die eigentliche Extension, durch das View. Da 2 Browsers getestet werden müssen, wird jeder Test zweimal durchgeführt. Weil es unwahrscheinlich ist, dass es grosse Unterschiede zwischen Chrome und Edge geben werden, werden die Tests, die gleich auf Chrome und Edge geführt werden, innerhalb der gleichen Test-Cases zusammengelegt. Unterschiede zwischen den Browser werden als A: für Chrome und B: für Edge markiert. Diese Tests werden manuell auf den Versionen 124 für beide Browsers durchgeführt. Obwohl es in der eigentlichen Requirements-analyse steht, dass die Extension auf die Versionen 110 für Chrome und 109 für Edge funktionieren soll, werden die Versionen nicht getestet, wegen der Schwierigkeiten mit der Installation von den alten Versionen des Browsers.

14.1.1 Testmittel

Jeder Test wird auf meinem Arbeitslaptop (zrh133-lap-133) und zum Arbeitsintranet (dc.wundermanthompson.ch) verbunden. Bewusst nicht getestet wird das Verwenden von der Extension ohne Verbindung zum Arbeitsintranet, weil Leute, die nicht bei Wunderman Thompson arbeiten, keinen Zugriff auf den Passwortservice haben sollen.

14.2 Testdurchführung

14.2.1 Unittests

Unittest 1: Construct Model	
Testperson	Hunter Wilson
Testzeitpunkt	14.05.2024, 10:16
Beschreibung	Das Model Class wird instanziiert, eine Json Datei «data1.json» wird als Parameter gegeben. Data1.json hat nur einen Passworteintrag. Das Model Class soll richtig instanziiert worden sein.
Testumgebung	Visual Studio Code: Jest Framework
Ausgangszustand	Die data1.json Datei wird vorher aufbereitet.
Testdurchführung	Ein Model wird instanziiert, eine «data1.json» Datei wird als Parameter gegeben. Um die Existenz des Models zu prüfen, wird das Item Count geprüft.
Erwartete Resultat	Das Item Count soll gleich «1» sein.
Tatsächliche Resultat	Das Item Count ist gleich 1.
Kommentar	Der Test ist wie erwartet gegangen.

Status	Erfolgreich

Tabelle 22: Unittest 1

Unittest 2: Test updateEntry	
Testperson	Hunter Wilson
Testzeitpunkt	14.05.2024, 10:16
Beschreibung	Das Model wird instanziiert, zwei Json Dateien «data1.json» und «password1.json». Password1.json enthält den detaillierten Passworteintrag für den Eintrag im Data1.json. Das Model wird mit Password1.json aktualisiert.
Testumgebung	Visual Studio Code: Jest Framework
Ausgangszustand	Die data1.json und das password1.json Datei wird vorher aufbereitet.
Testdurchführung	Ein Model wird instanziiert, die «data1.json» Datei wird als Parameter gegeben. Die updateEntry Methode wird mit der richtigen ID und der Password1.json Datei gegeben. Zu prüfen wird das Passwort dieses Eintrags.
Erwartete Resultat	Das Passwort soll gleich «12345» sein.
Tatsächliche Resultat	Das Passwort ist gleich 12345.
Kommentar	Der Test ist wie erwartet gegangen.
Status	Erfolgreich

Tabelle 23: Unittest 2

Unittest 3: Test getEntriesByName	
Testperson	Hunter Wilson
Testzeitpunkt	14.05.2024, 10:16
Beschreibung	Das Model Class wird instanziiert, eine Json Datei «data2.json» wird als Parameter gegeben. Diese Datei hat 5 Passworteinträge. Die Einträge werden nach ihrem Namen organisiert und abgegeben.
Testumgebung	Visual Studio Code: Jest Framework
Ausgangszustand	Die data2.json Datei wird vorher aufbereitet.
Testdurchführung	Ein Model wird instanziiert, eine «data2.json» Datei wird als Parameter gegeben. Die getEntriesByName Methode wird nachgefragt. Der erste Eintrag soll die spezifische ID 5 haben, weil dieser Eintrag mit A beginnt.
Erwartete Resultat	Der erste Passworteintrag soll die ID 5 haben.
Tatsächliche Resultat	Der erste Passworteintrag hat die ID 5.
Kommentar	Der Test ist wie erwartet gegangen.
Status	Erfolgreich

