

InfluenzaConnect: Eine React-basierte Webanwendung für Influencer-Marketing

Technical Reports: CL-2024-42, März 2024

Sebastian Weidner, Jonas Hermann, Nils Bayerl, Dominik Schwagerl,

Timon Spichtinger, Christoph P. Neumann 

CyberLytics-Lab at the Department of Electrical Engineering, Media, and Computer Science

Ostbayerische Technische Hochschule Amberg-Weiden

Amberg, Germany

Zusammenfassung—*InfluenzaConnect ist eine Webanwendung, die das Ziel der strategische Vernetzung von Unternehmen und Influencern verfolgt. Dabei soll das Influencer Marketing von Unternehmen, als auch die selbstständige Vermarktung als Influencer erleichtert werden. Die Besonderheit von InfluenzaConnect ist u.a. die automatisierte Analyse des Instagram-Profiles des Influencers bei der Registrierung.*

Die Architektur basiert im Frontend auf dem React JavaScript-Framework, im Backend auf dem Python-Framework Flask und für die Persistenz wird die NoSQL Datenbank MongoDB verwendet. Frontend, Backend und die MongoDB wurden mittels Docker containerisiert, um eine hochskalierung der Webanwendung zu ermöglichen. Als CSS Framework wurde Tailwind CSS und für die Kommunikation zwischen Frontend und Backend die Axios, ein Promise-basierter HTTP-Client sowie die Fetch API verwendet. Für die Datenaquise wurde auf die Instagram-API und ? zurückgegriffen.

Index Terms—**Influencer-Marketing; WebApp; Business; SocialMedia.**

I. ÜBERBLICK

A. Motivation

Unternehmen wollen möglichst viele ihrer Produkte verkaufen. Dazu braucht es eine gute Qualität der Produkte, preiswerte Verkaufspreise und Markenbekanntheit. Letzteres erfordert ein ausgereiftes Marketing-Konzept. Ein Marketing-Konzept ist dabei das Influencer-Marketing. Beim Influencer-Marketing lassen Unternehmen ihre Produkte von Influencern bewerben, um den Bekanntheitsgrad der Firma, sowie den Bekanntheitsgrad und die Umsatzzahlen ihrer Produkte weiter zu erhöhen.

Was ist das Problem?

Unternehmen wollen authentische, sowie glaubwürdige Stimmen von Menschen finden, die ihr Produkt präsentieren und bewerben können. Die Lösung auf dieses Problem ist das Influencer-Marketing. Dieses ist aber noch nicht so weit verbreitet und Unternehmen, die sich für dieses Marketing-Konzept entschieden haben, stehen wieder vor neuen Problemen.

- 1) Wie finde ich den passenden Influencer für meine Produkte?
- 2) Es ist aufwendig mit mehreren Influencern zu verhandeln
→ Wie kann ich mit mehreren Influencern effizient verhandeln?

- 3) Welche Social-Media-Plattformen kommen für das Bewerben meiner Produkte infrage, bzw. wie finde ich die richtige Plattform für meine Produkte? Soll es Instagram, Facebook, TikTok, YouTube, Pinterest, X, LinkedIn oder doch ein privater Blog sein, um nur einen Einblick über die Vielfalt der Plattformen zu bieten.

Dazu kommen generelle Probleme wie:

- 4) Es gibt noch nicht viele Influencer die im B2B-Space tätig sind.
- 5) Es gibt viele unbekannte Influencer.
- 6) Influencer mit hohen Reichweiten sehen sich selbst gar nicht als Influencer an.
- 7) Influencer mit kleineren Reichweiten sind schwer auffindbar und haben es schwerer, Aufträge von Unternehmen zu bekommen.

Durch unsere Webbasierte Influencer-Marketing-Plattform wollen wir es schaffen, diese Probleme zu beheben und das Influencer-Marketing zu vereinfachen.

B. MVP

Das MVP umfasst folgende Anforderungen:

- 1) *Registrierungs- und Anmeldungsseite für Influencer*
- 2) *Automatische Analyse des Instagram-Profiles des Influencers*
- 3) *Tabellarische Übersicht aller registrierten Influencer, inkl. Sortierung, Suche und Filterungsfunktion*
- 4) *Datailübersicht zum Influencer mit allen relevanten Informationen*

C. Implementierungsstand

Vom MVP wurden folgende Punkte umgesetzt:

- 1) *Registrierungs- und Anmeldungsseite für Influencer*
- 2) *Automatische Analyse des Instagram-Profiles des Influencers während der Registrierung*
- 3) *Übersicht über alle registrierten Influencer, inkl. Suche und Spaltenfilter*

Die *Datailübersicht zum Influencer* wurde nicht implementiert. Stattdessen wurde für den *angemeldeten Influencer eine Profilseite implementiert, in der er seine aktuellen Profildaten einsehen und ändern kann.* Um alle

Daten zum Influencer trotz der fehlenden Dateiansicht für die Besucher der Website bereitzustellen, wurden diese vorerst in die Tabelle in der Übersichtsseite zu den Influencern integriert.

