|  |  |  |
| --- | --- | --- |
|  | Architecture Concept Document   |  | | --- | | Lernplattform |   15.02.24  Gruppe 9  Ben Strobel, Kevin Knops |

Inhaltsverzeichnis

[1 Introduction 3](#_Toc190866697)

[2 Architecture Input 4](#_Toc190866698)

[2.1 Business Goals 4](#_Toc190866699)

2.1.1 Establishing a Hybrid Monetization Model 4

2.1.2 Scaling the User Base with a Focus on Conversion 4

2.1.3 Building an Ecosystem with Value-Added Services 5

2.1.4 Achieving Long-Term Market Dominance through Network Effects 5

[2.2 Requirements 6](#_Toc190866700)

[2.2.1 Functional Requirements 6](#_Toc190866701)

[2.2.2 Non-funtional Requirements 7](#_Toc190866702)

[2.3 Scope, Assumptions and Boundary Conditions 8](#_Toc190866704)

2.3.1 Scope 8

2.3.2 Assumptions and Boundary Conditions 8

[2.4 Architecture Drivers 9](#_Toc190866705)

[2.4.1 Scalability 9](#_Toc190866706)

[2.4.2 Security 9](#_Toc190866707)

[2.4.3 Ease of Use 9](#_Toc190866708)

[2.4.4 Integration 9](#_Toc190866709)

[3 Architecture Overview 10](#_Toc190866710)

3.1 Architecture Concept 10

3.2 Systemoverview & Diagramm 11

[4 Architecture Decisions 14](#_Toc190866711)

[4.1 Major Decisions, Challenges and Risks 14](#_Toc190866712)

[4.1.1 Technology Decisions 14](#_Toc190866713)

[4.1.2 Major Architecture Decisions 15](#_Toc190866714)

[4.1.3 Risks 15](#_Toc190866715)

[4.2 Security Concept 16](#_Toc190866716)

[4.2.1 Protection against attacks 16](#_Toc190866717)

[4.2.2 Data Privacy Protection 17](#_Toc190866718)

[5 Summary 18](#_Toc190866725)

# Introduction

Das Projekt ist eine moderne, cloudbasierte Online-Lernplattform (Ähnlich wie „Skillshare“), die es Nutzern ermöglicht, digitale Kurse in verschiedenen Fachbereichen wie Design, Business, Technologie, Fotografie, Marketing und vielen weiteren Themen zu erstellen, zu belegen und miteinander zu interagieren. Die Plattform richtet sich sowohl an Einsteiger als auch an Experten und bietet Lehrenden die Möglichkeit, ihr Wissen in Form von strukturierten Videokursen, interaktiven Aufgaben und begleitenden Materialien weiterzugeben. Lernende können flexibel und ortsunabhängig auf die Inhalte zugreifen und durch unterschiedliche Funktionen ihren individuellen Lernfortschritt verfolgen.

Neben der klassischen Kursbereitstellung legt die Plattform großen Wert auf Community-Interaktion. Nutzer können sich untereinander austauschen, Fragen zu Kursinhalten stellen und über eine integrierte Kommentar- und Bewertungsfunktion mit Lehrenden in Kontakt treten. Durch ein Empfehlungssystem werden ihnen basierend auf ihren Interessen und bisherigen Aktivitäten personalisierte Kursvorschläge gemacht.

Um Lehrenden eine professionelle Umgebung für ihre Inhalte zu bieten, unterstützt die Plattform verschiedene Monetarisierungsmodelle. Neben kostenlosen Kursen können Premium-Kurse angeboten werden, die entweder über Einmalzahlungen oder Abonnement-Modelle zugänglich sind. Die Integration von Zahlungsanbietern wie Stripe oder PayPal stellt eine sichere Abwicklung von Kurskäufen und Abonnements sicher.

Die Plattform ist für eine nahtlose Nutzung auf verschiedenen Endgeräten optimiert. Sie bietet eine Progressive Web App (PWA), die sowohl auf Desktop-Computern als auch auf Tablets und Smartphones eine optimale Benutzererfahrung gewährleistet. Durch eine intuitive Benutzeroberfläche mit klarem Design können sowohl Lehrende als auch Lernende effizient navigieren, Kurse entdecken und verwalten. Der Einsatz moderner Technologien wie Microservices, Containerisierung und Cloud-Hosting stellt sicher, dass die Plattform mit steigender Nutzerzahl problemlos erweitert werden kann.

