Übergabebericht 08.09 – Matbakh.app

Empfänger: cChat – neuer Product Owner

Erstellt von: ChatGPT (bisheriger PO)

Datum: 08.09.2025

Sprache: Deutsch

Status: Teil 1 von X



🧩 Teil 1: Projektüberblick & Zielsetzung



matbakh.app - Sichtbarkeits- und Performanceplattform für Gastronomie



Matbakh ist eine Plattform zur datengestützten Sichtbarkeitsanalyse und -optimierung für Gastronomiebetriebe. Ziel ist es, lokale Restaurants, Cafés und kreative Orte dabei zu unterstützen, über Google Business, Instagram & Co. besser sichtbar zu werden, mehr Buchungen zu erzielen und ihre digitale Präsenz zu professionalisieren – auch ohne eigene Marketingabteilung.

Zentrale Use Cases

- 1. Visibility Check (VC): KI-gestützte Analyse der digitalen Sichtbarkeit (Google, Instagram, Facebook, Website etc.)
- 2. Forecasting: Vorhersage kommender Reichweitenveränderungen (Trendanalysen, Benchmarks)
- 3. Al-Empfehlungen: Smart generierte To-Dos zur Verbesserung der Sichtbarkeit und Conversion
- 4. Google Profile Service (B2B): Pflege von Google Business Profiles als Dienstleistung für Restaurants
- Onboarding & Self-Service UI: Wizard-gestützter Flow für neue Betriebe



🗱 Teil 2: Technologiestack & Infrastruktur

Backend

- AWS Lambda Functions: Sichtbarkeits-Check, Forecast, File Upload
- Node.js Runtime: Hauptsächlich 20.x, Python 3.11 partiell
- AWS Services:
 - S3 (Uploads, Reports)
 - CloudFront (Website Delivery)
 - DynamoDB (Prompt Cache, VC Results)
 - Secrets Manager (API Keys)
 - Cognito (Auth im Umbau)
 - RDS PostgreSQL (für Businessdaten)
 - Bedrock (Claude 3.5 Sonnet f
 ür Al)

KI & Prompt-System

- Claude 3.5 Sonnet via Bedrock (alle Prompts, inkl. DSGVO-Konformität)
- Prompt Templates mit Versionierung
- Prompt Logging & Redaction Policies in Planung (Task 2.1f-j)
- Al-Orchestrierung über Lambda Agents (Claude, Gemini, später Meta/Opal)

Frontend

- React + Tailwind
- ShadCN UI
- Zustand, React Query, Vite
- Auth derzeit im Wechsel (SimpleAuth → Cognito/Supabase)

■ Daten

- Supabase (Alt-Projekt)
- RDS für Kernmodelle (Locations, Visibility Snapshots)
- DynamoDB (PromptCache, ResultCache)

i Teil 3: Stand der Umsetzung (Stand: 08.09.2025)

Abgeschlossene Core-Komponenten

- [x] Visibility Check Generator (Claude) mit Framework-Ausgaben
- [x] Claude Memory Layer (Kontexthistorie, Sessions)
- [x] Forecasting Engine (Score-Trends, Wochenvorschau)
- [x] Google My Business Crawler (Basic Profilanalyse)
- [x] DSGVO Consent Logger (serverseitig, /track-consent)
- [x] Upload System (S3, Tokens, MIME Check)
- [x] Template Security System (KMS-basiert, RSASSA Signatur, Audit-Trail)
- [x] Event Type System für Trendanalysen

• [x] Score-Based Recommendation Engine (Drops/Stagnation erkennen)



Aktueller Task (wichtig!)

Task 7.4 - Test Suite Cleanup & Business Validation Layer

Ziel: Alle Testsuiten im Business- und Memory-Bereich konsolidieren und stabilisieren:

Subtas k	Datei/Testbereich	Maßnahme
7.4.1	ab-testing-manager.test.ts	Realistische Mock-Daten, neue mockSend-Logik
7.4.2	rollback-manager.test.ts	Aufrufvalidierung vereinfachen
7.4.3	<pre>approval-workflow-manager.t est.ts</pre>	IDs dynamisch generieren
7.4.4	<pre>performance-tracking-manage r.test.ts</pre>	Zielkonflikte abbilden
7.4.5	memory-manager.test.ts	UUID-Mock final stabilisieren
7.4.6	mocks/uuid.ts	Konfigurierbarer Mock
7.4.7	tests/context-factory.t s	Zentrale Kontexterstellung
7.4.8	shared/setup.ts	Globale Mocks zentralisieren

📥 Nächste Teile folgen:

- Teil 4: Offene Aufgaben & ToDos
- Teil 5: Bugs, Fallstricke & Empfehlungen
- Teil 6: Al Layer & Prompt-System im Detail
- Teil 7: Deployment, CI/CD, GitHub & AWS Setup
- Teil 8: Personas, Businessmodell & Investorenstrategie
- Teil 9: Kontaktpunkte, Partner & Stakeholder

🧭 Überblick über den Arbeitsstand

Du übernimmst matbakh.app in einem sehr weit fortgeschrittenen, aber noch nicht finalen Zustand. Die meisten zentralen Komponenten sind umgesetzt und getestet. Jetzt beginnt die Phase der Stabilisierung, UI-Integration, DSGVO-Härtung und KI-Optimierung.

Wichtige Offene Aufgaben (Stand 08.09.2025)

[TASK 7.4] Test Suite Cleanup & Business Validation

Status: In Arbeit

Priorität: Hoch

Ziel: Alle Business-Tests auf grün bringen (Unit + Integration)

Beschreibung: Siehe Teil 3 oder TASK_ID: 7.4 – betrifft alle zentralen

.test.ts-Dateien

memory-manager.test.ts war der letzte aktive Fokus, inkl. Mock-Problemlösung für uuid

¡TASK 2.1f–2.1j] DSGVO Logging System für Claude Prompts

Status: Noch offen

Priorität: Sehr hoch (Audit-Konformität)

Komponenten:

- Prompt Logging & Redaction Rules
- DSGVO Audit API & Export
- Admin Audit Dashboard
- Prompt Logging Policy Generator
- Policy Enforcement im Claude-Lambda

Claude-Nutzung ist schon produktionsreif, aber noch ohne DSGVO-Export & Audit-Tooling. Dieser Task schützt das Projekt rechtlich & technisch.



Status: Fertiggestellt (09.09.2025)

Ziel: Vergleich eines Betriebs mit Branchenwerten (z. B. Google Reviews, Instagram

Engagement etc.)

Ergebnis: System erkennt Drop, Stagnation, Verbesserung – liefert

Handlungsempfehlungen

Verlinkt mit: VCResult Komponente & Claude Prompt Templates

[TASK 6.4.6] Forecast Preview Completion

Status: Chart-Komponenten fertig (06.09.2025)

Offen: Darstellung im Wizard & Integration in Sichtbarkeits-Auswertung

Widgets: ForecastChart, ForecastControls

[TASK ?] Threat Detection Engine (Backlog)

Status: Noch nicht gestartet

Ziel: KI-gestützte Erkennung von:

- Bot-Traffic (z. B. gekaufte Bewertungen)
- Massenänderungen (Plattform-Missbrauch)
- inkonsistentem Verhalten (z. B. lokaler Drop bei gleichzeitigem globalem Anstieg)

Technische Grundlage:

- Claude VC Generator
- EventTypeSystem (Score Drops, Plateaus, Anomalien)
- Möglichkeit zur Klassifizierung via ML oder heuristischer Regeln

Empfohlen:

- → Mit 6.4.6.x und 7.x konsolidieren
- → Hooks zu Claude + UI-Benachrichtigungen vorbereiten
- → DSGVO-konforme Speicherung aller Alarme (Auditfähig!)

(TASK ?) Persona-Adaptive Output Logik

Status: Teilweise begonnen

Beschreibung: Claude-Antworten unterscheiden je nach Zielpersona (Anna, Markus, Sophie, Jürgen)

- → Beispiel: Anna = Wellness & Bio, Markus = Familie & Planbarkeit
- → Anpassung des Tons, der Empfehlungen, sogar der UI-Botschaften

Empfohlene Umsetzung:

- Claude Prompt Templates mit persona_variant
- Dynamischer Prompt-Zusatz je nach Zielgruppe
- Später: dynamisches UI-Theming + Plattform-Priorisierung

DSGVO: Offene Risiken (hohe Priorität)

Bereich	Risiko	Empfehlung
Prompt Logging	Kein vollständiger Audit-Log	2.1f-j finalisieren
E-Mail-Flows (Upload, Consent)	Kein Double-Opt-In umgesetzt	SES + SNS Reminder-System aufbauen
Consent UI/API	Nur Grundversion aktiv	Weiterleitung + Tracking via /track-consent absichern
Claude Output	Kein Pseudonymisierungs-Che ck	Claude Risk Classification + Redaction starten

Bekannte Bugs & Fallstricke

Bereich	Problem	Status
Auth-Migration	Mismatch zwischen SimpleAuth und Supabase	In Bearbeitung, Kontext prüfen
useAuth() Hooks	Kontextfehler bei mehreren Providern	ggf. auf Single-Provider-Verschaltun g umstellen
ForecastDemo.ts x	UI lädt Chart, aber keine erklärenden Texte	Inhalt nachreichen

memory-manager.
test.ts

UUID-Mock greift nicht bei Referenz-Check Gefixt (siehe 7.4.5)

mockSend-Handling

Teilweise zu eng validiert (toHaveBeenCalledWith)

Wird in 7

in Zentrale KI-Komponenten & Claude Integration

1. Claude VC Generator (via AWS Bedrock – Claude 3.5 Sonnet)

- Aktuell verwendetes Modell: Claude 3.5 Sonnet (AWS Bedrock)
- Orchestriert via Kiro
- Prompt-Aufrufe erfolgen über den zentralen vc_generator Agent

Input-Quelle:

- Sichtbarkeitsdaten (Google, Instagram)
- Benchmarks, Forecast-Daten, EventType-Auswertung
- Persona, Kategorie, Plattformfokus

Output:

- VCResult mit Handlungsempfehlung
- SWOT, Balanced Scorecard, Porter's Five Forces
- Plattform-Strategien & Zielgruppenspezifik

Aktive Claude Prompt Templates

Gespeichert in: .kiro/templates/claude/

Template	Beschreibung	Persona-Unter stützung	DSGVO-k onform
vc_core.txt	Hauptlogik für Sichtbarkeitsauswertung	V	Logging fehlt
swot_analysis.txt	SWOT-Generierung pro Betrieb	V	V

platform_strategy.txt	Plattform-Empfehlungen (z. B. Google vs. TikTok)	V	V
persona_adaptive.txt	Ausgabe-Stil je nach Zielgruppe	in Arbeit	X
<pre>forecast_interpretatio n.txt</pre>	Erklärt Score-Verläufe mit Trends	✓	V
benchmark_comparison.t	Vergleich mit Markt & Top-Performern	V	V

Agentenstruktur (Claude / Prompt-bezogen)

Agent-ID	Aufgabe	Input	Output	Provide r
vc_generator	Sichtbarkeitsanalyse + Handlung	VC-Daten, Persona, Plattform	VCResult	Claude
<pre>prompt_manag er</pre>	Verwaltung & Kombination von Templates	TaskID, Persona	Prompt	lokal
memory_manag er	Kontextspeicherung & Optimierung	History, Tasks	MemoryContex t	lokal
audit_logger	Prompt-Log mit DSGVO-Redaktion	Prompt, Response, Meta	LogEntry	lokal
risk_classif ier	Prüfung auf Claude-Risiken	Claude-Output	RiskScore, Warning	geplant

Rollenmodell – Claude Output Personas

Person a	Kurzprofil	Output-Stil	Plattformpriorität
Anna	gesundheitsbewusste Genießerin	achtsam, emotional, Wellness-orientiert	Instagram, Google
Marku s	organisierter Familienvater	rational, planbar, praktisch	Google, Facebook

Sophie	trendaffine Studentin	hype-driven, emojis, Social-First	TikTok, Insta, Spotify
Jürgen	geselliger Best-Ager	persönlich, klassisch, offline-affin	WhatsApp, E-Mail, Google Maps

Ziel:

- → Adaptive Claude-Antworten,
- → Adaptive UI-Komponenten (z. B. CTA, Farbwahl, Tonalität)
- → Persona-Einstellungen künftig dynamisch vom Nutzer aus wählbar

Prompt Memory Layer

- Memory-System zur Nachvollziehbarkeit, Konversationshistorie, Claude-Lernfortschritt
- Enthält: conversationHistory, taskHistory, contextMetadata
- UUIDs werden gemockt in Tests (`mocks/uuid.ts

Sichtbarkeits-UI-Komponenten (React)

Matbakh.app nutzt ein modulares Widget-System, mit dem sich die Claude-Ergebnisse, Scores und Handlungsempfehlungen dynamisch im UI abbilden lassen.