II. PROJEKTSETUP

III. ARCHITEKTUR

A. Containerisierung

Das Projekt wurde mittels Docker containerisiert, um eine flexible und skalierbare Umgebung für die verschiedenen Komponenten der Anwendung zu schaffen. Dabei wurden das Frontend, Backend sowie die MongoDB in separate Docker-Container aufgeteilt, was eine einfache Verwaltung und Skalierung der einzelnen Dienste ermöglicht. Dockerfiles legen dabei fest, welche Software und Abhängigkeiten in den virtuellen Containern installiert werden. Für das Backend wird ein Python 3.10 Slim Image verwendet. In dieses werden beim Starten des Containers die notwendigen Python-Pakete automatisiert installiert und der Anwendungscode kopiert. Das Frontend basiert auf einem Node.js 16 Image, das ebenfalls automatisch die benötigten Node-Module installiert und den Build-Prozess des Frontends startet (Kein multi stage).

B. Deployment

InfluenzaConnect verwendet GitHub als Git-Repository. Das Deployment in unsere Hetzner-Cloud erfolgt demnach mittels GitHub Actions. Der CD-Workflow befindet sich wird dabei ausgelöst, sobald Änderungen auf den `prod` Branch gepusht werden. Der anschließende Deployment-Job läuft auf einer Ubuntu-Maschine in der Hetzner-Cloud und umfasst folgende Schritte:

- Checkout Code: Der Quellcode wird aus dem Repository ausgecheckt.
- Setup SSH: SSH wird eingerichtet, um eine sichere Verbindung zum Hetzner-Server herzustellen.
- Deploy to Hetzner: Der Code wird auf den Hetzner-Server deployt.

Beim deployen wird dabei unser Projektverzeichnis aktualisiert, Docker Compose verwendet, um die bestehenden Container zu stoppen, neu zu bauen und schließlich die Anwendung in die Produktionsumgebung hochzufahren. Durch diese Pipeline wird sichergestellt, dass jede Änderung im `prod` Branch automatisch auf dem Produktionsserver deployt wird. Dadurch wird eine kontinuierliche Integration und Bereitstellung neuer Features und Bugfixes ermöglicht, Entwicklungszyklen beschleunigt und die Qualität der Anwendung verbessert.

C. Frontend-Architektur

D. Backend-Architektur

IV. TECHSTACK

A. Frontend

Für die Realisierung des Frontends wird das JavaScript-Framework React verwendet. Es erleichtert die Entwicklungsarbeit im Frontend, indem es dabei hilft, wiederverwendbaren Code zu schreiben, sowie graphische Inhalte dynamisch

aufzubauen und zu befüllen. HTML, CSS und JavaScript-Code wird dabei in sogenannten Komponenten gekapselt. In React gibt es sog. Hooks, die dazu benutzt werden, bestimmte Funktionalitäten in den Komponenten zu erreichen. Erwähnenswert ist hier der Hook `useState()`, der bestimmte Zustände zwischenspeichern kann, sodass diese beim erneuten Rendern der Komponente nicht verloren gehen. Desweiteren haben wir uns für die Gestaltung der Webpages für Tailwind CSS entschieden. Tailwind CSS wird oft mit React verwendet und ist zur Zeit sehr im Trend, weil es leicht zu erlernen ist, gutes vordefiniertes Design und trotzdem große Gestaltungsfreiräume bietet, da man notfalls interne CSS-Werte speziell für sein Projekt überschreiben kann. Außerdem haben wir beschlossen, anstatt JavaScript auf TypeScript zu setzen, da TypeScript eine Typisierung zulässt und somit logische Typfehler beseitigt.

TypeScript ist komplett kompatibel mit JavaScript. `Fetch!!`

B. Backend

Im Backend haben wir uns für das Python-Webframework Flask entschieden. Flask ermöglicht die Erstellung von Webanwendungen in Python durch eine minimalistische und flexible Struktur, sodass es sich gut für kleine bis mittelgroße Projekte eignet. Durch das Definieren von Routen, basierend auf den Anforderungen der Clients werden die HTTP-Anfragen verarbeitet und entsprechende Antworten zurückgeben. Zum persistieren der Daten setzen wir auf eine MongoDB als Datenbanksystem. Die `pymongo`-Bibliothek von Python fungiert dabei als offizieller Treiber für die MongoDB, die Funktionen für die Interaktion der Datenbank bereitstellt, um Daten zu speichern, abzurufen, aktualisieren und zu löschen.