Die Architektur des Systems wird in diesem Dokument detailliert analysiert, wobei insbesondere folgende Aspekte betrachtet werden:

* **Architektonische Treiber und Anforderungen**: Identifikation der geschäftlichen und technischen Anforderungen, die maßgeblich die Architektur beeinflussen.
* **Architekturübersicht**: Beschreibung der Gesamtstruktur der Plattform, einschließlich der eingesetzten Technologien und Systemkomponenten.
* **Architektonische Entscheidungen**: Dokumentation der zentralen Architekturentscheidungen, einschließlich der Designprinzipien, Alternativen und technischer Herausforderungen.
* **Sicherheitskonzept**: Analyse der Schutzmaßnahmen zur Sicherstellung der Plattformintegrität und zum Schutz sensibler Nutzerdaten gemäß DSGVO.
* **Deployment und Betrieb**: Strategien zur Bereitstellung, Skalierung und Wartung der Plattform, unter anderem durch den Einsatz von Container-Technologien und Cloud-Diensten.
* **Testing und Qualitätssicherung**: Konzepte zur Testautomatisierung sowie Continuous Integration und Deployment (CI/CD), um eine hohe Softwarequalität und Verfügbarkeit sicherzustellen.

# Architecture Input

## 2.1 Business Goals

**2.1.1 Establishing a Hybrid Monetization Model**

**Ziel:** Kombination aus Abonnements, Transaktionsgebühren und Partnerprogrammen zur nachhaltigen Maximierung der Umsatzströme und Schaffung eines attraktiven Angebots für Lernende und Creator.

**Freemium-Subscription-Modell:**

* Nutzer zahlen monatlich oder jährlich für exklusive Inhalte, offizielle Zertifikate, Offline-Zugriff und eine werbefreie Nutzung. Dies ermöglicht ihnen einen qualitativ hochwertigen Lernprozess mit erweiterten Funktionen.
* Lernende erhalten kostenfreien Zugang zu einer begrenzten Auswahl an Kursen und Community-Funktionen. Dies erleichtert den Einstieg und erhöht die Nutzerakzeptanz, indem Interessierte zunächst ohne finanzielles Risiko teilnehmen können.

**Transaktionsbasierte Monetarisierung für Creator:**

* Plattformgebühren von 15–20 % auf Kursverkäufe oder einmalige Zahlungen für Premium-Inhalte (z. B. spezialisierte Masterclasses). Dies ermöglicht Creatorn eine faire Vergütung und finanziellen Anreiz zur Erstellung hochwertiger Inhalte.
* Optionale Gebühren für "Pay-per-View"-Live-Workshops oder individuelle Coaching-Sessions, die es Lernenden ermöglichen, gezielt Expertenwissen von Top-Creatorn zu erwerben.

**2.1.2** **Scaling the User Base with a Focus on Conversion**

**Ziel:** Steigerung der aktiven Nutzer (Learner und Creator) und Erhöhung der Premium- Conversion-Rate.

**Virale Akquisition durch Creator-Netzwerke:**

* Anreize für Creator, eigene Communities (z. B. Social-Media-Follower) auf die Plattform zu bringen (Bonus bei Neuanmeldungen).
* „Refer-a-Friend“-Programme mit Gratismonaten für erfolgreiche Weiterempfehlungen.

**Gezieltes Onboarding für Creator:**

* Bereitstellung von intuitiven Kurserstellungstools (Templates, interaktive Elemente, Analytics), um den Einstieg für neue Creator zu erleichtern und qualitativ hochwertige Inhalte zu fördern.
* Dynamisches Revenue-Share-Modell, bei dem Top-Creator durch höhere Provisionen und exklusive Vermarktungsmöglichkeiten belohnt werden, um ihre langfristige Bindung an die Plattform zu stärken.

**2.1.3 Building an Ecosystem with Value-Added Services**

**Ziel:** Erweiterung des Geschäftsmodells durch zusätzliche Mehrwertdienste, die die Lernplattform über reinen Kursverkauf hinaus attraktiv machen.