Hauptkomponenten

Komponente	Zweck	Status	Datenquelle
VCResult.tsx	Haupt-Output-Kompon ente für Claude-Ergebnisse	✓ Fertig	Claude VC Generator
ScoreTrendChart.tsx	Sichtbarkeitsverlauf in Linienform (Score + Zeit)	✓ Fertig	Forecast + EventType
ForecastChart.tsx	Prognose-Zeitverlauf + Interaktion	Fertig	Forecast Engine
ForecastControls.tsx	UI für Prognoseoptionen	Fertig	useForecast()

	(z. B. Szenarioauswahl)		
PlatformRecommendatio n.tsx	Empfehlung pro Plattform (Google, IG, TikTok)	✓ Fertig	Claude
PersonaOutputBox.tsx	Stil- und Sprachanpassung für Persona-Ausgabe	In Umsetzun	Claude, PersonaContext
SWOTDisplay.tsx	SWOT-Tabelle oder -Grid mit Visualisierung	✓ Fertig	Claude SWOT Engine
BenchmarkComparison.t sx	Branchendurchschnitt vs. Standort	✓ Fertig	VCResult, Branchenbenchmark s
ClaudeWarning.tsx	Claude-spezifische Hinweise (z. B. DSGVO-Risiko)	Prototyp	Risk Classifier

Wizard-Flows

Die App nutzt derzeit folgende Flows zur Darstellung und Navigation durch die Analyse:

Flow	Beschreibung	Status
VC Wizard	Schritt-für-Schritt durch Visibility Check, Claude-Auswertung, Benchmark, Forecast	Beta
Forecast Preview	Isolierter Flow zur Darstellung des Prognoseverlaufs	✓ Aktiv
Claude Output Viewer	Entwickleransicht für Claude Prompts + Ausgaben	✓ Nur Debug-Modus
Consent + Upload Flow	Upload und Consent-Vergabe für Nutzer + Profilverknüpfung	Nur Grundversion, DSGVO-Lücken



Ziel:

- Claude erkennt Zielpersona automatisch
- UI zeigt Ergebnis im passenden Stil und Kanal
- Output-Widget (Text + CTA + visuelle Rahmung) wird angepasst

Beispiel:

- Anna → "¼ Deine Sichtbarkeit auf Instagram kann durch Reels mit Entspannungsfaktor 🌈 gesteigert werden."
- Markus → "Optimieren Sie Ihre Google-Einträge: 3 familienfreundliche Merkmale fehlen noch."

Entwicklerhinweise

- Alle Komponenten sind modular, zustandsbasiert und nutzen zentrale Hooks wie useVCResult(), usePersona(), useForecast()
- Zustand wird via Zustand oder Context API gehalten
- Forecast-Komponenten sind getrennt vom Hauptflow, aber über vcResult.forecast verbunden
- DSGVO-Relevanz: Keine Claude-Ausgabe darf ohne Consent eingeblendet werden \rightarrow TODO!

Externe & Interne Datenquellen (Visibility Check)

Die VC-Analyse nutzt verschiedene Datenquellen, die über REST APIs oder Supabase-Tabellen eingespeist werden. Alle Quellen fließen in die Sichtbarkeitsbewertung, Claude-Prompts und die Forecast-Logik.

Plattformdaten

Quelle	Zweck	API / Zugriff	Status
Google Business	Bewertungen, Beiträge, Öffnungszeiten, Keywords, Sichtbarkeit	Google Business Profile API (aktiv)	✓ Aktiv
Instagram	Posts, Follower, Engagement Rate, Bio	Instagram Graph API (manuell via Token)	Eingeschränkt
Facebook	Bewertungen, Beiträge, Interaktionen	FB Graph API	Eingeschränkt

MyBusiness Benchmark API	Durchschnittswerte je Kategorie	Custom (interner Cache)	✓ Fertig
Claude Events	Generiert aus	intern,	✓ Implementiert
API	VCResult-Analyse	EventTypeSystem	
Forecast	Score-Prognose auf Basis vergangener Verläufe	intern, Regression /	✓ Getestet,
Engine		Pattern-Matching	UI-ready

📊 Supabase Tabellenstruktur (Auszug)

Tabelle	Inhalt	Relevanz
<pre>visibility_check_lea ds</pre>	Lead-Info, Consent-Status, Basisdaten	UI, DSGVO
<pre>visibility_check_act ions</pre>	Einzelne Bewertungseinträge	Claude-Prompt
<pre>visibility_forecast_ data</pre>	Verlauf + Prognosewerte pro Betrieb	Forecast UI
<pre>visibility_event_log s</pre>	Trigger für EventTypes (Drop, Plateau…)	Claude-Prompt + Alerts
platform_scores	Einzelwertungen je Plattform (0–100)	UI
persona_profiles	Interne Klassifikation nach Persona	Adaptive Output

Forecasting & Anomalieerkennung

Die Score-Prognose basiert auf historischen Datenpunkten (`{ date,

Externe & Interne Datenquellen (Visibility Check)

Die VC-Analyse nutzt verschiedene Datenquellen, die über REST APIs oder Supabase-Tabellen eingespeist werden. Alle Quellen fließen in die Sichtbarkeitsbewertung, Claude-Prompts und die Forecast-Logik.



Quelle	Zweck	API / Zugriff	Status
Google Business	Bewertungen, Beiträge, Öffnungszeiten, Keywords, Sichtbarkeit	Google Business Profile API (aktiv)	✓ Aktiv
Instagram	Posts, Follower, Engagement Rate, Bio	Instagram Graph API (manuell via Token)	Eingeschränkt
Facebook	Bewertungen, Beiträge, Interaktionen	FB Graph API	Eingeschränkt
MyBusiness Benchmark API	Durchschnittswerte je Kategorie	Custom (interner Cache)	✓ Fertig
Claude Events API	Generiert aus VCResult-Analyse	intern, EventTypeSystem	✓ Implementiert
Forecast Engine	Score-Prognose auf Basis vergangener Verläufe	intern, Regression / Pattern-Matching	Getestet, UI-ready

Tabelle	Inhalt	Relevanz
<pre>visibility_check_lea ds</pre>	Lead-Info, Consent-Status, Basisdaten	UI, DSGVO
<pre>visibility_check_act ions</pre>	Einzelne Bewertungseinträge	Claude-Prompt
<pre>visibility_forecast_ data</pre>	Verlauf + Prognosewerte pro Betrieb	Forecast UI
<pre>visibility_event_log s</pre>	Trigger für EventTypes (Drop, Plateau…)	Claude-Prompt + Alerts
platform_scores	Einzelwertungen je Plattform (0–100)	UI
persona_profiles	Interne Klassifikation nach Persona	Adaptive Output

Forecasting & Anomalieerkennung

Die Score-Prognose basiert auf historischen Datenpunkten wie:

type ScorePoint = { date: string; score: number };

Diese Punkte werden pro Plattform (Google, Instagram etc.) gespeichert und analysiert.

Verfügbare Analyseverfahren

Methode	Beschreibung	Einsatz	
Lineare Regression	Einfacher Trend über X Wochen	Standardprognose	
Gleitender Durchschnitt (MA)	Glättung von Ausreißern, besser für volatile Daten	Optional über Controls	
Heuristische Rules	"Wenn 3 Tage kein Post, dann sinkt Score"	♠ In Arbeit	
EventType Mapping	Sucht Muster wie Drop, Plateau, Anstieg	✓ Aktiv	
Claude Forecast Prompt	Interpretiert Verlauf textlich ("Seit Juni fallender Trend")	✓ Aktiv	
Benutzerdefinierter Einfluss	User plant Kampagne → Einfluss simulieren	Geplant	
Forecast Output-Shape (Beispiel) { forecast: { platform: "Instagram", predictedScore: 62, trend: "falling", confidence: 0.78, explanation: "Dein Engagement ist seit 3 Wochen rückläufig. Keine neuen Reels seit KW 35." } }			



EventType-System – technisches Fundament

Ermöglicht die dynamische Auslösung von KI-Ereignissen (Trigger für Claude, UI-Hinweise, E-Mail-Alerts etc.)



Beispiele für EventTypes:

Type	Auslöser	Technischer Schwellenwert
drop	Sichtbarkeits-Score fällt stark	>10 Punkte in <7 Tagen
platea u	Keine Veränderung über Zeitraum	Δ < 2 Punkte in 14 Tagen
spike	Sichtbarkeit springt plötzlich nach oben	>20 Punkte Anstieg
gap	Keine Daten verfügbar	z. B. keine IG Posts > 14 Tage
anomal y	Inkonsistenz zw. Plattformen	z.B. FB steigt, IG fällt

Diese EventTypes erzeugen Einträge in visibility_event_logs, die in Claude-Prompts oder Widgets verwendet werden.

Claude Forecast Interpretation Prompt (Template)

Der Prompt forecast_interpretation.txt ist aktiv und erwartet folgendes Format:

```
{
  platform: "Google",
  scorePoints: [
    { date: "2025-08-01", score: 80 },
    { date: "2025-08-08", score: 75 },
    ...
],
  eventTypes: ["drop", "gap"],
  persona: "Markus",
  businessContext: { ... }
}
```

Claude generiert daraus:

"In den letzten Wochen ist dein Google-Score leicht gefallen. Der Rückgang fällt zusammen mit einer Phase ohne neue Rezensionen und einem veralteten Eintrag auf Google Maps. Für Markus als Familienvater sind aktuelle Infos entscheidend – z. B. ob man draußen sitzen kann."

X Integration in UI

Komponenten:

- < <ForecastChart />
- <ForecastControls />
- <VCResultSection type="forecast" />

Status:

- Komponenten sind umgesetzt
- Demo-Seite ForecastDemo.tsx zeigt Chart + Interaktion
- Integration in den vollständigen VC Wizard ist offen

ToDo:

- Forecast in finaler VC-Auswertung anzeigen
- EventTypes als Badges o. Alerts hervorheben
- Claude-Ausgabe im Forecast-Abschnitt anzeigen
- Nutzerinteraktion ermöglichen (Simuliertes Szenario "Was wäre wenn...")

Claude Prompt Templates

Alle Claude-Prompts sind modular, versionierbar und zentral in .kiro/templates/claude/gespeichert.

Sie bilden das Herzstück der VC-Analyse und sind mit dynamischen Parametern befüllbar.

Aktuelle Templates (Stand 08.09.2025)

Template	Funktion	Persona-A daption	DSGVO-Konf ormität
vc_core.txt	Hauptauswertung Sichtbarkeit	V	Logging fehlt
swot_analysis.txt	SWOT-Analyse nach Unternehmensdaten	V	V
platform_strategy.txt	Empfohlene Plattform-Priorisierung	V	V
<pre>benchmark_comparison.t xt</pre>	Vergleich mit Branchendurchschnitt	V	V

<pre>forecast_interpretatio n.txt</pre>	Erklärt Verlauf & Prognose	V	V
persona_adaptive.txt	Output-Stil, Sprache, Fokus je Persona	In Arbeit	×

Alle Templates verwenden { {handlebars} }-Syntax mit dynamischer Datenfüllung durch den prompt_manager.