V. IMPLEMENTIERUNG

A. Frontend

1) Landing Page

: Routing auch mit erwähnen

2) Registrierung

: Die Registrierung dient dazu, sich als Influencer registrieren und vermarkten zu können. Die Registrierung ist in drei Schritten aufgebaut: Zuerst muss er seine Anmeldeinformationen - eingeben. Im Zweiten Schritt werden seine Persönliche Daten abgefragt, die später teilweise für die Unternehmen veröffentlicht werden und im Dritten Schritt muss er seinen Instagram-Account hinterlegen, der anschließend im Backend analysiert wird.

Nachfolgend finden Sie eine Detaillierte Auslistung der Eingaben:

- Anmeldeinformationen - Email, Passwort
- Persönliche Daten - Anrede, Vorname, Nachname, Nationalität, Telefonnr., Gesprochene Sprachen, Beschreibung über sich selbst
- Verknüpfung Social-Media Accounts - Instagram Username

Jeder der drei Schritte wurde in einem Eigenständigen Formular realisiert, zwischen denen durch einen Weiter- und Zurück-Button hin- und hergewechselt werden kann.

Für die Eingabefelder wurden von Grund auf eigene Komponenten erstellt. Ein Eingabefeld besitzt immer ein Label und ein Ausgabefeld für Fehlermeldungen. Dabei taucht unter jedem Eingabefeld eine Fehlermeldung auf, falls die eingegebenen Daten nicht den Richtlinien entsprechen. Um das Datenhandling mit den Input-Felder zu vereinfachen und um Codezeilen zu sparen, wurde die Bibliothek `React Hook Form` im Frontend eingebunden. Durch diese kann man vereinfacht die Logik hinter den Formularen erstellen. Formulareingaben können so z.B. ganz einfach zwischenspeichert werden, um diese z.B. beim zurückgehen nicht zu verlieren. Auch das einbinden einer Datenvalidierung wird unterstützt. So werden die Eingaben mit der Bibliothek `yup` validiert, die gleichzeitig das Setzen eigener Fehlermeldungen, nach jedem Validierungsschritt erlaubt. Die Fehlermeldungen werden anschließend in ein internes Objekt, dass `React Hook Form` zu jedem registrierten Input-Element erstellt, geschrieben und durch unsere Komponente automatisch unter dem Input-Feldern angezeigt.

Wurden alle Eingaben getätigt werden diese gesammelt mittels `fetch()` API an das Backend gesendet und dort nochmals überprüft. Neben einer einfachen Eingabefeld, wurden Komponenten für ein Select-, ein Multiselectdropdown und für zwei Buttons zur Registrierung erstellt.

3) An- und Abmeldung

: - Wiederverwendung der Input-Komponenten - Popup-Dialog Komponente von XXX verwendet - Daten ans Backend gesendet - Response - Session gesetzt - Button Logout und Email erscheint - Logout -> Session gelöscht, Erscheinungsbild wie zuvor

4) Profildaten Bearbeiten

: - Anpassung der Input-Komponenten, Standardmäßig non-Editable - Daten werden anhand der Email, die in der Session gespeichert ist rausgeholt. - durch Edit-Button kann man die Eingabefelder bearbeiten. Gleichzeitig erscheint unten links ein Abbrechen- und ein Speichern-Button. Wird der letztere Button betätigt, werden die Daten im Frontend Validiert und anschließend ins Backend gesendet. Dort werden Sie nochmals auf validiert und schlussendlich persistiert, indem die alten mit den neuen Daten überschrieben werden.

5) Influencerübersicht

: Die Influencerübersicht dient dazu, dass Unternehmen einen passenden Influencer für sich finden können. Demnach sind in einer Tabelle alle Registrierten Influencer mit allen relevanten Daten, inkl. Informationen, die durch die Analyse des Instagram-Accounts gewonnen wurden zu sehen. Für die Datendarstellung in der Tabelle wurde in TypeScript eine Datenstruktur erstellt, auf die die Daten vom Backend gamappt werden.

Ein **Spaltenfilter** ermöglicht es, Spalten ein- und wieder auszublenden. Die Komponente ist dabei so ähnlich wie das Multiselect-Dropdown in der Registrierung aufgebaut, nur das die Werte mittels einer Checkbox ausgewählt werden können und ausgewählte Einträge nicht außerhalb des Dropdowns sichtbar werden. Die Komponente wurde ohne `React Hook Form` realisiert. Stattdessen wird ihr eine `onChange()` Event-Handler übergeben, der die ausgewählten Spalten in einer Zustands-Variable vom Typ `useState<string[]>()` speichert. Vor dem

Rendering der Spalten muss somit nur noch überprüft werden, ob die angegebene Spalte in dieser Variable enthalten ist.