Tabelle 24: Unittest 3

Unittest 4: Test getEntriesByUrl	
Testperson	Hunter Wilson
Testzeitpunkt	14.05.2024, 10:16
Beschreibung	Das Model Class wird instanziiert, eine Json Datei «data2.json» wird als Parameter gegeben. Diese Datei hat 5 Passworteinträge. Die Einträge werden nach ihrer URL organisiert und abgegeben.
Testumgebung	Visual Studio Code: Jest Framework
Ausgangszustand	Die data2.json Datei wird vorher aufbereitet.
Testdurchführung	Ein Model wird instanziiert, eine «data2.json» Datei wird als Parameter gegeben. Die getEntriesByURL Methode wird

	nachgefragt. Der erste Eintrag soll die spezifische ID 3 haben, weil dieser Eintrags-URL mit A beginnt.
Erwartete Resultat	Der erste Passworteintrag soll die ID 3 haben.
Tatsächliche Resultat	Der erste Passworteintrag hat die ID 3.
Kommentar	Der Test ist wie erwartet gegangen.
Status	Erfolgreich

Tabelle 25: Unittest 4

14.2.2 Integrationstests

Integrationstests 1: Construct Controller	
Testperson	Hunter Wilson
Testzeitpunkt	14.05.2024, 13: 24
Beschreibung	Der Controller wird initialisiert. Mit einem API-Call auf eine Mock: Localhost:3000/favorites{pageNumber} (pageNumber = 0). Der Controller soll richtig erstellt sein.
Testumgebung	Visual Studio Code: Jest Framework API-Call von Mock Localhost Server
Ausgangszustand	Das Mock Server wird vorher aufbereitet.
Testdurchführung	Der Controller wird instanziiert. Seine Existenz wird durch den Item Count vom Model überprüft.
Erwartete Resultat	Das Model soll eine Item Count von 30 haben.
Tatsächliche Resultat	Das Item Count ist 30
Kommentar	Der Test ist wie erwartet gegangen.
Status	Erfolgreich

Tabelle 26: Integrationstest 1

Integrationstests 2: Test callPasswordEntry	
Testperson	Hunter Wilson
Testzeitpunkt	14.05.2024, 13: 24
Beschreibung	Der Controller wird initialisiert. Die callPasswordEntry Methode mit dem Passworteintrag ID wird aufgerufen. Der Controller macht einen API-Call dafür und aktualisiert das Model damit.
Testumgebung	Visual Studio Code: Jest Framework API-Call von Mock Localhost Server
Ausgangszustand	Das Mock Server wird vorher aufbereitet.
Testdurchführung	Der Controller wird instanziiert. Der Passworteintrag wird aufgerufen. Das Passwort dieses Eintrags wird geprüft.
Erwartete Resultat	Das Passwort soll gleich «12345» sein.
Tatsächliche Resultat	Das Passwort ist «12345».
Kommentar	Der Test ist wie erwartet gegangen.
Status	Erfolgreich

Tabelle 27: Integrationstest 2

Integrationstests 3: Test changePage	
Testperson	Hunter Wilson
Testzeitpunkt	14.05.2024, 13: 24
Beschreibung	Der Controller wird initialisiert. Die nächste Seite soll aufgerufen werden. Diese neue Liste von Passwörtern soll richtig gespeichert werden.
Testumgebung	Visual Studio Code: Jest Framework API Call von Mock Localhost Server

Ausgangszustand	Das Mock Server wird vorher aufbereitet.
Testdurchführung	Der Controller wird instanziiert. Die nächste Seite wird aufgerufen. Geprüft werden das CurrentPage und der Name des ersten Eintrags dieser neuen Seite.
Erwartete Resultat	Das Models CurrentPage soll 20 sein, und der erste Eintrags Name soll «A Test Password» sein.
Tatsächliche Resultat	Das CurrentPage ist 20 und der Name ist «A Test Password».
Kommentar	Der Test ist wie erwartet gegangen.
Status	Erfolgreich