K Claude Prompt Runtime Stack

Komponente	Aufgabe
vc_generator (Agent)	Orchestriert Claude Calls mit korrekt befülltem Template
prompt_manager	Kombiniert Template + Input zu finalem Prompt
audit_logger (geplant)	Zeichnet Prompt + Response DSGVO-konform auf
risk_classifier (geplant)	Bewertet Claude Output auf Datenschutzrisiken
memory_manager	Speichert Kontext, Tasks, Verlauf (optional Claude-Vergessen)



Transparenz, Datenschutz, Pseudonymisierung und Rechtssicherheit beim Einsatz von Claude via Bedrock

Geplante Komponenten

Teilaufgabe	Beschreibung
2.1f – Prompt Logging & Redaction Rules	Logging sensibler Prompts + automatische Schwärzung von personenbezogenen Feldern (E-Mail, Adresse, etc.)
2.1g – DSGVO Audit API & Export	JSON- und PDF-Export aller Prompts + Claude-Antworten mit Zeitstempel, Kontext, Trigger
2.1h – Admin Audit Dashboard	Einsicht, Suche, und Überprüfung der Prompt-Logs mit Filterfunktionen

2.1i – Prompt Logging Policy Generator	Automatische Erstellung von Logging-Policies nach Claude-Typ, Nutzerrolle, Zweck
2.1j – Prompt Policy Enforcement	Einbau der Policy-Prüfung vor Claude-Aufruf – Prompt wird nur erlaubt, wenn Logging-Pflicht erfüllt

Supabase Integration (Alt-Projekt)

Matbakh.app ist ursprünglich auf Supabase gestartet. Einige Module (z. B. Leads, Consent, Forecasting) sind weiterhin aktiv.

Achtung: Das Projekt läuft aktuell parallel mit **AWS-Infrastruktur**. Supabase dient als Übergangs-Backend für nicht-kritische Daten und UI-Schnittstellen.

Relevante Tabellen (Alt-Supabase-Projekt: uheksobnyedarrpgxhju)

Tabelle	Zweck	Status
<pre>visibility_check_lead s</pre>	Leads + Einwilligungen (DSGVO)	Aktiv
visibility_check_actions	Bewertungsdaten (Einzelplattform)	Aktiv
<pre>visibility_forecast_d ata</pre>	Score-Zeitreihen + Prognosen	Aktiv
visibility_event_logs	Drops, Anomalien, Claude-Trigger	Aktiv
persona_profiles	Mapping zu Claude Personas (Anna etc.)	Aktiv
platform_scores	Scores pro Plattform (0–100, gerundet)	Aktiv

Diese Tabellen werden später in ein AWS-basiertes System überführt (RDS oder DynamoDB), sobald DSGVO-Stack und Forecast stabil sind.





Dienst	Aufgabe	Ressource / ARN
SES (Simple Email Service)	Transaktionale E-Mails (Consent, Reports)	SNS Topic: arn:aws:sns:eu-central-1:055062 860590:matbakh-ses-notify
SNS	Benachrichtigungs-Weiterleit ung (z. B. Fehler, Klicks)	Aktiv
SQS (DLQ)	Dead Letter Queue für gescheiterte E-Mails	ses-events-queue, ses-events-dlq

™ Geplante Flows

- Consent Bestätigung: Double-Opt-In mit E-Mail-Link
- VC Report Versand: Direktversand nach Analyse (optional mit PDF)
- **Reminder Flow**: Für Leads ohne abgeschlossenen Upload (Cron-basiert via Lambda)

Noch offen: Consent-Flow vollständig automatisieren (POST /track-consent + E-Mail-Versand via SES Trigger)

🔐 Authentifizierung

Aktueller Stand: SimpleAuth + Supabase Auth (Alt)

Provider	Zielgruppe	Zustand
SimpleAuth	VC-Wizard (Lead UI)	✓ Aktiv
Supabase Auth	Admin-Panel / UI-Test	O Übergangsweise

⚠ Problematisch: Einige Komponenten nutzen useAuth() aus Supabase, andere aus $Simple Auth-Context \rightarrow Konflikte\ bei\ Helmet Dispatcher,\ Provider-Mismatch\ usw.$

Empfehlung für cChat:

- Nur ein Auth-Provider für alle internen Uls einsetzen (Cognito bevorzugt)
- Alle useAuth()Aufrufe zentralisieren
- Upload-Wizard & VC-Analyse öffentlich zugänglich lassen, aber mit Tracking-Token absichern

Upload-System

Ziel: DSGVO-konformer, sicherer Upload von Screenshots (z. B. Instagram Insights)

Technische Basis

Element	Beschreibung
S3 Bucket	matbakhvcstack-webbucket12880f5b-svct6cxfbip5
Pre-Signed URL Upload	Client-seitiger Upload via /api/upload-url
Token-Validation	tokenManager.ts prüft Upload-Tokens (z.B. Lead-Zugehörigkeit)
Metadata-Verarbeitung	Upload-Lambda validiert Bild, Größe, Berechtigung
DSGVO-Logik	Consent-Check vorher via /track-consent

Zusätzliche DSGVO-Anforderungen

Koadmap: Phase 2–4 (Stand: 08.09.2025)

Die künftige Entwicklung ist in 6 Phasen aufgeteilt. Aktuell befinden wir uns in **Phase 2**, mit vorbereiteten Strukturen für Phase 3–4.

Phase 2: Stabilisierung & DSGVO-Compliance

Ziel: System konsolidieren, Tests grün, Auditfähigkeit herstellen

Meilenstein	Beschreibung	Status
✓ Claude VC Generator	Claude 3.5 Sonnet via Bedrock, Output-Templates	Fertig
✓ Forecast Engine	Verlauf + Prognose (Regression, Anomalie-Erkennung)	Fertig

⚠ DSGVO Logging	Prompt-Logging, Policy Check, Audit-Dashboard	Offen (TASK 2.1f–j)
Upload-System Finalisierung	Tokenprüfung, Consent-Sicherung, Logging	90%
Test Cleanup (TASK 7.4)	Alle relevanten Tests grün + Mocks vereinheitlicht	In Arbeit
Auth-Zentralisierung	Konflikte Supabase/SimpleAuth lösen	Teilweise

Ziel: Claude-Ausgaben an Zielgruppen und UI-Elemente anpassen

Task	Beschreibung	Status
7.5 – Persona-Adaptive Output	Tonalität, Empfehlungen, Design-Entscheidungen je Persona	Teilweis e
6.4.6.2 – Forecast in VC-UI integrieren	ForecastWidgets in Wizard sichtbar machen	Offen
Claude CTA Mapping	Empfehlungen als klickbare Elemente / Widgets	Offen
Theme Switching	Farben, Icons, Layouts je Persona adaptiv laden	geplant

Ziel: Claude-Ausgaben analysieren, klassifizieren und kontrollieren

Task	Beschreibung	Status
Risk Pattern Detection	Claude-Antworten nach GDPR Risk / Halluzination prüfen	geplanter Agent: risk_classifier
Prompt Logging Policy Engine	Logging-Regeln je Rolle, Typ, Kontext	TASK 2.1i
Runtime Enforcement	Claude-Call nur bei gültiger Policy	TASK 2.1j
Admin Risk Dashboard	Übersicht von Claude-Ausgabe-Sicherheiten	geplant
Diese Phase ist entscheidend für Enterprise-Readiness (z. B. Hotelketten,		

Franchise-Systeme, Kliniken).

Investment Readiness & Stakeholder Impact

Das System erfüllt bereits jetzt:

Kriterium	Status
Multi-Tenant Struktur	✓ vorhanden
DSGVO-Architektur	A aktiv in Arbeit
Claude Prompt Engine (modular)	✓ vorhanden

Sichtbarkeits-Analyse (B2B) ✓ einsatzbereit

Forecast + Anomalie-Logik ✓ vorhanden

Leadgenerierung + Funnel ✓ produktiv

UI-Basis (Wizard, Ergebnis, Upload) ✓ produktiv

Nächste Schritte zur Investmentfähigkeit:

- DSGVO-Stack vollständig implementieren (Teilaufgaben 2.1f–j)
- Claude Risk-Scoring + Audit-Dashboard live
- 3–5 zahlende Pilotkunden onboarden (z. B. Giesinger Garten, Sapralott)
- Demo-Video + Landingpage für Investoren
- KPI-Monitoring (z. B. CTR auf Empfehlungen, Upload-Quote, Conversion von Leads
 → Kunden)

Empfehlung für cChat (neuer PO)

- Behalte <u>TASKS.md</u> + <u>TODO.md</u> im Blick
- Nutze .kiro/agents/, .kiro/templates/, .kiro/tasks/ als Steuerzentren
- Nächste Meilensteine (September–Oktober):
 - DSGVO Export & Consent (TASK 2.1g, 2.1h)
 - Persona-Adaptive VCResult Komponente + Forecast-Zusatz (TASK 6.4.6.x)
 - Logging Policy Enforcement & Claude Risk Scoring (TASK 2.1j, 2.1k, 2.1l)

Projektstruktur (monorepo-ähnlich)

Die Projektstruktur ist in logische Domänen und technische Layer aufgeteilt:

KI-Agenten, Prompt-Vorlagen, Tasks

| — agents/ # Claude-Agenten (vc_generator etc.)

| templates/ # Prompt-Templates (claude/)

```
– tasks/
               # Dokumentierte Tasks + Spec-Einträge
              # App-Code (Frontend + Utils)
- src/
— pages/
           # Wizard UI, ForecastDemo, Upload
  — components/
                   # VCResult, ForecastChart, CTAWidgets
              # ClaudeHandler, Auth, UploadLogic
  — lib/
  — hooks/
                # useVisibility, useConsent, useForecast
  — config/
                 # Plattformen, Personas, Thresholds
— tests/
                # Business Tests (ab-testing, approval etc.)
 ___ mocks__/
                   # Globale Mocks (uuid.ts, send.ts)
# AWS Lambda Funktionen (VC Flow, Upload etc.)
– lambda/
— vc-handler/ # Orchestriert Claude Engines
L— upload-handler/ # Upload-Security, Consent-Logging
— infra/
               # AWS CDK Infrastruktur (S3, RDS, SES, etc.)
```

Claude Prompt Pipeline (VC Generator Flow)

Die KI-Auswertung im Visibility Check folgt einer exakt definierten Pipeline:

1. Trigger

Ein VC-Request (via Upload oder Wizard) sendet Unternehmensdaten an den Server.

2. Datenquelle anreichern

- Benchmarks
- EventTypes
- Forecast-Verlauf
- Persona-Zuweisung
- → Alles via Supabase oder interne APIs

3. Prompt-Bau (vc_generator)

- kombiniert alle Inputs
- lädt Handlebars-Template aus .kiro/templates/claude/vc_core.txt
- befüllt Prompt dynamisch
- sendet an Claude 3.5 (AWS Bedrock)

4. Antwortverarbeitung

- Claude-Antwort wird in VCResult überführt
- Handlungsempfehlungen, Plattformstrategien etc.
- Ggf. Weiterleitung an Forecast, Benchmark, RiskClassifier

5. Output-Rendering

- UI-Komponente VCResult.tsx rendert Empfehlungen + Plattform-CTAs
- Adaptiv je nach Persona (→ persona_adaptive.txt)

🔐 Dev Standards & Quality Gates

Teststrategie

- Jest, ts-jest
- Fokus: Businesslogik, Claude Prompt Validierung, Risk Prevention
- TASK 7.4 → Test Suite Cleanup (Unit + Edge Cases + MockValidierung)

Security & GDPR

- Consent-Capture via /track-consent
- Upload-Security via Token, IP-Check
- DSGVO Logging Stack (geplant: 2.1f–j)
- Claude Prompt Safeguards in Vorbereitung (Pseudonymisierung etc.)