Durch **Suchfeld** kann der User die Tabelle direkt nach dem Namen eines bestimmten Influencer durchsuchen. Dazu wird ebenfalls der Wert des Suchfeldes in eine Variable vom Typ `useState<string>()` gespeichert. Anschließend wird eine Filterfunktion mithilfe des Inhalts dieser Variable auf allen Daten angewendet und die Tabelle aktualisiert.

B. Backend

Die Hauptaufgaben des Backends umfassen die Aufbereitung und Bereitstellung von Daten für das Frontend sowie die Verarbeitung von Benutzeranfragen. Dabei haben wir uns für das Python-Webframework Flask entschieden. Flask ermöglicht die Erstellung von Webanwendungen in Python durch eine minimalistische und flexible Struktur, sodass es sich gut für kleine bis mittelgroße Projekte eignet. Durch das Definieren von Routen, basierend auf den Anforderungen der Clients werden die HTTP-Anfragen verarbeitet und entsprechende Antworten zurückgeben. In unserem Fall haben wir folgende Routen definiert:

Route	Zweck der Route
/signup	Registrierung: Die eingegebenen Daten werden validiert und verschlüsselt in der MongoDB gespeichert.
/login	Anmeldung: Die Eingaben werden überprüft, und bei erfolgreicher Authentifizierung eine Session gestartet.
/profile	Profildaten ändern: Gibt die Profilinformationen des eingeloggten Benutzers zurück.
/logout	Abmelden: Beendet die aktuelle Sitzung und meldet den Benutzer ab.
/collect	Influencer Übersichtstabelle: Gibt die Daten aller registrierten Influencer gesammelt zurück, um diese im Frontend anzuzeigen
/<session>	Routen für Session-Handling

Tabelle I
DEFINIERTE API-ROUTEN

1) **Datenvalidierung und -speicherung:** Die Validierungsfunktionen prüfen die Benutzereingaben mithilfe von regulären Ausdrücken und einfachen Bedingungen, um sicherzustellen, dass sie den erwarteten Standards entsprechen. Dabei wird z.B. überprüft, ob die E-Mail im richtigen Format vorliegt, das Passwort ausreichend lang ist oder obligatorische Informationen wie Vor- und Nachname korrekt angegeben wurden. Spezielle Validierungen wie die Überprüfung des Instagram-Benutzernamens und optionaler Informationen wie der Telefonnummern werden ebenfalls durchgeführt. Darüber hinaus nutzen diese Funktionen externe Ressourcen, beispielsweise eine Webanfragen für die Überprüfung von Instagram-Benutzernamen auf ihre Existenz und ob es sich um ein privates oder öffentliches Profil handelt.

2) **Session:** In unserer Flask-Anwendung nutzen wir die mitgelieferte Session-Verwaltung der Flask-Bibliothek. Wenn sich ein Benutzer anmeldet oder registriert, wird seine E-Mail-Adresse in einer Session gespeichert. Diese Funktionalität gewährleistet, dass der Benutzer während seines Besuchs auf der Webseite identifiziert bleibt, was essenziell ist, um personalisierte Inhalte und Funktionen bereitzustellen. Die Flask-Bibliothek bietet uns ebenfalls die Möglichkeit, Session-Daten

sicher zu verwalten. Zum Schutz sensibler Informationen wie der Benutzer-E-Mail verwenden wir Konfigurationsoptionen um die Datenübertragung zu verschlüsseln. Zusätzlich sorgt die eingestellte Sessionlaufzeit dafür, dass die Session nach 30 Minuten der Inaktivität automatisch abläuft und die Daten somit vor unautorisiertem Zugriff geschützt werden.

C. Instagram-Account Analyse

Daten an ChatGPT senden, -> wie, durch welche Methode (Fetch, oder axios, ajax-API) Prompt kurz und knapp beschreiben. Überprüfung ob Produkt-WerbeSparte-Tags richtig sind(?) kurz noch beschreiben, welche Daten sonst noch gescrap werden Libraries erwähnen und vll. einzelene Methoden, instascrape, ...

D. Persistenz

In der Datenbank sind die Benutzerinformationen strukturiert in einem Schema gespeichert, die nachfolgend definiert sind.