Tabelle 28: Integrationstest 3

14.2.3 Systemtest

Systemtest 1: Standartanwendung	
Testperson	Hunter Wilson
Testzeitpunkt	14.05.2024, A: 14:34 B: 14:38
Beschreibung	Die Extension wird normal verwendet, um ein Passwort zu finden.
Testumgebung	A: Google Chrome Version 124 B: Microsoft Edge Version 124
Ausgangszustand	Windows Integrated Authentication ist deaktiviert. Mehrere Passwörter werden am Vault.wundermanthompson.ch vorher favorisiert.
Testdurchführung	Die Extension wird aktiviert. Der nächste Seite-Button wird gedruckt, es wird zum Passworteintrag gescrollt und das Display Credentials Button geklickt, dann wird das Copy Password Button gedruckt und das Passwort soll jetzt im Clipboard sein.
Erwartete Resultat	Das Passwort soll im Clipboard sein.
Tatsächliche Resultat	Das Passwort ist im Clipboard.
Kommentar	Der Test ist wie erwartet gegangen, die erste Anwendung dieser Extension braucht mehrere Sekunden, um mit der API zu authentisieren, zu können; nachdem sind alle API-Calls viel schneller gegangen. Es gab keine Unterschiede zwischen den Browsern
Status	A: Erfolgreich, B: Erfolgreich

Tabelle 29: Systemtest 1

Systemtest 2: Keine Internetverbindung vor Extension Initialisierung	
Testperson	Hunter Wilson
Testzeitpunkt	14.05.2024, A: 14:51 B: 14:52
Beschreibung	Die Extension wird ohne Internet aktiviert.
Testumgebung	A: Google Chrome Version 124 B: Microsoft Edge Version 124
Ausgangszustand	Windows Integrated Authentication ist deaktiviert. Das Internet für den Computer vor dem Test ausgeschaltet.
Testdurchführung	Die Extension wird aktiviert.
Erwartete Resultat	Eine Error-Nachricht soll dargestellt werden.
Tatsächliche Resultat	Eine Error-Nachricht ist dargestellt worden.
Kommentar	Der Test ist wie erwartet gegangen. Es gab keine Unterschiede zwischen den Browsern.
Status	A: Erfolgreich, B: Erfolgreich

Tabelle 30: Systemtest 2

Systemtest 3: Keine Internetverbindung nach Extension Initialisierung	
Testperson	Hunter Wilson
Testzeitpunkt	14.05.2024, A: 14:55 B: 14:57
Beschreibung	Die Extension verliert Internet, nach der initialen Aktivierung.
Testumgebung	A: Google Chrome Version 124 B: Microsoft Edge Version 124
Ausgangszustand	Windows Integrated Authentication ist deaktiviert.
Testdurchführung	Die Extension wird aktiviert. Das Internet geht verloren; nach einem Klick auf entweder die Seiten-Buttons oder die Credentials-Buttons soll eine Error-Nachricht dargestellt werden.
Erwartete Resultat	Eine Error-Nachricht soll dargestellt werden.
Tatsächliche Resultat	Keine Error-Nachricht ist gezeigt worden.
Kommentar	Der Test war leider nicht erfolgreich. Das Problem liegt wahrscheinlich, mit dem useEffect Function,
Status	A: Gescheitert, B: Gescheitert

Tabelle 31: Systemtest 3

Systemtest 4: Zugriff auf Credentials ohne richtige Berechtigungen	
Testperson	Hunter Wilson
Testzeitpunkt	14.05.2024, A: 15:03 B: 15:04
Beschreibung	Ein Passworteintrag ohne die richtigen User-Berechtigungen soll die Credentials nicht anzeigen (Username und Passwort)
Testumgebung	A: Google Chrome Version 124 B: Microsoft Edge Version 124
Ausgangszustand	Windows Integrated Authentication ist deaktiviert. Ein Arbeitskolleg hat ein Passwort erstellt, ohne mir die Berechtigungen zu geben. Dieses Passwort ist vor die Testausführung favorisiert (nicht erfolgreich, sehe tatsächliche Resultat).
Testdurchführung	Die Extension wird aktiviert und der Passworteintrag wird in der Listensicht nachgesucht.
Erwartete Resultat	Die Passwortcredentials können nicht angezeigt werden, weil der Button deaktiviert ist.
Tatsächliche Resultat	Der Passworteintrag konnte an der ersten Stelle nicht favorisiert werden, weil es nicht zu mir angezeigt worden ist.
Kommentar	Während der Entwicklung dieser Extension habe ich mehrere Male versucht, Passwörter zu finden, die ich nicht berechtigt bin, es anzuschauen. Ich dachte, ich konnte mindestens den Eintrag sehen, aber das ist nach diesem Test öffentlich falsch gewesen. Obwohl die Sicherheit dieser Passwörter eingehalten worden ist, muss ich leider diese Tests als gescheitert markieren.
Status	A: Gescheitert, B: Gescheitert

Tabelle 32: Systemtest 4

15 Reflexion

15.1 Probleme

Mehrere Probleme sind während der Testdurchführung entdeckt worden, hier unten werden sie diskutiert.