Claude Safeguards

- Keine Prompt-Nutzung ohne Kontext
- Keine PII ohne Redaction (geplant via audit_logger)
- Rate Limits & Cost-Tracking (über Bedrock)

API-Landschaft

API Route	Methode	Zweck	Auth	Statu s
/upload	POST	Datei-Upload mit Consent	Token	V
/vc	POST	Sichtbarkeitsauswertung starten	Token	V
/track-conse nt	POST	Consent-Logging + EventTrigger	offen	<u> </u>
/forecast	GET	Verlauf + Prognose	öffentlich	V
/benchmark	GET	Branchendaten laden	öffentlich	V

Deployment & Infrastruktur

Hosting

- Frontend:
 - Deployment via GitHub Actions
 - S3 Bucket: matbakhvcstack-webbucket12880f5b-svct6cxfbip5
 - O CloudFront Distribution:

- ID: E2W4JULEW8BXSD
- Domain: d23vp19r70m1vk.cloudfront.net
- Aliases: matbakh.app, www.matbakh.app
- Deployments triggern npm run build + aws s3 sync

• Lambdas:

- Deploy via GitHub Actions mit IAM Role:
 - arn:aws:iam::055062860590:role/gh-actions-web-deploy
- o index.ts als zentraler Entry-Point für vc-handler
- upload-handler nimmt Consent + Dateisecurity entgegen

GitHub Repositories

- Hauptrepo: matbakh-app/matbakh-visibility-boost
- Alle Deploy-Skripte unter .github/workflows/
- Claude Templates & Kiro Tasks unter .kiro/

Supabase Konfiguration

Supabase Projekt

- Ref: uheksobnyedarrpgxhju
- URL: https://uheksobnyedarrpgxhju.supabase.co
- Wichtig: Projekt bleibt für Übergangszeit bestehen, Mid-Term Migration zu RDS geplant

7111001

🂳 Relevante Tabellen

Taballa

labelle	Zweck
<pre>visibility_check_lead s</pre>	Lead-Daten, Consent, Metadaten
visibility_check_acti ons	Einträge pro Sichtbarkeitsfaktor
platform_scores	Score je Plattform (0–100)

visibility_forecast_d Historie & Prognosepunkte
ata

visibility_event_logs EventTypes (Drop, Plateau, etc.)

persona_profiles Kategorisierte

Persona-Zuweisung

Authentifizierung

Auth-Migration: Supabase → SimpleAuth (Custom)

Status: Mischbetrieb mit Inkonsistenzen

→ DRINGENDER TODO-Punkt!

Problem:

- Teile der App verwenden SimpleAuthProvider
- Andere Komponenten (z. B. useAuth()) erwarten Supabase-Kontext
- → Fehler: useAuth returns undefined oder context mismatch

Empfehlungen:

- Kurzfristig: SimpleAuthProvider überall durchziehen (Single-Source)
- Langfristig: Re-Migration zu Cognito ODER Supabase v2 klären
- Authentifizierte Routen: nur bei /upload und /vc erforderlich

Monitoring & Logging

📦 Claude Logging (geplant):

Audit-Stack mit:

- audit_logger Agent
- o /admin/audit/export
- DSGVO-Konformität (siehe TASK 2.1f–j)

📬 E-Mail / Events

- SES Events Notifications:
 - o SNS Topic:

```
arn:aws:sns:eu-central-1:055062860590:matbakh-ses-notify
```

- SQS Queue für E-Mail-Tracking:
 - ses-events-queue
 - DLQ: ses-events-dlq
- Noch keine Alerts eingebaut (Reminder-System geplant)

📊 Datenübertragung an Claude

Wichtig:

- Claude Prompting läuft über:
 - vc_generator Agent (Claude 3.5 Sonnet)
 - Via AWS Bedrock (kostenpflichtig, rate-limitiert)
- Zentrale Claude-Aktivität erfolgt über vc-handler (Lambda)
- Prompt-Input

Forecast System & EventType-Erkennung

Das Forecast-Modul gehört zu den zentralen Analysewerkzeugen und unterstützt Nutzer*innen dabei, Trends zu erkennen, Maßnahmen abzuleiten und Plattformstrategien anzupassen.

Forecast-Datenquelle

- Tabelle: visibility_forecast_data
- Struktur:

```
{ date: string, score: number, source: "real" | "forecast", confidence?: number }[]
```

Daten stammen aus:

- Historischen Scores (platform_scores)
- Ereignissen (visibility_event_logs)
- Forecast-Engine (interne Regression + Pattern-Matching)

EventType System

Erkennt folgende Muster im Score-Verlauf automatisch:

EventTyp e	Bedeutung	Beispielreaktion Claude
drop	Sichtbarkeit nimmt stark ab	Warnung + schnelle Maßnahme
plateau	Score stagniert über längeren Zeitraum	Empfehlungen für Reaktivierung
rise	Sichtbarkeit verbessert sich messbar	Verstärkung von Maßnahmen
anomaly	Inkonsistenter Verlauf, z. B. ungewöhnlicher Anstieg bei schlechter Bewertung	Claude-Risiko-Analyse

- Event-Auswertung erfolgt serverseitig
 - → gespeichert in visibility_event_logs
- Verknüpft mit Claude Prompt Templates
 - \rightarrow beeinflusst die Inhalte in vc_core.txt + forecast_interpretation.txt

ForecastChart & ForecastControls

UI-Komponenten:

- ForecastChart.tsx: D3/Chart.js Komponente mit Zeitverlauf + Markern
- ForecastControls.tsx: User-Interaktion, Zielsetzung, Erklärung

Aktueller Status:

✓ Komponenten vollständig umgesetzt

Noch nicht in finalen Wizard eingebunden

Claude Forecast-Prompt

Template: .kiro/templates/claude/forecast_interpretation.txt

Zweck:

- Erklärt, was der Score-Verlauf bedeutet
- Liefert kontextbasierte Empfehlungen (z. B. bei Drops oder Plateaus)
- Kann auf Persona eingehen

Input:

- Score-Verlauf (visibility_forecast_data)
- Aktuelles EventType
- Persona + Plattform

Aufgabe

🧱 Empfehlung für nächste Schritte

·	
<pre>ForecastDemo.tsx finalisieren</pre>	Beispielseite als vollständige Demo fertigstellen
Integration in VC-Wizard	Forecast-Auswertung nach Consent/Upload anzeigen
/ UI-Test (TASK 7.4.6.2)	Forecast Rendering + Event-Mapping testen

Beschreibung

Claude-Rückbindung aktivieren	Forecast-Ergebnis in vc_generator Prompt einbauen
	Forecast-Werte + Events in Export-Log aufnehmen (siehe TASK 2.1g–j)

Ziel dieses Moduls

Matbakh generiert auf Basis der Sichtbarkeitsdaten nicht nur abstrakte Scores, sondern konkrete Handlungsempfehlungen, welche Plattformen für welchen Betrieb und für welche Persona den größten Hebel bieten. Claude erstellt diese Strategien individuell anhand von Scoring, Forecast, Branchendaten und Zielgruppentypik.

Zielgruppen (Persona-System)

Aktuelle Zielgruppenstruktur:

Person a	Beschreibung	Entscheidungsstil	Plattformfokus
Anna	Achtsame Genießerin, 32, Urban, Bio	Emotional, visuell	Instagram, Google
Marku s	Organisierter Familienvater, 45	Rational, planungsorientiert	Google, Facebook, WhatsApp
Sophie	Trendbewusste Studentin, 24	Hype-orientiert, social	TikTok, Insta, Spotify
Jürgen	Best Ager, 58, klassisch, offline-stark	Vertrauen, Empfehlungen	Google Maps, WhatsApp

X Claude Templates (Persona-abhängig)

- .kiro/templates/claude/persona_adaptive.txt
- → Dynamisch befüllbar mit Zielpersona + Plattformdaten

Aktueller Status:

- In Arbeit (TASK 7.5)
- ▼ Teile werden bereits aus persona_profiles geladen
- X Prompt-Stil nicht vollständig angepasst

Ziel:

- Inhalt, Tonalität, CTA-Formulierung je nach Persona automatisch anpassen
- Claude lernt:
 - Welche Keywords bei welcher Persona wirken
 - o Welche Plattform mit welcher Botschaft verknüpft ist

Plattformstrategien (Claude-Output)



Claude gibt pro Plattform eine Einschätzung + Empfehlung, z. B.:

Plattform	Bewertung	Empfehlung	
Google	92/100	Standortdaten aktuell, viele Bewertungen, mehr Fotos	
Instagram	63/100	Feed okay, aber Stories und Reels ausbauen	
Facebook	41/100	Wenig Interaktion, evtl. schließen oder repurposen	

 $[\]rightarrow$ Empfehlung wird später durch CTAWidgets.tsx als Handlungsvorschlag dargestellt (TASK 6.5)

🗩 Adaptive Handlungsempfehlungen – Pipeline

1. Input:

- Plattformdaten (platform_scores)
- Persona (persona_profiles)
- Forecast-EventType (drop, plateau, etc.)
- Kategorie & Zielgruppe

2. Claude Templates:

- vc_core.txt (Kombiniert alles)
- persona_adaptive.txt (Wählt Ton & Stil)
- platform_strategy.txt (Plattform-Fokus)

3. Output (Claude):

- o Zielgruppenoptimierte Vorschläge
- Plattformpriorisierung mit Ranking
- Aktions-CTAs (z. B. "Poste ein Reel zum Lunch-Menü mit Standort-Tags")

4. UI-Verknüpfung:

- VCResult.tsx → rendert Empfehlungen
- \circ PlatformCTA.tsx \rightarrow call-to-action Buttons pro Plattform
- Persona-Stil → optional via adaptives UI-Theming (später)

Nächste Schritte für dich (cChat)

Aufgabe	Beschreibung
Persona-Adaption abschließen	Prompt persona_adaptive.txt fertigstellen + in vc_generator verlinken
Adaptive Tests einbauen	Unit-Test für unterschiedliche Persona → Claude Output verifizieren
UI-Anpassung vorbereiten	CTA-Komponenten adaptiv gestalten (z. B. Farbwelt, Icons, Sprache)
CTA-Tracking aufbauen	Jede Handlung via Button → Logging für Erfolgsbewertung

© Ziel dieses Moduls

Matbakh verfolgt ein klares Ziel: **Sichtbarkeit** → **Handlung** → **Buchung**.

Dazu erzeugt Claude aus den Sichtbarkeitsdaten **plattformspezifische Empfehlungen**, die im UI als **konkrete Call-To-Actions (CTAs)** erscheinen.

Diese CTAs sind der direkte Hebel für Gastronomen – und mittelfristig auch der **Umsatz-Hebel für Matbakh.app**.

CTA-System – Architektur

Komponente	Beschreibung
VCResult.tsx	Rendert die Claude-Ausgabe als adaptiven Block
PlatformCTA.tsx	Einzeln renderbare CTA-Karten pro Plattform
persona_adaptive.tx t	Stimmungs- & Sprachsteuerung des CTA (z. B. "Zeig dich ")
platform_strategy.t xt	Plattform-Scoring + ToDos (z. B. "Reel mit Standort posten")
<pre>cta_widgets.config. ts</pre>	Konfigurierbare Darstellung (z. B. Icons, Farben, Platzierung)
(geplant) cta-tracker.ts	Event-Tracking je CTA-Auslösung zur Erfolgsmessung



Persona: Anna

Plattform: Instagram

Betrieb: Bio-Café mit 78/100 Sichtbarkeit

Claude-Ausgabe (gekürzt):

Dein Instagram-Auftritt zeigt bereits viel Gefühl für Ästhetik.

Poste ein neues Reel mit einer Nahaufnahme eurer goldenen Kurkuma-Latte.

→ Nutze Standort-Tags & #veganmunich für mehr lokale Reichweite.

Wird von PlatformCTA.tsx automatisch so dargestellt:

| Plattform: Instagram

| Aktion: Reel posten mit Standort-Tags

| ★ Empfehlung: Fokus auf vegane Getränke & visuelle Details

Ziel: +15% Reichweite bei Suchanfragen nach "vegan café münchen"

| W Button: "Als ToDo speichern"

CTA-Konfiguration (.ts config)

```
Beispiel: cta_widgets.config.ts

export const PLATFORM_CTA_CONFIG = {
  instagram: {
    icon: 'ioo',
    color: 'gradient-pink',
    personaTone: {
      anna: 'soft',
      sophie: 'hype',
    },
```

```
},
 google: {
  icon: ' 📍 ',
  color: 'blue',
  personaTone: {
    markus: 'rational',
   jürgen: 'klassisch',
  },
 },
};
```

Monetarisierungsstrategie (B2B)

- 1. Free VC Report (aktuell aktiv)
 - → Initialer Check gratis
 - → Consent via /track-consent
- 2. Geplante Premium-Features

Feature	Modell	Status
Wöchentlicher VC-Report per Mail	Abo (9 €/Monat)	geplant
CTA-Umsetzungsdienst (Google/IG)	1:1 Service	in Planung

Dashboard mit Tracking & Forecast SaaS-Plan specced

Sichtbarkeits-Coaching via Partner Coaching Bundle Partnerschafts-Modell

1. Upsell-Idee: "Google Business Pflege-Abo"

- ightarrow Auf Basis der Claude-CTAs: "Wir übernehmen das für dich"
- ightarrow Monetarisierung direkt aus der CTA-Logik ableitbar
- → Beispiel:
 - o Handlung: "Öffnungszeiten regelmäßig aktualisieren"
 - o Lösung: 15 €/Monat wir machen das

■ ToDos für cChat

Aufgabe	Beschreibung
CTA-Tracking einbauen	Jede Interaktion (Klick, Speichern etc.) mitloggen
CTA-Stil adaptiv gestalten	Farben, Icons, Wording je Persona/Plattform
Prompt-Tests: CTA-Validität prüfen	Claude-Ausgaben auf Plattform-Trefferquote testen
Monetarisierung mitStrategieboard verknüpfen	Verlinkung der CTA-Metriken mit Sales-Tickets (geplant in Lovable Agent)

Tuploadsystem – Architekturüberblick

Das Upload-System ist der Einstiegspunkt für viele Nutzer:innen von Matbakh.app. Es ermöglicht die Einreichung eines Betriebs zur Analyse, kombiniert mit einem einfachen DSGVO-konformen Consent-Prozess.