Datenbank-Schema

Die Hauptaufgaben der Datenbankschicht umfassen das Speichern, Aktualisieren und das Abrufen dieser Benutzerdaten. Es wird sichergestellt, dass zurückgegebene Daten immer das definierte Standardformat einhalten, um die Konsistenz weiterer Verarbeitungsschritte innerhalb der Anwendung sicherzustellen. Fehler während dem Speichern oder dem Abrufen von Benutzerdaten werden für eine zuverlässige Fehlerbehandlung genau protokolliert.

VI. TESTEN

Ihr müsst nicht all so viel übers Testing schreiben. Kurz das Testing-Tool beschreiben und deren Besonderheiten / wie es grob funktioniert. Anschließend noch ein paar Worte zur Test-Coverage und evtl. entdeckte Fehler.

A. Frontend

kurz Jest beschreiben, wie es grob funktioniert, z.B. dass in Jest die Komponenten selbstständig gerendert werden können und man Werte in Inputfelder setzen kann und man anhand geschmissener Fehlermeldungen der Formulare die Richtigkeit überprüfen kann...

B. Backend

Die Verwendung von pytest ermöglichte uns umfassendes, aber effizientes Testen unserer Datenvalidierungslogik. Pytest ist ein Testframework für Python, das das Schreiben, Organisieren und Ausführen von Tests vereinfacht, da diese mit einfachen Mitteln zu implementieren sind. Durch die Nutzung der Funktionen von pytest wie Fixtures, Mocks und Parametrisierung konnten wir eine robuste Validierungslogik für die eingegebenen Daten des Users sicherstellen, bzw. potenzielle Probleme beseitigen und so schlussendlich die Zuverlässigkeit der gesamten Anwendun verbessern.

VII. FAZIT UND AUSBLIK

InfluenzaConnect bietet zum aktuellen Zeitpunkt nur einen kleinen Ausschnitt der geplanten Funktionalität. Das MVP wurde in der zu verfügbaren Zeit nur knapp verfehlt, dafür wurde die WebApp sehr ansprechend und solide aufgebaut. Ein erster Schritt wurde hinsichtlich der automatischen Analyse des Instagram-Profiles des Influencers getan. Diese kann zukünftig noch erweitert, verbessert und auf weitere Social-Media-Plattformen angewendet werden.

Diese Strukturierung und Automatisierung mittels Docker und GitHub Actions bietet eine robuste Basis für den Betrieb und die Weiterentwicklung von InfluenzaConnect".

VIII. FUNCTIONAL REQUIREMENTS

Influencer sollen sich auf der Plattform registrieren, damit diese von Unternehmen leichter aufgefunden werden, um ihre Produkte zu vermarkten. Nachfolgend werden die besuchenden User von InfluenzaConnect, die ihr Produkt vermarkten wollen als 'Unternehmen' und die User, die sich vermarkten wollen, als 'Influencer' bezeichnet.

A. MVP

1) *Influencer sollen sich registrieren*

Influencer sollen sich auf der Plattform registrieren, damit Sie auf der Plattform angezeigt und von Unternehmen gefunden werden können.

Akzeptanzkriterien:

Influencer sollen bei der Registrierung folgendes in ihrem Profil hinterlegen:

- Ihre Kontaktdaten
- eine kleine Beschreibung über sich selbst
- Ihr Instagram-Profil

Nach der Registrierung ist der Influencer angemeldet und es wird der Home-Screen angezeigt

2) *Webscraping des Instagram-Profiles*

Das Instagram-Profil des Influencers wird automatisch analysiert, um relevante Informationen für die bessere Selbstvermarktung zu speichern. Diese Informationen werden den Unternehmen später bereitgestellt, um den für sie richtigen Influencer zu finden.

Akzeptanzkriterien:

- Influencer - falls möglich - einer Produkt-Werbe-Sparte zuordnen. Falls dies fehlschlägt
→ manuelle Eintragung der Produkt-Werbe-Sparte vom Influencer selbst
- Analyse der Reichweite des Influencers, indem die Anzahl der Follower, Likes und Anzahl der Posts mit berücksichtigt werden.
- Das Profilbild soll von Instagram gescrapt werden

3) *Übersicht über alle registrierten Influencer*

Die Unternehmen möchten auf einer Übersichtsseite einen Überblick über alle auf der Plattform registrierten Influencer bekommen, damit sie den richtigen Influencer, zu ihrem Produkt finden können.

Akzeptanzkriterien:

- Pro Influencer soll eine kleine Auswahl relevanter Informationen auf der Übersichtsseite angezeigt werden.
- Influencer sollen sortiert angezeigt werden
- mittels Suche und Filter können die angezeigten Einträge eingeschränkt werden.