15.1.1 Fehlende Error Message

(Sehe Systemtest 3)

Im Controller, wenn es ein Problem mit einem API-Call hat, wird so ein Objekt zurückgegeben: [-1, null]. Beim Erhalten ein Objekt mit dieser spezifischen Zahl wird eine OK Boolean im Controller geflippt und die View soll das Erkennen und eine Error Nachricht darstellen. Diese Nachricht wird durch eine UseEffect Funktion aktiviert, aber es hat nicht funktioniert.

Es ist mir nicht sehr klar, warum es nicht funktioniert hat. Dieses Problem kann mit der Async Funktionen liegen, weil Sie einige Zeit brauchen, richtig auszuführen, aber diese Erklärung ist mir nicht sicher.

15.1.2 Berechtigungen

(Sehe Systemtest 4)

Das Anzeigen von Passworteinträgen habe ich während meiner Recherche an die Vault Webseite falsch verstanden. Ich habe versucht, einen Eintrag zu finden, ohne die Berechtigungen zu haben, es anzuschauen, und keine davon habe ich gefunden.

Damals dachte ich, es war nur wegen schlechten Sicherheitsmassnahmen, weil die meisten Einträge mit «Everyone can read» markiert waren. Den richtigen Grund dafür habe ich nur während der Testdurchführung verstanden, nachdem ich ein Arbeitskolleg ein neues Passwort zu erstellen gebeten habe, ohne mir die Anschauberechtigungen zu geben. Dieses Passwort habe ich in der Webseite nicht gesehen, und daher nicht favorisieren können. So war es auch für eine Menge von anderen Passwörtern auf der Vault Webseite.

Dieses Missverständnis hat die Sicherheit der Passwörter nicht verletzt, es hat aber wertvolle Entwicklungszeit aufgebraucht, was für andere Teilen des Extensions nutzvoller wäre.

15.2 Verbesserungen

Wegen der kurzfristigen 10-Tage Durchführungszeit dieses Projekts konnten mehrere Features für die Extension nicht entwickelt werden. Solche möglichen Features werden unten diskutiert.

15.2.1 Props vs Context

React-Apps werden aus mehreren Komponenten aufgebaut. Komponenten können innerhalb von anderer Komponente platziert werden, zum Beispiel in der Extension wird mehrere «Entry» Komponente im «View» Komponente platziert; diese Entry Komponente werden als «Child» sowie die View wird als «Parent» bezeichnet. Child Komponente können normalerweise die Informationen aus dem Parent nicht zugreifen und um Informationen vom Controller und Model zu kriegen, muss das Parent der Controller als Prop abgegeben werden, ähnlich zu einem Funktion Parameter.

Für eine einfache React-App wie diese Extension reicht das. Aber bei einer grösseren App mit mehreren Komponenten, mit mehreren Ebenen von Parents und Children, können Props unhandlich werden. Dafür kann man Context verwenden. Contexts werden durch eine Provider-Komponente gegeben und können auf jeder Ebene der Child-Komponente verwendet werden. Eine Contextprovider habe ich in der Extension implementiert; ich denke aber, es könnte besser integriert sein, da es nur einmal verwendet wird.

15.2.2 Loading Display

Weil diese Extension für eine API aufgebaut wird, werden API-Calls oft aufgerufen. API-Calls brauchen immer ein bisschen Zeit auszuführen, und deswegen hat fast jeder Knopfdruck in der Extension eine kleine Verzögerung. Diese Verzögerung kann ich nicht herausnehmen oder herumarbeiten, weil man immer die am aktuellsten Dateien aus dem Vault API herausnehmen möchte. Man kann aber dem User zeigen, dass es ein bisschen Zeit braucht, durch einen Loading screen.

Die Extension hat bei der Initialisierung einen sehr einfachen Loading Screen mit einer einfachen Nachricht. Aber noch besser wäre eine spezialisierte Loading Komponente, die mehrmals verwendet werden kann.