Ablauf

1. Upload Ul

Nutzer:innen geben ein:

- Betriebsname, Adresse, Kategorie
- o optional: Website, Instagram, Facebook
- Benchmarks zur Auswahl
- Datei-Upload (z. B. Menükarte, Screenshot etc.)

2. Client-Validierung

Vorverarbeitung durch useUpload():

- Kategorie validieren (gegen interne Kategorien)
- o ggf. Auto-Vervollständigung durch Google Places

3. Consent-Prozess (2-stufig)

- Nutzer gibt aktives Einverständnis zur Analyse
- POST /track-consent wird ausgelöst
- Consent + Metadaten landen in visibility_check_leads

4. Presigned Upload URL

- Lambda /upload generiert signierte S3-URL
- Datei wird direkt zu AWS S3 geladen (keine Supabase-Storage mehr)

5. Sicherheit

- o Token-Check
- IP-Logging (via Lambda-Header)
- Valid MIME-Type
- Einmal-Upload pro Link
- o Ablaufzeit (15 min) via S3 Policy

🔐 DSGVO Consent & Event Logging

Matbakh behandelt Consent nicht als Checkbox, sondern als nachvollziehbaren Prozess mit auditierbaren Events.



Parameter Beschreibung

platform ['google', 'instagram', 'fb'] etc.

S

purpose z. B. 'visibility_analysis'

timestam Zeitstempel

EventType

ip_hash DSGVO-harmloser IP-Hash

- ightarrow Alle Werte werden in visibility_check_leads gespeichert
- → Claude darf nur nach erfolgreichem Consent ausgelöst werden

Event-System für DSGVO-Aktivitäten

Jeder Consent erzeugt ein Event (analog zu Mailtracking oder Score Drops):

Beschreibung

	3
consent_grant ed	Consent erfolgt für Sichtbarkeitsanalyse
file uploaded	Datei erfolgreich via Presigned URL übermittelt

 $\rightarrow \textbf{Events werden in } \texttt{visibility_event_logs } \textbf{gespeichert}$

SES E-Mail Events (geplant)

Komponente	Beschreibung
matbakh-ses-notify (SNS)	E-Mail-Zustellstatus (Success, Bounce, Complaint)
ses-events-queue	Haupt-Queue zur Verarbeitung
ses-events-dlq	Dead Letter Queue

ightarrow Ziel: DSGVO-konforme Nachverfolgung + Alert bei Zustellproblemen

★ ToDos für cChat

Aufgabe	Beschreibung
<pre>/track-consent Logging erweitern</pre>	Alle relevanten Felder + optional Browser-Daten loggen
Consent-Export ermöglichen	DSGVO-Anfrage beantworten (PDF + JSON-Export)

Claude darf ohne Consent-Event nicht starten

Consent-Status im UI visualisieren

Nutzer:innen zeigen, was getrackt wurde

Persona-System (Zielgruppenlogik)

Matbakh verwendet ein **explizit definiertes Set aus 4 Haupt-Personas** für alle B2C-orientierten Empfehlungen, Claude-Prompts und UI-Darstellungen.

Aktuelle Personas (persistiert in persona_profiles)

Anna Gesundheitsbewusste Genießerin (32), Bio-Restaurants, Yoga, Instagram

Markus Erlebnisorientierter Familienvater (45), plant gern, wandert, nutzt Apps

Sophie Trendbewusste Studentin (24), liebt Streetfood, TikTok, Gruppenentscheidungen

Jürgen Best Ager (58), gesellig, liest Online-Magazine, WhatsApp-affin

→ Zentrale Personas sind durch strukturiertes User Research entstanden

→ Nutzer:innen können (bald) ihre Persona bei der Analyse aktiv auswählen

Claude Prompt Adaption (Persona-Adaptive Output)

Die Claude-Ausgabe wird dynamisch an den Stil und die Bedürfnisse der Persona angepasst.

Prompt-Komponente persona_adaptive.txt

Ebene	Anpassung bei Claude	
Tonfall	Emotional (Anna) vs. Rational (Markus)	
Empfehlungen	Plattformauswahl, Inhalte, Sprache	
CTA-Texte	Direktive (Sophie) vs. beratend (Jürgen)	
Symbolik & Beispiele	Social Dining, Familienangebote, Bio-Siegel etc.	
\rightarrow Prompt Manager erkennt Persona aus dem Lead und setzt den passenden Zusatz im Template		
→ Diese Logik ist bereits in der Claude-Pipeline aktiv , aber noch nicht final getestet		

W UI-Komponenten: Persona-Spezifische Darstellung

Ziel: Inhalte werden nicht nur semantisch, sondern auch visuell & funktional angepasst.

🔧 Aktueller Umsetzungsstand

Komponente	Persona-Adaptiv	Status
VCResult.tsx	Teilweise	in Arbeit
CTAWidgets.tsx	✓ Ja	implementiert

PlattformPriorities Ja implementiert

X Nein ForecastChart.tsx geplant

→ Beispiel: Anna bekommt andere CTA-Texte ("Finde dein Bio-Konzept!") als Sophie ("Boost dein TikTok-Game!")

🧡 Lovable-Ausrichtung (Design & Wirkung)

Ziel ist eine emotional ansprechende, personalisierte Nutzererfahrung, die Nutzer:innen überrascht und begeistert.

🌸 Lovable Prinzipien in Matbakh.app

Hebel	Umsetzung	
Überraschungseffek t	Score-Auswertung mit inspirierendem Zitat	
emotionale Sprache	Je nach Persona ("dein Wohlfühlort")	
Gamification	geplanter Score-Badge + Persona-Signet	
Storytelling	Claude-Ausgabe beginnt mit Kontext-Intro	
visuelle Qualität	Forecast & VCResult in ruhigem, klaren Stil	
→ Ziel: Erinnerungswürdige Momente schaffen, nicht nur Ergebnisse liefern		



_		_	
_	_	_	-
	-		•
	а.	Э1	•

Beschreibung

persona_adaptive.txt finalisieren	Claude-Output vollständig auf Tonalität und Beispiele prüfen
VCResult.tsx erweitern	Textbausteine & CTA nach Persona differenzieren
Persona-UI-Auswahl ermöglichen	User kann Persona im Wizard aktiv auswählen
Persona-System mit Forecast verknüpfen	Zukünftige Empfehlungen auch im Score-Kontext adaptiv darstellen

Plattformstrategien (VC Output)

Matbakh.app analysiert, **welche Plattform** für einen Betrieb **besonders relevant** ist – basierend auf Branche, Zielgruppe, Engagement-Faktor und aktuellem Content.

Claude Prompt: platform_strategy.txt

Claude generiert eine Plattform-Empfehlung wie:

- "Google Business Profil stärken Bewertungen sind entscheidend für Familienkunden."
- "Instagram ist dein Schaufenster zeig Bilder von deinen veganen Lunch-Angeboten."

🔑 Entscheidungskriterien:

Kriterium	Beispielhafte Ausprägung
Sichtbarkeit	Google Reviews, Google Maps Views
Engagement	Likes, Kommentare, Shares

Aktualität Letzter Post, Story, Event etc.

Persona-Fit Anna = Insta, Jürgen = Google Maps

Kategorie-Match Streetfood = TikTok, Fine Dining = Google

- → Empfehlung ist stets **begründet**, kein Ranking, sondern strategischer Fokus
- ightarrow Claude Output wird in VCResult . tsx als Plattform-Fokus-Box dargestellt

Feature Flags

Feature Flags steuern, **welche Funktionen aktiv** sind – ideal für schrittweise Rollouts & Tests.

Aktuelle Flags (in .kiro/config/featureFlags.json)

Flag Name	Wirkung	Status
enableForecastChart	Zeigt Prognose-Chart im Wizard	✓ Aktiv
enableClaudeRiskChec ks	Trigger für Risk Classifier	Deaktiviert
personaAdaptiveOutpu t	Aktiviert Claude-Persona-Tonfall	Teilweise
<pre>consentPolicyStrictM ode</pre>	Erzwingt DSGVO-Check vor Claude-Aufruf	Geplant

- → Flags können jederzeit erweitert werden
- → Nutzung via useFeatureFlag() Hook (in hooks/useFeatureFlag.ts)

Matbakh Future Enhancements (Phasenplan)

Die langfristige Vision ist bereits in einem konsolidierten Dokument enthalten, mit über 65 Einzelaufgaben (persistiert in .kiro/tasks/future_enhancements.md).

Phasenübersicht:

Phas e	Fokus	Status
1	Core VC Flow + Claude Integration	✓ Abgeschlossen
2	DSGVO Logging & Security Stack	In Arbeit
3	Forecast & Anomalieerkennung	Teilweise
4	Al Risk Detection & Prompt Auditability	Geplant
5	Persona-Adaptive Output & UI	In Arbeit
6	Behavioral Economics & Gamification	Backlog

- → cChat hat volle Kontrolle über die Priorisierung
- → Claude wird künftig auch Preise, Events und Angebotsideen ausgeben (Phase 6)

***** Empfehlungen an cChat

Handlung	Warum?
DSGVO Tasks (2.1f–j) zuerst fertigstellen	Rechtliche Sicherheit ist zwingend
Prompt Logging & Memory System ausbauen	Claude-Outputs analysieren, vergleichen, auditieren
Forecast Chart vollständig in Wizard einbinden	Steigert Nutzerbindung & Planbarkeit
Persona-UI-Switch sichtbar machen	Mehrwert für User + Test neuer Strategien
Platform-CTA-Verlinkungen setzen	Handlungsempfehlung mit Conversion verbinden

Solution Claude Kostenkontrolle (Cost Control Engine)

Die Nutzung von **Claude 3.5 Sonnet via AWS Bedrock** ist kostenpflichtig – jedes Prompt verursacht Kosten pro Zeichen (input + output).



Minimierung der API-Kosten bei gleichzeitiger Maximierung der Qualität.

Aktuelle Umsetzung:

Mechanismus	Beschreibung	Status

Prompt-Length Optimizer	Entfernt irrelevante Felder, kürzt zu lange Bio-/Review-Texte	✓ Aktiv
Persona-based Prompt Scope	Kürzt Prompt abhängig von Persona-Detailgrad (z. B. Jürgen = weniger Emojis)	Teilweise
MaxTokenLimiter	Stellt sicher, dass output nicht über MaxToken läuft	Aktiv
MemoryCleaner	Vermeidet das Mitsenden irrelevanter Historie	In Arbeit

- → Promptlänge wird **vor** Absenden an Bedrock validiert
- \rightarrow Promptkosten werden über eigene Metriken **geschätzt** (estimateClaudeCost() in lib/costTracker.ts)

Prompt-Metriken (Logging & Reporting)

Ziel:

Monitoring & Analyse aller Claude-Calls \rightarrow für Qualitätssicherung, Kostenanalyse, DSGVO-Audit

Erhobene Metriken (geplant / teilweise umgesetzt):

Metrik	Beschreibung	Status
Prompt Länge (Tokens)	Anzahl Zeichen / Tokens input/output	✓ Implementiert
Claude Provider Info	Modell, Version, Call-Timestamp	✓ Implementiert

Audit-Telemetrie (DSGVO + Prompt-Historie)

Ziel:

Vollständige Telemetrie pro Claude-Aufruf – inkl. DSGVO-Nachvollziehbarkeit, Rolle, Zweck, Policy-Match

Core-Felder pro Log-Entry:

Feld	Beschreibung
uuid	eindeutige ID des Calls
timestamp	Zeitstempel (ISO)
userId	Nutzer-ID oder System-Kontext
promptText	Finaler Claude-Prompt (gefiltert)

responseText	Claude Output (ggf. gekürz	

VC, Forecast, SWOT, etc. purpose

Zielpersona (Anna, Markus...) persona

platformFocus z. B. Google, Instagram

policyMatched Welche DSGVO-Policy gegriffen hat

pseudonymizati Art der Schwärzung (falls erfolgt) on

- → Geplant: Speicherung in audit_prompt_logs (Supabase oder RDS)
- → Exportfunktionen (/admin/audit/export) für JSON & PDF sind vorbereitet

🚨 Claude-Ausfall & Fallback-Strategie

Da Claude (Bedrock) zentral ist, muss ein Fallback-System bei Ausfällen verfügbar sein.