4) *Informationen zum Influencer*

Unternehmen möchten durch Auswahl eines Influencers auf der 'Übersichtsseite der registrierten Influencer' nähere

Informationen zu diesen angezeigt bekommen, um den Influencer besser beurteilen zu können, ob dieser zum Unternehmen passt.

Akzeptanzkriterien:

- Durch Auswahl eines Eintrages auf der Übersichtsseite werden alle relevanten Informationen zum ausgewählten Influencer angezeigt.
- Kontaktinformationen sollen verlinkt werden, um die Influencer schnell kontaktieren zu können.

B. Optionale Anforderungen

1) *Integration von Instagram-Posts in der Detailansicht*

Unternehmen wollen die neuesten Instagram-Posts eines Influencers in der Plattform-Detailansicht angezeigt bekommen, um einen Einblick in deren Inhalte zu erhalten.

Akzeptanzkriterien:

- Die letzten Instagram-Posts werden in die Detailansicht integriert.
- Eine Vorschau der Posts ist sichtbar, einschließlich Bild und Bildunterschrift.

2) *Bildanalyse für Influencer-Profil*

Unternehmen möchten eine Bildanalyse für Influencer-Profile, um die Inhalte besser zu verstehen und relevante Bilder zu identifizieren.

Akzeptanzkriterien:

- Die Bildanalyse identifiziert Bilder, in denen der Influencer selbst zu sehen ist.
- Diese Bilder werden im Profil oder in der Detailansicht angezeigt.

3) *Produkthochladen für die Influencer-Werbung*

Unternehmen möchten Produkte auf der Plattform hochladen, um passende Influencer für Werbekampagnen zu finden und diese mit relevanten Tags zu kennzeichnen.

Akzeptanzkriterien:

- Unternehmen können Produktbilder hochladen.
- Eine Bildanalyse weist den Produkten passende Tags zu.
- Basierend auf den Tags werden passende Influencer vorgeschlagen.

4) *Registrierung und Direktkontakt mit Influencern*

Unternehmen möchten sich auf der Plattform registrieren, um direkt mit Influencern in Kontakt zu treten.

Akzeptanzkriterien:

- Unternehmen können sich registrieren und ein eigenes Profil erstellen.
- Die Plattform ermöglicht direkte Kommunikation zwischen Unternehmen und Influencern.

5) *Übersicht über aktuelle Kontakte mit Influencern*

Unternehmen möchten eine Übersicht über ihre aktuellen

Kontakte mit Influencern haben, um die Zusammenarbeit im Blick zu behalten und mögliche Partnerschaften zu verwalten.

Akzeptanzkriterien:

- Unternehmen können auf ihrer Profilseite eine Liste der Influencer sehen, mit denen sie in Kontakt stehen.
- Die Plattform zeigt an, ob Nachrichten oder Kooperationen zwischen Unternehmen und Influencern bestehen.

6) *Vergleichsfunktion für Influencer*

Unternehmen möchten eine Vergleichsfunktion haben, um verschiedene Influencer direkt nebeneinander vergleichen zu können und so die beste Entscheidung für Kooperationen zu treffen.

Akzeptanzkriterien:

- Die Plattform ermöglicht es, mehrere Influencer auszuwählen und deren Profile nebeneinander zu vergleichen.
- Die Vergleichsfunktion zeigt relevante Metriken wie Follower-Zahl, Engagement-Rate und Produktübereinstimmung an.

IX. DATA ACQUISITION

Um relevante Informationen über Instagram-Influencer zu sammeln, verwenden wir eine Kombination aus der Instagram-API und Web-Scraping. Die API liefert grundlegende Daten wie Follower-Zahlen und Engagement-Raten, während Web-Scraping zusätzliche Informationen wie Profildetails, Beiträge und Story-Daten extrahiert. Wir nutzen Python und Bibliotheken wie instascrape, um diese Daten zu sammeln und in einer Datenbank zu speichern. Die gesammelten Informationen können dann für potenzielle Werbepartner zugänglich gemacht werden. Die Umsetzung wird dabei in einen Web-Service ausgelagert.

X. ARCHITECTURAL GOALS

Die Architektur von InfluenzaConnect zielt darauf ab, ein robustes und skalierbares System zu schaffen, das effizient mit externen API's und den Benutzern der WebApp interagiert. Durch Nutzung modernster Webtechnologien, soll eine hohe Verfügbarkeit und Wartbarkeit des Systems gewährleistet werden.