15.2.3 Aussehen

Das Aussehen der Extension war manuell mit Css entworfen und erstellt. Ich habe sehr wenig Erfahrung als Websitedesigner und mein Extensions-Design ist meiner Meinung nach bland und langweilig. Eine Idee für ein besseres Design wäre, das Vault Webseite selbst zu emulieren. Auch wenn man keine anderen Webseiten kopieren möchte, gibt es auch die Möglichkeit, ein Css Framework wie Bootstrap und Tailwind Css zu verwenden. Eine Css Framework zu verwenden habe ich während der Planung daran gedacht, ich habe aber dagegen entschieden, weil ich keine Erfahrung mit Css Frameworks habe.

16 Schlusswort

Diese IPA war eine verwirrende Zeit für mich, nicht weil die Implementation für mich schwierig war, eigentlich war es das Gegenteil. Bei der PA-Planung am Ende 2023 dachte ich dieses Projekt war für mich genügend schwierig, wegen die starke Fokusse auf die Sicherheit den Daten, aber während die Vorbereitungsarbeiten dafür, habe ich gelernt, dass diese Sicherheitskonzerne schon im Vault API abgedeckt werden. Der Hauptexpert war am ersten Expertenbesuch auch mit dem niedrigen Schwierigkeitsgrad dieses Projekt einverstanden.

Die eigentliche Programmierung war nicht schwierig, aber deswegen hatte ich negative Gedanken. «Braucht die Entwicklung wirklich 4 ganze Tage, geht es nicht schneller?», «Das ist eigentlich nur Datenspeicherung, warum braucht es einen ganzen Tag?», «Brauchte ich ein einfaches Projekt? Bin ich für ein schwierigeres Projekt ungenügend?». Obwohl ich diese Gedanken hatte, haben sie mich nicht sehr stark beeinflusst. Alle Arbeit braucht Zeit, auch die einfache Arbeit. Und alle Arbeit bringt Wert damit.

Die Dokumentation war für mich schwieriger. Ich bin nicht ein sehr verboser Mensch, und ich finde es schwierig vieles zu schreiben. Ich denke aber diese Dokumentation ist gut gegangen.

Mit der Extension selbst bin ich okay. Es gab keine besonders schwierigen Funktionen zu implementieren, und es gibt klar mögliche Verbesserungen zum Code. Obwohl es eine kleine Hiccup am Anfang der IPA bei der Planung gab, war ich immer vor meinen Zeitplan. Und am Schluss habe ich eine funktionierende und komplette Chrome / Edge Extension (So lange während unseres Rebrandings der Name der Vault Webseite sich nicht verändert).

17 Confluence Dokumentation

Wie im Firmenstandards aufgeschrieben ist, wird eine einfach Dokumentation auf Confluence geschrieben. Unten ist eine Kopie der Dokumentation.

Description

This Browser Extension (Designed for Chrome and Edge) allows you to see your favorited Passwords inside of a Browser Extension for your convenience.

Github Repository:
https://github.com/Hunter-1/vault-password-extension

Dokumentation Repository:
https://github.com/Hunter-1/HW-IPA-Dokumentation

Installation

Vault Password Extension.zip

Download this zip file and extract the folder to the desired location.
Go to the Extensions page of your chosen browser and enable Developer Mode
Select "Load Unpacked" and select the extracted folder.

If done correctly, An Extension named "Vault Password Extension" should appear in the list of your extensions.

Enable Auto Login (Windows only)

To enable automatic Login into the Vault Website and Extension with your Windows Account:

Search for "Internet Options" in the windows searchbar.
Goto the Advanced Tab and make sure "Enable Integrated Windows Authentication" is activated.
Move the the Security Tab and select Local Intranet zone → custom level
Scroll to the bottom for User Authentication and check if either "Automatic logon only in Intranet zone" or "Automatic logon with current user name and password" are enabled.
Then move to: the Local Intranet zone → Sites → Advanced and add to the list:
https://vault.wundermanthompson.ch

If done correctly, you will be automatically logged in when using the Website or Extension