Komponente	Verhalten bei Claude-Failure	Status
Claude Timeout Handler	Retry mit Delay (max 3×)	✓ Aktiv
FallbackTemplate	Zeigt statische Tipps + Fehlermeldung	In Arbeit
UI-Warnsystem	"Analyse derzeit nicht verfügbar"	☐ Geplant

Empfehlungen an cChat

Prompt Logging (DSGVO) + Cost-Tracking kombinieren

Frühzeitiges Alert-System bei Claude-Fehlfunktion

Bessere UX

Claude-Fehlfunktion

Mehrwert bei gleichbleibender Kostenstruktur

Prompt-Previews intern testen

Redundanzen, Wiederholungen erkennen & kürzen



Die **VCResult-Komponente** ist das Haupt-UI-Element für den Claude-Output und zeigt die Analyseergebnisse im Wizard nach dem Visibility Check.



src/components/VCResult.tsx



Bereich Beschreibung

Score-Anzeige Gesamt-Sichtbarkeitswert + Plattform-Scores

Handlungsempfehlung	Claude-generierte Tipps für Plattform, Content, lokale Optimierung etc.	
SWOT-Anzeige	Optional eingeblendet, via Claude Prompt (swot_analysis.txt)	
Plattformstrategie	Welche Plattformen nutzen? (via platform_strategy.txt)	
Persona-abhängige CTAs	Z.B. Google Review Push vs. Instagram Story-Vorschläge, je nach Zielgruppe	
$ ightarrow$ Die Darstellung wechselt adaptiv je nach Persona (wenn persona_adaptive.txt aktiv ist)		

Output-Widgets

Jede Claude-Analyse kann **mehrere Widgets** generieren – die VCResult-Komponente entscheidet dann, **welche angezeigt werden**.

Widget-Komponenten (src/components/Widgets/)

Komponente	Zweck	Persona-Adapt ion
PlatformRecommendatio n.tsx	Plattformpriorisierung (Google, Insta, TikTok)	V
ContentIdeas.tsx	3 konkrete Content-Vorschläge	V
QuickWins.tsx	Sofort umsetzbare Maßnahmen	V

PersonaMatch.tsx	Zeigt wie gut der Content zur Zielgruppe passt		
ScoreProgression.tsx	Historischer Score-Verlauf (Chart + Text)	(geplant)	
\rightarrow Alle Widgets folgen einem modularen Design. Neue Claude-Outputs können als neue Widgets eingebaut werden.			

•• Invisible Mode – UI-Minimierung für B2B-Kunden

Viele Gastronom:innen und B2B-Partner bevorzugen eine **nicht überladene UI**. Daher gibt es den **Invisible Mode**, der:

- auf Text-Outputs fokussiert,
- interaktive Komponenten nur bei Mouseover zeigt,
- automatisch den Persona-Stil (Farbpalette, CTA-Sprache etc.) reduziert,
- insbesondere für WhatsApp-/PDF-Export vorbereitet ist.

Aktivierung:

- Wird automatisch aktiviert, wenn "userMode" === 'b2b_simple'
- Alternativ manuell per URL-Parameter: ?invisible=true

Test-Cases:

Testfall	Erwartetes Verhalten
Invisible Mode aktiv	Nur Texte sichtbar, keine Charts
Persona = Markus	Fokus auf Planbarkeit & Familiennutzen
Claude-Ausgabe ohne Forecast	Forecast-Widget wird nicht geladen

Empfehlung

Nächste Schritte für dich, cChat:

Aktion

persona_adaptive.txt finalisieren	Für Output-Tuning pro Zielgruppe
RiskClassifier einbinden	Claude-Ausgaben überwachen & bewerten

DSGVO-Prompt-Logging finalisieren Für vollständige VCResult-Nachvollziehbarkeit

ForecastChart tiefer integrieren Kontextuell passende Vorschau basierend auf Score

Forecast Engine – Prognosebasierte Sichtbarkeitsbewertung

Die Forecast Engine prognostiziert zukünftige Sichtbarkeitsentwicklungen auf Basis historischer Score-Daten ({ date, score }[]). Die Ergebnisse fließen in den Claude-Prompt forecast_interpretation.txt ein und dienen als Frühwarnsystem.

🗱 Technische Grundlagen

Komponente

Beschreibung

visibility_forecast_d Supabase-Tabelle mit Scores & Datum ata

40010100000()	
ForecastChart.tsx	Darstellung des Score-Verlaufs mit Prognose

Hook zum Abruf + Glättung (z. B. linear, sigmoid)

ForecastControls.tsx UI-Kontrollen zum Wechseln von Datenquellen/Algorithmen

Claude interpretiert den Verlauf über einen strukturierten Prompt (→ siehe forecast_interpretation.txt) und ergänzt automatisch Ursachen & Empfehlungen.

EventType-System – Automatische Mustererkennung

Das **EventType-System** analysiert die Score-Zeitreihe und markiert signifikante Veränderungen als "Events".

Technische Implementierung

useForecast()

Komponente	Funktion
<pre>calculateEventType s()</pre>	Findet Plateaus, Drops, Rebounds
visibility_event_l ogs	Persistenz der Events für Claude
eventTypeRules.ts	Business-Logik je Branche & Plattform
Claude Events API	Liefert Events dynamisch als Prompt-Input

EventTypen (Stand 08.09.2025)

EventType	Beschreibung	
DROP	Sichtbarkeitswert sinkt unter Branchenschwelle	
PLATEAU	Sichtbarkeit stagniert trotz Aktivität	
SURGE	Positiver Ausschlag, z. B. viraler Post	
MISALIGNME NT	Inkonsistenz: Lokal schlechter, Plattform-Werte steigen	
SPIKE_FALL	Auffälliger Anstieg + unmittelbarer Rückgang (Bot-Verdacht)	
EventTypes dienen künftig auch als Trigger für Alerts & DSGVO-Logging (z. B. Anomalieprüfung mit risk_classifier).		

Claude-basierte Interpretation (forecast_interpretation.txt)

Dieser Prompt analysiert:

- Ursachen für aktuelle Entwicklung (z. B. zu seltene Posts, Öffnungszeiten falsch)
- Empfehlungen pro Plattform
- Prognosebewertung (z. B. "Wenn keine Änderungen, dann -15 % in 30 Tagen")

Persona-bezogene Interpretation (geplant)

Person	Fokus der Interpretation
а	

Anna Vertrauen, Beständigkeit, emotionale

Sichtbarkeit

Markus Planung, Stabilität, Familiennähe

Sophie Momentum, Trend, FOMO-Vokabular

Jürgen Authentizität, lokale Nähe, Offline-Strategien

forecast_interpretation.txt wird durch persona_variant ergänzt → TASK 7.5

Nächste Schritte

Bereich Aktion

ForecastChart erweitern Beschreibungstext + Tooltip-Interaktivität

Claude Prompt erweitern Integration von EventTypes als Kontextblock

DSGVO-Logging vorbereiten Forecast-Einträge ebenfalls loggen

(Consent-Tracking)

Widget-Integration prüfen Forecast-Insights als VCResult-Widget anzeigen



Matbakh verwendet Claude 3.5 Sonnet (via AWS Bedrock) zur Analyse und Generierung strategischer Empfehlungen. Um DSGVO-Risiken, Halluzinationen oder fehlerhafte Outputs zu vermeiden, wurde ein mehrstufiges Safeguard-System geplant und teilweise vorbereitet.

Safeguard-Komponenten

Komponente	Status	Zweck
prompt_manager	Aktiv	Validiert Templates & Input vor Claude-Aufruf
audit_logger	Geplant	Zeichnet Prompt + Response DSGVO-konform auf
risk_classifier	In Vorbereitung	Bewertet Claude-Outputs auf Risikofaktoren
memory_manager	✓ Aktiv	Speichert Task-Kontext, erkennt Wiederholungen
<pre>prompt_policy_enfo rcer</pre>	Geplant	Prüft: Darf dieser Prompt mit diesem Inhalt gesendet werden?

Risk Classifier – Claude Output Safety Check (Task 2.1k)

Geplant ist eine Risikoanalyse jedes Claude-Outputs, basierend auf:

- DSGVO-Risiko (PII, personenbezogene Infos)
- Halluzinationserkennung (nicht belegte Aussagen, Faktenfehler)
- Policy-Verletzung (z. B. politische Aussagen, Bewertungen von Personen)
- Tonalitätsanalyse (z. B. ungewollt aggressiv, falsch emotional)
- Claude-Vorgabeabweichung (Prompt wurde ignoriert)

Output des Classifiers:

```
type RiskClassification = {
 gdpr_risk: boolean
 hallucination_score: number // 0-1
 policy_violation: boolean
 tone_alert?: string // z. B. "zu werblich für Sophie"
}
```

→ Output fließt in: Audit Log, Claude Memory, Admin Dashboard

Ziel: Transparenz & Kontrolle über Claude-Prompts

Mit zunehmender Nutzung von Claude-Prompts, personalisierten Ausgaben und Al-basierten Empfehlungen wird ein dediziertes Admin-Dashboard notwendig. Dieses soll sowohl DSGVO-Konformität als auch Monitoring, Debugging und Prompt-Fehlermustererkennung ermöglichen.



💻 Admin Audit Dashboard (Task 2.1h)

Geplante Features:

Bereich	Funktion
Prompt Log Viewer	Übersicht aller gespeicherten Claude-Prompts inkl. Zeitstempel, Nutzer, Plattform, Persona
Risk Filter	Anzeige aller Einträge mit RiskClassification: gdpr_risk, hallucination score, tone alert

Kontextanzeige	Visualisierung des	Prompt-Kontextes	(task_id, session_id,
			(

forecast_data, event_trigger)

Prompt-Template-Tr

Anzeige, welches Template verwendet wurde, welche Variablen

eingesetzt wurden

Manuelles Neuauslösen von Claude mit korrigierten Daten Re-Roll Button

Export-Funktion Export als JSON, CSV, PDF (Task 2.1g)

Persona-Vergleich Vergleich von Claude-Antworten nach Persona für denselben Input

(A/B Prompting)



Claude Monitoring (Task 7.6.x – in Planung)

Ziel: Echtzeit-Überwachung der Claude-Nutzung, Kosten, Fehler und Reaktionen

Komponente	Beschreibung
Prompt Volume Tracker	Anzahl Prompts / Tag, Monat, User
Cost Monitoring	Tokenverbrauch via AWS Bedrock pro Prompt, pro Modul
Error Rate Monitor	Anteil fehlerhafter oder abgelehnter Claude-Antworten
Top Templates	Ranking der meistverwendeten Claude-Templates

Persona-Impact Viewer

Zeigt Wirkung von persona_adaptive auf Clickrate & Handlungsempfehlung

→ Langfristig kombinierbar mit Segmentierung aus VCResult (z. B. welche Empfehlungen wurden angenommen)



Debugging Tools & Log Hooks

Prompt History Viewer

- Zugriff auf alle vergangenen Prompts pro Lead / Session
- Kombination mit memory_manager: Was wusste Claude zu diesem Zeitpunkt?