XI. ARCHITECTURE OF INFLUENZACONNECT

A. Technology Stack

Um die genannten Anforderungen zu erfüllen haben wir uns für folgenden Tech-Stack entschieden:

Das Backend basiert auf Python mit Flask, da diese Sprachen einen sehr modernen Ansatz bieten. Außerdem fördert ein in Python geschriebenes Backend die Möglichkeit, dass auch Junior Developer einen einfachen Einstieg in den Code finden können. Das Frontend wird mit React und TypeScript entwickelt, wobei Tailwind CSS für Design und Styling verwendet wird. MongoDB dient als Datenbanksystem, mit GraphQL

für Datenbankabfragen, was eine flexible und leistungsfähige Datenmanipulation ermöglicht.

B. Frontend

Das Frontend von InfluenzaConnect ist darauf ausgerichtet, eine intuitive und interaktive Benutzeroberfläche zu bieten, die Nutzern eine effiziente Navigation durch die Plattform ermöglicht. Für die Entwicklung dieser interaktiven Komponenten nutzen wir React und TypeScript, die eine solide Struktur bereitstellen. Tailwind CSS erleichtert uns das schnelle und flexible Design der Benutzeroberflächen.

C. Backend

Python gepaart mit Flask bildet das Rückgrat unseres Backends und ermöglicht die Entwicklung von API-Services, die essenziell für Funktionen wie Benutzerauthentifizierung und Datenmanagement sind. Diese technische Wahl gewährleistet Flexibilität im Umgang mit verschiedenen Datenquellen. Zusätzlich erleichtert es die Integration mit Plattformen wie Instagram, wodurch unsere Anwendung eine breitere Palette von Marketing-Tools effizient unterstützen kann.

D. Persistence

Die Daten werden mittels einer MongoDB persistiert, ein NoSQL-Datenbanksystem, das für seine Flexibilität und Leistungsfähigkeit bei der Verwaltung großer Mengen unstrukturierter Daten bekannt ist. GraphQL wird eingesetzt, um effiziente und flexible Datenabfragen zu ermöglichen, die es Entwicklern erlauben, genau die Daten abzurufen, die sie benötigen.