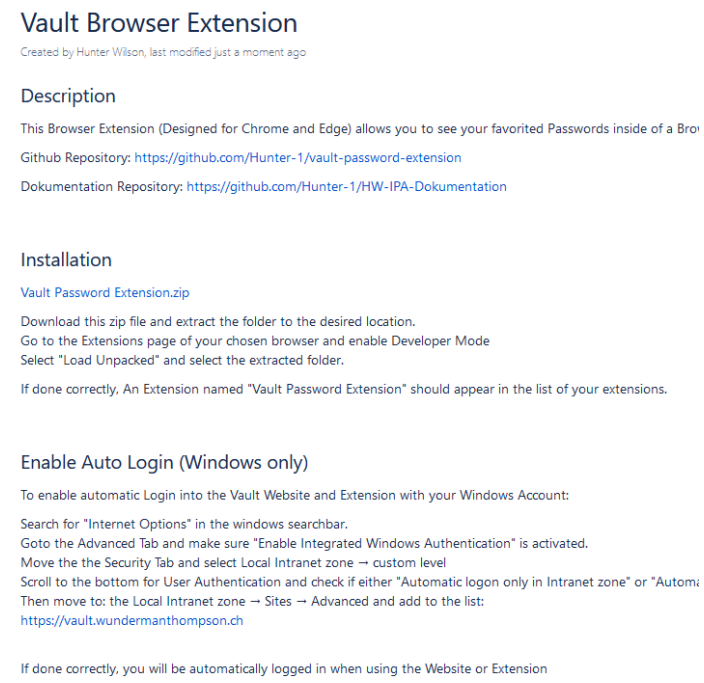

Abbildung 16: Confluence Dokumentation

18 Glossar

Begriff	Beschreibung
API	«Application Programming Interface»: Funktionen und Protokolle, die zur Kommunikation mit anderen Applikationen dienen.
Clipboard	Ein Ort im Computer, wo man schnell Text oder Bilder speichern können.
Confluence	Confluence ist ein Wiki-type Dokumentationstool für Firmen.
Extension	Eine Applikation, die die Funktionalität von Browser erweitern soll.
Integrated Windows Authentication	Integrated Windows Authentication erlaubt man automatisch auf Webseiten mit Kerberos oder NTLM Authentisierung mit dem Users Windows Login einzuloggen.
Intranet	Ein Intranet ist ein privates Netzwerk, wird oft bei Firmen verwendet.
MVC-Pattern	Das MVC-Pattern ist ein Muster für Softwareentwicklung, die das UI für eine Applikation in 3 Teile separiert. Das Model, der Controller, und die View.
React	React ist eine Javascript Library für die Entwicklung von UI-Komponente.
Webpack	Webpack ist ein Code Bundler, es reduziert die Menge von Assets, die zum Ausführen der Applikation erforderlich sind.

Tabelle 33: Glossar

19 Quellenverzeichnis

1. **Joshi, Harshita.** Creating a Chrome Extension with React: A Step-by-Step Guide. *medium.com*. [Online] 14. Februar 2023. [Zitat vom: 06. 05 2024.] https://medium.com/@tharshita13/creating-a-chrome-extension-with-react-a-step-by-step-guide-47fe9bab24a1.

2. **MicroStrategy.** Configure Web Browser for Integrated Authentication. *microstrategy.com*. [Online] [Zitat vom: 06. 05 2024.] https://www2.microstrategy.com/producthelp/Current/SystemAdmin/WebHelp/Lang_1033/Content/integrated_auth_web_browser.htm.

3. **Hotinger, Eric.** JS sort() empty to end. *Stackoverflow*. [Online] 8. 10 2015. [Zitat vom: 08. 05 2024.] https://stackoverflow.com/questions/33016087/js-sort-empty-to-end.

4. **Ortiz, Basti.** The Proper Way to Write Async Constructors in Javascript. *dev.to*. [Online] 18. 08 2021. [Zitat vom: 07. 05 2024.] https://dev.to/somedood/the-proper-way-to-write-async-constructors-in-javascript-1o8c.

5. **VanBuskirk, Adam.** How to pass props to a functional React component using typescript. *blog.wordbot.io*. [Online] 18. 06 2023. [Zitat vom: 07. 05 2024.] https://blog.wordbot.io/tech/how-to-pass-props-to-a-functional-react-component-using-typescript/.

6. **OpenJS Foundation.** *jestjs.io*. [Online] [Zitat vom: 14. 05 2024.] https://jestjs.io/.

7. **Boateng, Dickson.** How to Use the React Context API in Your Projects. *Freecodecamp.org*. [Online] 29. 03 2023. [Zitat vom: 08. 05 2024.] https://www.freecodecamp.org/news/context-api-in-react/.

20 Figurenverzeichnis

21 Tabellenverzeichnis