Prompt Audit Export (Task 2.1g)

- Format: PDF und JSON
- Inhalt: Prompt, Response, Meta-Daten, Klassifizierung, Warnhinweise, Risikoscore
- Zielgruppe: Admins, Datenschutzbeauftragte, Investoren

Risk Alert Hook

- Bei kritischen Outputs (hallucination score > 0.8 oder policy violation = true)
- → E-Mail + UI-Benachrichtigung an Admins

Weitere geplante Tools (Fortsetzung)

Tool **Zweck Status**

prompt_replay.ts

Claude mit exakt gleichem Prompt + Input erneut aufrufen (Debugging)



<pre>risk_policy_tester.t s</pre>	Neue Prompt-Policies testen auf DSGVO-Konformität + Claude-Reaktion	Offen
tone_feedback_collec tor	User-Feedback zu Claude-Tonalität sammeln und speichern	Offen
persona_influence_tr acer	Analysiert, wie stark Persona den Claude-Output beeinflusst	Offen
<pre>prompt_policy_checke r.ts</pre>	Prüft Prompt vor Laufzeit gegen Policy-Regeln (intern + DSGVO)	 Geplant

Claude Memory Layer & Forgetting Strategy

- Effiziente Nutzung von Claude-Kontext über Sessions hinweg
- Wiedererkennung von Leads / Nutzern / Tasks
- Optionale Vergessensstrategie bei GDPR-Sensitivität
- Historienanalyse zur Verbesserung von Prompts & UX

memory_manager – Funktion & Struktur

Der memory_manager dient als lokaler Kontext-Cache, gespeist durch alle Interaktionen mit Claude oder internen Tasks.

* Hauptfunktionen:

Methode Zweck

addConversationEntry Speichert Prompt, Response, Persona, Task, Session

Zuordnung von Claude-Ausgaben zu internen Tasks addTaskEntry optimizeMemory() Entfernt irrelevante, alte oder konflikthafte Einträge Löscht gesamten Verlauf zu einer Session (z. B. forgetSession(session DSGVO-Löschung) _id) Baut Prompt-Kontext aus allen relevanten Memory-Daten loadMemoryContext()

Memory-Datenstruktur

```
type MemoryContext = {
 session_id: string
 task_id?: string
 timestamp: string
 persona: PersonalD
 platform?: string
 prompt: string
 response: string
 risk_classification?: RiskClassification
 forecast_snapshot?: Forecast[]
}
```

[→] Diese Daten werden beim Claude-Prompt (optional) beigelegt, um Kontext-awareness zu ermöglichen.

🧽 Forgetting Strategy – Datenschutzorientiert

Matbakh plant eine dynamische "Forgetting Engine", die je nach:

- Rolle des Users (Lead, Admin, Test)
- **DSGVO-Feldstatus** (personenbezogen, nicht-personenbezogen)
- Speicherzeitraum (TTL 30/90/180 Tage)
- Logging Consent

entscheidet, ob und wann ein Prompt oder eine Session aus dem Memory entfernt werden

Geplante Kontrollmechanismen:

Mechanismus	Status
TTL-basierte Auto-Löschung	Teilweise
DSGVO-opt-in Policy Erkennung	Geplant
forceForget() API Endpoint	Offen
Persona-Schutzregel (z. B. keine Sophie-Logs speichern)	Offen

Memory als Prompt-Modifikator

Künftige Claude-Aufrufe (z. B. via vc_generator) können automatisch mit Memory-Daten angereichert werden:

- Letzter Score-Trend
- Vorherige Empfehlung
- Letzte Plattform-Priorität

[→] Damit kann Claude **stringente**, **konsistente** Empfehlungen geben ("Beim letzten Mal empfahl ich...").

Neitere geplante Tools

Tool	Zweck	Status
prompt_replay.ts	Claude mit exakt gleichem Prompt + Input erneut aufrufen (Debugging)	Geplant
<pre>risk_policy_tester.t s</pre>	Neue Prompt-Policies testen auf DSGVO-Konformität + Claude-Reaktion	Offen
tone_feedback_collec tor	User-Feedback zu Claude-Tonalität sammeln und speichern	Offen
persona_influence_tr acer	Analysiert, wie stark Persona den Claude-Output beeinflusst	Offen
<pre>prompt_policy_checke r.ts</pre>	Prüft Prompt vor Laufzeit gegen Policy-Regeln (intern + DSGVO)	 Geplant

Claude Memory Layer & Forgetting Strategy

- Effiziente Nutzung von Claude-Kontext über Sessions hinweg
- Wiedererkennung von Leads / Nutzern / Tasks
- Optionale Vergessensstrategie bei GDPR-Sensitivität
- Historienanalyse zur Verbesserung von Prompts & UX

memory_manager – Funktion & Struktur

Der memory_manager dient als lokaler Kontext-Cache, gespeist durch alle Interaktionen mit Claude oder internen Tasks.

* Hauptfunktionen:

Methode	Zweck	
addConversationEntry	Speichert Prompt, Response, Persona, Task, Session	
addTaskEntry	Zuordnung von Claude-Ausgaben zu internen Tasks	
optimizeMemory()	Entfernt irrelevante, alte oder konflikthafte Einträge	
<pre>forgetSession(session _id)</pre>	Löscht gesamten Verlauf zu einer Session (z. B. DSGVO-Löschung)	
<pre>loadMemoryContext()</pre>	Baut Prompt-Kontext aus allen relevanten Memory-Daten auf	

Memory-Datenstruktur

type MemoryContext = {

session_id: string

task_id?: string

timestamp: string

persona: PersonalD

platform?: string

prompt: string

response: string

risk_classification?: RiskClassification

forecast_snapshot?: Forecast[]

ightarrow Diese Daten werden beim Claude-Prompt (optional) beigelegt, um **Kontext-awareness** zu ermöglichen.

🧽 Forgetting Strategy – Datenschutzorientiert

Matbakh plant eine dynamische "Forgetting Engine", die je nach:

- Rolle des Users (Lead, Admin, Test)
- **DSGVO-Feldstatus** (personenbezogen, nicht-personenbezogen)
- Speicherzeitraum (TTL 30/90/180 Tage)
- Logging Consent

entscheidet, ob und wann ein Prompt oder eine Session **aus dem Memory entfernt** werden soll.

Geplante Kontrollmechanismen:

Mechanismus	Status
TTL-basierte Auto-Löschung	
DSGVO-opt-in Policy Erkennung	Geplant
forceForget() API Endpoint	Offen
Persona-Schutzregel (z. B. keine Sophie- speichern)	Logs Offen

Memory als Prompt-Modifikator

Künftige Claude-Aufrufe (z. B. via vc_generator) können automatisch mit Memory-Daten angereichert werden:

- Letzter Score-Trend
- Vorherige Empfehlung
- Letzte Plattform-Priorität
- → Damit kann Claude **stringente**, **konsistente** Empfehlungen geben ("Beim letzten Mal empfahl ich...").



Teil 26 - Claude Template System: Versionierung, Hashing & Template Manager

→ Technisches Backbone für Sicherheit, Transparenz & Wiederverwendbarkeit der Claude-Vorlagen (inkl. Task 4.1 abgeschlossen)

You said:

weiter bitte

ChatGPT said:

Übergabebericht − Matbakh.app

Teil 26 von X - Claude Template System: Versionierung, Hashing & Template Manager

Empfänger: cChat (neuer Product Owner)

🧩 Ziel: Sichere, versionierbare Prompt-Verwaltung

Claude-Prompts sind nicht statisch, sondern müssen regelmäßig:

- aktualisiert (z. B. neue Variablen, neue Logik),
- verifiziert (z. B. durch Hash-Werte, Template-Gültigkeit)
- und rückverfolgbar (z. B. welche Version erzeugte welchen Output)

bereitgestellt werden.

Deshalb wurde ein vollständiges Template Management System umgesetzt (V Task 4.1 abgeschlossen).

Template-Verwaltung: Struktur

Alle Templates liegen unter:

.kiro/templates/claude/

Beispiel:

vc_core.txt

swot_analysis.txt

platform_strategy.txt

...

Dazu gehört eine **Metadaten-Tabelle** in Supabase:

Feld	Тур	Beschreibung
id	UUID	interne ID
template_na me	string	z.B. "vc_core.txt"
version	string	Semantic Versioning, z. B. "v1.4.2"
sha256_hash	string	Vollständiger Hash der Textdatei

created_at timestamp Upload-Zeitpunkt

created_by string z.B. "ci-bot" oder

"admin-user-1"

status enum "active", "deprecated",

"test"

notes text Changelog, Meta, Verwendung

→ Jeder Claude-Prompt enthält bei Ausführung einen Verweis auf diese Version (→ Prompt-Audit Traceability)

X Template Verification Engine

Jede Template-Nutzung wird:

- 1. gehasht (SHA-256)
- 2. mit der Supabase-Version verglichen
- 3. ggf. abgelehnt, wenn:
- Template als **deprecated** markiert
- Template nicht registriert
- Hash abweicht (→ tampered template alert)

Claude Prompt Flow mit Template Check

- 1. const template = loadTemplate("vc_core.txt")
- 2. const hash = sha256(template)
- 3. checkTemplateInDB("vc_core.txt", hash) // → version, status
- 4. if (!allowed) → throw PromptSecurityError
- 5. buildFinalPrompt(template, dynamicInputs)
- 6. sendToClaude(prompt)

→ Dieser Ablauf ist vollständig implementiert und Cl-getestet.

Prompt-Testabdeckung (Task 7.4.1–3)

Für alle Templates wurden:

- ▼ template-version-manager.test.ts
- ✓ prompt-validation.test.ts
- ✓ mock-template-data.ts
- ✓ uuid-mocking.ts
- ✓ send.ts Mock

implementiert.

Beispielhafte Tests:

- Template korrekt geladen
- Hash stimmt
- Prompt enthält alle Variablen
- Persona-Switch greift korrekt
- PII wird geschwärzt (geplant via Redaction)

Tooling für Admins & Developer

Tool	Beschreibung	Status
template_upload.ts	Automatisierter Upload inkl. Hash & Metadaten	✓
template_diff.ts	CLI-Tool zur Differenz zweier Versionen	(in dev)

```
Zeigt Historie aller Ausführungen eines
                                            (in
template_audit.ts
                Templates
                                            Planung)
Offen
                Template erzeugt wurden
t.ts
```

🧭 Ziel: Claude-Ergebnisse visuell und zielgruppenspezifisch darstellen

Claude liefert strukturierte Empfehlungen (Plattform, Content, Strategie), die im UI personalisierbar, kontextsensitiv und aktionsfördernd abgebildet werden sollen.

Claude Output Typen (Struktur – VCResult)

```
type VCResult = {
 summary: string
 platform_recommendations: PlatformTip[]
 strategic_analysis: StrategyBlock[]
 actions: CTAAction[]
 score_change?: ScoreChangeForecast
 persona_style: PersonaUIStyle
}
```

Diese Felder sind in die Claude-Promptantworten eingebettet und werden im Frontend über UI-Komponenten gerendert.



Claude Output	UI-Komponente	Beschreibung
summary	<resultsummary /></resultsummary 	Kurze Zusammenfassung in Persona-Sprache
platform_recommendati ons[]	<platformtips></platformtips>	Plattform-Karten mit CTA & Icon
strategic_analysis[]	<strategyblockli st /></strategyblockli 	SWOT, Balanced Scorecard, etc.
actions[]	<ctawidgets></ctawidgets>	Visuelle Handlungsaufforderungen (Cards)
score_change	<forecastchart></forecastchart>	Prognosekurve mit Trendtext
persona_style	UI Style Tokens	z.B. Farbwahl, Emoji, Tonalität

[→] Alle Komponenten reagieren dynamisch auf Claude-Ausgabe + Persona-Kontext.

Persona-Adaptive UI (Beispiel)

Person a	Farbe	Emoji	Sprachstil
Anna	Grün / Beige	经计	achtsam, einladend
Sophie	Pink / Schwarz	*	hype-driven, kurz & knackig

Markus Blau / Grau rational, faktenbasiert

Jürgen Braun / Gold gemütlich, klassisch

Die Zuordnung erfolgt über:
import { usePersonaStyle } from '@/hooks/usePersonaStyle'

const style = usePersonaStyle('anna')

// → liefert Farbpalette, Schriftstil, Emoji-Set, CTA-Texte

ightarrow Diese Daten kommen aus src/config/personas.ts

Beispiel-Komponenten (technisch)

<VCResult>

- <ResultSummary text={vcResult.summary} style={personaStyle} />
- <PlatformTips tips={vcResult.platform_recommendations} />
- <StrategyBlockList blocks={vcResult.strategic_analysis} />
- <CTAWidgets actions={vcResult.actions} persona={personaStyle} />
- <ForecastChart data={vcResult.score_change} />
- </VCResult>
- → Das gesamte VCResult-System ist modular aufgebaut und kann leicht erweitert werden.