LITERATUR

- [1] Paul Brandl, Manuel Kalla, Dominik Panzer, Kevin Paulus, Manuel Pickl, Franziska Rubenbauer, Berkay Yurdagül und Christoph P. Neumann. *NeunerIn: Eine MEVN-basierte Webanwendung zum kompetitiven Kartenspielen*. Techn. Ber. CL-2023-11. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, Juli 2023. DOI: 10.13140/RG.2.2.33933.31209.
- [2] André Kestler, Antonio Vidos, Marcus Haberl, Tobias Dobmeier, Tobias Lettner, Tobias Weiß und Christoph P. Neumann. *Computer Vision Pipeline: Eine React- und Flask-basierte Webanwendung zur No-Code-Bildverarbeitung mit Cloud-Deployment*. Techn. Ber. CL-2023-08. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, Juli 2023. DOI: 10.13140/RG.2.2.23866.98248.
- [3] Jakob Götz, Uwe Kölbel, Maximilian Schlosser, Oliver Schmidts, Jan Schuster, Philipp Seufert, Fabian Wagner und Christoph P. Neumann. *Nautical Nonsense: Eine Phaser3- und FastAPI-basierte Webanwendung für Schiffe-Versenken mit Cloud-Deployment*. Techn. Ber. CL-2023-07. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, Juli 2023. DOI: 10.13140/RG.2.2.17156.09601.
- [4] Lukas Feil, Stefan Reger, Timon Spichtinger, Manuel Pickl, Gian Piero Cecchetti, Alexander Hammer, Berkay Yurdagül und Christoph P. Neumann. *Torpedo Tactics: Eine MEVN-basierte Webanwendung für Schiffe-Versenken mit Cloud-Deployment*. Techn. Ber. CL-2023-06. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, Juli 2023. DOI: 10.13140/RG.2.2.22608.69120.
- [5] Rebecca Kietzer, Baran Baygin, Carl Küschall, Jonathan Okorafor, Luca Käsmann, Michael Zimmet, Michael Ippisch und Christoph P. Neumann. *Stockbird: Eine React-basierte Webanwendung mit serverless Cloud-Deployment zur Analyse des Einfluss von Tweets auf Aktienkurs-Schwankungen*. Techn. Ber. CL-2023-04. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, Juli 2023. DOI: 10.13140/RG.2.2.32675.02083.
- [6] Christian Rute, Alex Müller, Alexander Rudolf Wittmann, Arthur Zimmermann, David Nestmeyer, Julian Tischlak, Matthias Wolfinger und Christoph P. Neumann. *FancyChess: Eine Next.js-basierte Cloud-Anwendung zum Schachspielen*. Techn. Ber. CL-2023-03. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, Juli 2023. DOI: 10.13140/RG.2.2.19253.24802.
- [7] Anastasia Chernysheva, Jakob Götz, Ardian Imeraj, Patrice Korinth, Philipp Stangl und Christoph P. Neumann. *SGDb Semantic Video Game Database: Svelte- und Ontotext-basierte Webanwendung mit einer Graphen-Suche für Videospiele*. Techn. Ber. CL-2023-02. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, März 2023. DOI: 10.13140/RG.2.2.11272.60160.
- [8] Johannes Horst, Manuel Zimmermann, Patrick Sabau, Saniye Ogul, Stefan Ries, Tobias Schotter und Christoph P. Neumann. *OPCUA-Netzwerk: Angular- und FastAPI-basierte Entwicklung eines OPC-UA Sensor-Netzwerks für den Heimbereich*. Techn. Ber. CL-2023-01. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, März 2023. DOI: 10.13140/RG.2.2.22177.79209.
- [9] Alexander Ziebell, Anja Stricker, Annika Stadelmann, Leo Schurrer, Philip Bartmann, Ronja Bäumel, Ulrich Stark und Christoph P. Neumann. *Wo ist mein Geld: Eine MERN-basierte Webanwendung für gemeinsame Ausgaben mit Freunden oder Kollegen*. Techn. Ber. CL-2022-11. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, Juli 2022. DOI: 10.13140/RG.2.2.28888.67847.
- [10] Bastian Hahn, Martin Kleber, Andreas Klier, Lukas Kreussel, Felix Paris, Andreas Ziegler und Christoph P. Neumann. *Twitter-Dash: React- und .NET-basierte Trend- und Sentiment-Analysen*. Techn. Ber. CL-2022-07. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, Juli 2022. DOI: 10.13140/RG.2.2.15466.90564.
- [11] Tobias Bauer, Fabian Beer, Daniel Holl, Ardian Imeraj, Konrad Schweiger, Philipp Stangl, Wolfgang Weigl und Christoph P. Neumann. *Reddiment: Eine SvelteKit- und Elasticsearch-basierte Reddit Sentiment-Analyse*. Techn. Ber. CL-2022-06. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, Juli 2022. DOI: 10.13140/RG.2.2.32244.12161.
- [12] Florian Bösl, Helge Kohl, Anastasia Chernysheva, Patrice Korinth, Philipp Porsch und Christoph P. Neumann. *Explosion Guy: Cloud-basiertes Matchmaking für einen graphischen Bombenspaß*. Techn. Ber. CL-2022-05. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, Juli 2022. DOI: 10.13140/RG.2.2.18822.34882.
- [13] Dominik Smrekar, Johannes Horst, Patrick Sabau, Saniye Ogul, Tobias Schotter und Christoph P. Neumann. *OTH-Wiki: Ein Angular- und FastAPI-basiertes Wiki für Studierende*. Techn. Ber. CL-2022-04. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotech-

- nik, Medien und Informatik, Juli 2022. DOI: 10.13140/RG.2.2.25533.23526.
- [14] Johannes Halbritter, Helge Kohl, Lukas Kreussel, Stephan Prettner, Andreas Ziegler und Christoph P. Neumann. *Graphvio: Eine Graphdatenbank-Webanwendung für integrierte Datensätze von Streaminganbietern*. Techn. Ber. CL-2022-01. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, März 2022. DOI: 10.13140/RG.2.2.12111.46244.
 - [15] Tobias Bauer, Albert Hahn, Lukas Kleinlein, Nicolas Proske, Leonard Wöllmer, Andrei Trukhin und Christoph P. Neumann. *Covidash: Eine MEAN-Variation-basierte Webanwendung für Inzidenz-Zahlen und Impffortschritt in Deutschland*. Techn. Ber. CL-2021-06. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, Juli 2021. DOI: 10.13140/RG.2.2.33921.84321.
 - [16] Cameron Barbee, Tim Hoffmann, Christian Piffel, Tobias Schotter, Sebastian Schuscha, Philipp Stangl, Thomas Stangl und Christoph P. Neumann. *FireForceDefense: Graphisches Tower-Defense-Spiel mit Kubernetes-Deployment*. Techn. Ber. CL-2021-05. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, Juli 2021. DOI: 10.13140/RG.2.2.20500.07048.
 - [17] Egidia Cenko, Madina Kamalova, Matthias Schön, Christoph Schuster, Andrei Trukhin und Christoph P. Neumann. *MedPlanner: Eine Angular- und Django-basierte Webanwendung um ärztliche Termine übersichtlich zu verwalten*. Techn. Ber. CL-2021-04. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, Juli 2021. DOI: 10.13140/RG.2.2.19409.71528.