Claude UI-Antwort testen

Um neue Outputs zu testen:

- Lokale Claude-Mockdatei (vc_result.mock.ts)
- 2. ForecastDemo.tsx oder VCPreview.tsx öffnen
- 3. const result = mockVCResult("sophie") einfügen
- 4. Ergebnis in dev-Umgebung prüfen (localhost:5173)

Test-Strategie (TASK 7.4.3)

Bereich	Status	Beschreibung
Widget Mapping	✓ Abgedeckt	vcresult.test.tsx + Snapshots
Persona Styles	✓ Abgedeckt	personaStyle.test.ts
Forecast Chart Logik	✓ Abgedeckt	forecastChart.test.tsx
Claude-Mocks	✓ Vorbereitet	mocks/vc_result.mock .ts

Ziel: Adaptive Analyse und Darstellung für unterschiedliche Nutzerpersönlichkeiten

Das Persona-System in Matbakh dient zwei Hauptzwecken:

- 1. Anpassung der Claude-Ausgabe (Tonalität, Fokus, Stil)
- 2. Anpassung der UI-Komponenten (Farben, Emojis, Texte, Empfehlungen)



Person a	Kurzprofil	Plattformfokus	Output-Stil
Anna	Gesundheitsbewusste Genießerin	Google, Instagram	achtsam, ruhig, visuell
Markus	Familienvater, Planer	Google, Facebook	rational, strukturiert
Sophie	Trendaffine Studentin	TikTok, Spotify, Insta	hype-driven, jung, mit Emojis
Jürgen	Klassischer Best-Ager	Google Maps, WhatsApp	verbindlich, persönlich

Technische Persona-Zuweisung

Die Persona wird aktuell automatisch vergeben auf Basis von:

• Standortdaten (z. B. urbane vs. vorstädtische Umgebung)

→ Siehe auch: src/config/personas.ts (Mapping, Farben, Emojis)

- Plattform-Aktivitäten (z. B. mehr IG als FB)
- Keywords in der Google-Beschreibung (z. B. "Bio", "Vegan", "Familienbetrieb")
- Bewertungs-Tonalität (z. B. "hip", "gemütlich", "authentisch")
- → Diese Regeln liegen in src/lib/personaMatcher.ts

Beispiel-Zuordnung (Pseudo-Logik)

```
if (bio.includes('bio') || tags.includes('achtsam')) return 'anna'; if (tags.includes('trend') || tiktok > 0) return 'sophie'; if (kidsCorner && googleRating > 4.2) return 'markus'; return 'jürgen';
```

🧩 Persona-Nutzung in Claude

Beim Claude-Prompt-Aufruf wird die Persona-ID mitgegeben: { ...promptInput, persona: 'sophie'

→ Claude wählt damit aus persona_adaptive.txt den passenden Sprachstil

}

🎨 Persona-Nutzung in Ul

Die Hook usePersonaStyle() lädt für jede Komponente:

- Farbpalette (Primary, Secondary)
- Emojis (z. B. ¼ vs. ♂)
- Sprachstil (z. B. "Probier es aus" vs. "Empfehlung")
- CTA-Formulierung (z. B. Button-Text, Headline-Ansprache)

Beispiel:

```
const style = usePersonaStyle('markus')
const cta = style.cta("plattform_tipp")
```

→ Output: "Prüfen Sie jetzt Ihre Sichtbarkeit auf Google."

Persona Tests (TASK 7.4.4)

Testkomponente	Statu s	Beschreibung
personaMatcher.test.t s	V	Testet Matching-Logik anhand Beispielen
personaStyle.test.ts	V	Prüft Style-Konfiguration pro Persona
vcresult.test.tsx	V	UI-Darstellung bei personaStyles

Persona Switching (in Planung)

Ziel: Nutzer:innen können ihre Persona künftig selbst wählen oder ändern, z. B. via:

- Dropdown im Onboarding
- Button auf dem Dashboard: "Wie möchtest du angesprochen werden?"
- Künstliche Auswahl auf Basis der bevorzugten Plattformen oder Interaktionsstile
- → Claude & UI reagieren dann dynamisch

M Offene Punkte

Punkt	Status
Persona-A/B-Test mit echten Leads	InPlanung
Dynamische Persona-Erkennung live	Beta
Claude persona_adaptive.txt	In Arbeit

@ Ziel:

Differenzierte Claude-Outputs abhängig von der Zielpersona – sowohl für **B2C** (**Endnutzer:innen**) als auch **B2B** (**Gastrobetriebe**). Die aktuelle Implementierung stützt sich v. a. auf die B2C-Testphase, während B2B-Personas strategisch in der Investoren-Validierung und Sichtbarkeitsberatung definiert wurden.

B2C Personas (aktuell live im System)

Quelle: src/config/personas.ts

Status: Aktiv, in VCResult-Ausgabe eingebaut (adaptive Empfehlungen, UI-Elemente,

Sprachstil)

Person a	Profilbeschreibung	Stil der Claude-Ausgabe	Plattformfokus
Anna	Gesundheitsbewusste Genießerin, 32, Yoga, Bio	Achtsam, emotional, Storytelling	Instagram, Google
Marku s	Familienvater, 45, organisiert, lokalorientiert	Praktisch, planbar, faktenbasiert	Google, Facebook
Sophie	Studentin, 24, trendaffin, Streetfood, TikTok	Hype-driven, emojis, TikTok-Sprache	TikTok, Spotify, IG
Jürgen	Best Ager, 58, kochinteressiert, offline-stark	Persönlich, ruhig, textfokussiert	Google Maps, E-Mail

Diese Personas wurden für das Matching von Lokalpräferenzen, UI-Elementen (CTAs, Texte) und Claude-Ausgabeformaten genutzt.

B2B Personas (nur konzeptionell definiert)

Quelle: Validierungsstrategie & Investorenunterlagen (Juni–Juli 2025)

Status: Noch nicht im Code oder Claude Prompt verankert → Umsetzung in Phase 3 geplant

Persona	Profilbeschreibung	Sichtbarkeits-Pain Points	Geplanter Claude-Stil
Sabine	Betreibt kleines	Google nicht	Fachlich,
(Gastro-Inhabe	Premium-Bistro, keine Zeit	gepflegt, kein	pragmatisch, visuell
rin)	für Onlinepflege	Reels-Content	gestützt
Thomas	Verantwortlich für 12	Plattform-Scoring,	Bulletpoint-Strategie
(Franchise-Leit	Standorte, Fokus:	ROI-getriebene	n, Scores,
ung)	Reporting & Effizienz	Sicht	Benchmarks
Miriam	Kreative Location, aber kein Instagram-Konzept	Sichtbar, aber	Szenarien,
(Event-Vermiet		Zielgruppe erreicht	Empfehlungen,
ung)		sie nicht	Story-Taktik

Diese B2B-Personas sollen später in VCResult, Claude Prompts, Forecast-Modulen und Benchmarks getrennt behandelt werden. Eine persona_business.ts Datei ist vorgesehen.

✓ Persona A/B Testing Strategie (in Planung)

Ziel: Effektivität von Claude-Ausgaben pro Persona evaluieren (Clickrate, Handlung, Feedback)

Konkrete Ansatzpunkte:

- Prompt-Variation pro Persona (z. B. emotional vs. rational)
- UI-A/B-Tests: Unterschiedliche Empfehlungen, CTA-Sprache
- Erfolgsmessung über:
 - o Click-Through-Rates
 - Conversion bei Sichtbarkeitsaktionen
 - o Follow-Up Bookings / In-App Events

Segmentierungslogik kann automatisiert über Google-Daten, Standorttyp, oder manuelle Klassifikation erfolgen.

Prompt Template Infrastruktur für Persona-Variation

Ordner: .kiro/templates/claude/persona_adaptive.txt (Work-in-Progress)

Geplant:

{{#if persona.anime}}

Empfehle auf TikTok trendende Inhalte mit Spaßfaktor.

{{else if persona.business}}

Liste konkrete Maßnahmen mit Aufwand/Nutzen-Verhältnis.

{{/if}}

Zusammenführung der Systeme geplant

- → persona_profiles Supabase-Tabelle enthält Mapping B2C / B2B
- ightarrow Claude soll künftig automatisch unterscheiden, ob Endnutzer:in oder Betrieb angesprochen wird
- → Prompt-Auswahl + UI + Plattformstrategie folgen dann dynamisch der Persona

To Do für dich als neuer PO (cChat):

- [] persona_adaptive.txt finalisieren und testen
- [] B2B-Persona-Logik implementieren (config + Claude Prompting)
- [] UI-Komponenten (CTA, Sprache) adaptiv gestalten
- [] A/B Testplanung finalisieren (Messkriterien definieren)

Region 2 Page 1

Sichere, auditierbare und versionierbare Verwaltung aller Claude-Prompts – mit Schutz vor unautorisierten Änderungen, DSGVO-konformer Dokumentation und granularen Zugriffsregeln.

Template Management System

Pfad: .kiro/templates/claude/

Verwaltet über: template-version-manager.ts

Features:

Funktion	Statu s	Beschreibung
SHA256-Hash jeder Version	V	Jede gespeicherte Prompt-Vorlage wird automatisch gehasht
Versionierung pro Datei	V	Jede Änderung erstellt eine neue Version mit Timestamp und Meta-Info
Signierung mit KMS-Key	V	Signatur via AWS KMS (Key: matbakh-template-key) zur Integritätsprüfung
Policy-Tags pro Template	V	Angabe: "DSGVO-relevant", "personabezogen", "risk_high", etc.
Template-Prüfung beim Prompt	V	Claude-Request nur erlaubt, wenn geprüfte & signierte Version vorliegt

Beispiel Eintrag in template_versions (Supabase):

```
"template_name": "vc_core.txt",

"version": "3",

"hash": "ed1b2c8...",

"signed": true,

"signed_by": "system",

"created_at": "2025-08-23T13:24:00Z",

"policy": "ds_relevant;persona_dependent"
}
```

{

}

Representation of the contract of the contract

Ziel: Sicherheit + Nachvollziehbarkeit, v. a. bei sensiblen Prompts (Forecast, Persona, Risikoanalysen)

Komponenten:

Modul Aufgabe Prüft vor jedem Claude-Aufruf: Ist Template gültig, signiert, prompt_policy_enforce policykonform? r.ts Loggt alle signierten Prompts + Antworten inkl. Policy-Tag audit_logger.ts Erstellt automatisch Policies pro Template + promptPolicyGenerator Claude-Nutzungstyp .ts Policies (Beispiele): { "vc_core.txt": { "requires_signature": true, "allowed_roles": ["admin", "system"], "log_to_audit": true, "pseudonymize_output": true, "auto_redact_fields": ["email", "phone", "address"] }

Test Coverage

Testdatei: template-version-manager.test.ts

Status: Alle 11 Tests erfolgreich (siehe Teil 21)

Geprüft werden:

- Signaturprüfung
- Änderungsprotokoll
- Redaktionsregeln bei PII
- Hash-Matching bei Claude-Aufruf
- Sicherheitsabbruch bei fehlender Policy-Freigabe

Zusammenspiel mit Audit-System

- ullet Jeder Claude-Call o Verweis auf Template-Version
- Alle Versionen rückverfolgbar über template_versions
- DSGVO-konforme Prompt-Ablage + Löschfristen via TTL (geplant)

To Do für dich als PO (cChat):

- [] Rollout aller Claude Templates in Versionierungspflicht aufnehmen
- [] prompt_policy_enforcer.ts in Claude Lambda vollständig integrieren
- [] Prompts ohne Policy (z. B. forecast_interpretation.txt) ergänzen
- [] UI-Audit-Dashboard um Version-Verfolgung erweitern (siehe Teil 24)