**Zertifizierungsprogramme:**

* Einführung kostenpflichtiger offizieller Zertifikate, die in Zusammenarbeit mit Unternehmen und Bildungseinrichtungen entwickelt werden (z. B. LinkedIn-Integration). Lernende profitieren von einem anerkannten Nachweis ihrer Qualifikationen.
* Möglichkeit zur Branchenzertifizierung, um Karrierechancen der Nutzer zu verbessern und das Vertrauen von Arbeitgebern in die Kursinhalte zu stärken.

**Premium-Support & Beratung:**

* Erweiterte Dienstleistungen wie individuelle Coachings, Karriereberatung oder CV-Checks durch Experten, die Lernenden gezielte Unterstützung bei ihrer beruflichen Weiterentwicklung bieten.
* Unternehmen können kostenpflichtige Team-Lizenzen und Weiterbildungspakete erwerben, um ihren Mitarbeitern gezielte Fortbildungsmöglichkeiten zu bieten.

**2.1.4 Achieving Long-Term Market Dominance through Network Effects**

**Ziel:** Positionierung als führende Plattform durch Community-Wachstum, steigende Interaktionsraten und eine globale Expansion mit regionaler Anpassung.

**Globale Skalierung & Lokalisierung:**

* Erweiterung des Angebots durch Übersetzungen und regionale Inhalte in Wachstumsmärkten (z. B. Spanisch, Hindi, Portugiesisch). Dies erleichtert den Zugang für neue Nutzergruppen.
* Zusammenarbeit mit lokalen Creatorn und Experten, um spezifische Bildungsbedarfe einzelner Märkte gezielt zu bedienen.

**Branchenspezifische Vertikalisierung:**

* Fokussierung auf stark nachgefragte Themenbereiche, z. B. Tech-Skills (Softwareentwicklung, KI) oder Unternehmertum, um gezielt neue Zielgruppen anzusprechen.
* Ausbau eines intelligenten Empfehlungssystems auf Basis von KI, das Nutzer mit den relevantesten Kursen verbindet und so die Engagement-Raten erhöht.

## 2.2 Requirements

Die Anforderungen an die Skillshare-Plattform lassen sich in funktionale und nicht-funktionale Anforderungen unterteilen. Diese Anforderungen haben direkte Auswirkungen auf die Architektur, insbesondere in Bezug auf Skalierbarkeit, Sicherheit, Performance und Integration mit Drittanbieterdiensten.

### 2.2.1 Functional Requirements

Die funktionalen Anforderungen beschreiben die zentralen Features und Funktionalitäten der Plattform, die durch die Architektur unterstützt werden müssen:

1. **Kurserstellung und -verwaltung**
   * Lehrende müssen in der Lage sein, Kurse mit strukturierten Inhalten (Videos, Texte, Aufgaben) zu erstellen und zu verwalten.
   * Unterstützung für unterschiedliche Kursformate, einschließlich On-Demand-Videos und Live-Sessions.
   * Versionierung von Kursinhalten, um Aktualisierungen und Erweiterungen zu ermöglichen.
   * Anbindung an ein Content Delivery Network (CDN) zur effizienten Bereitstellung von Videoinhalten.
2. **Kursbelegung und Nutzerverwaltung**
   * Lernende können Kurse über eine intuitive Benutzeroberfläche durchsuchen, belegen und ihre Fortschritte (Levelsysteme, Zertifikate nach Abschluss) nachverfolgen.
   * Implementierung eines Rollen- und Rechtemanagements zur Unterscheidung zwischen Lehrenden und Lernenden.
   * Unterstützung für verschiedene Anmelde- und Authentifizierungsmethoden (z. B. OAuth, SSO, Social Logins).
3. **Interaktion und Community-Funktionen**
   * Ermöglichung von Diskussionen innerhalb der Kurse durch Kommentare und Foren.
   * Implementierung eines Bewertungssystems, um die Qualität von Kursen und Lehrenden sicherzustellen.
   * Benachrichtigungen und E-Mail-Integrationen für Kursupdates und Community-Interaktionen (z.B. Quiz, Umfragen und Gruppenprojekte)
4. **Personalisierte Kursempfehlungen**
   * Implementierung eines Empfehlungssystems basierend auf Benutzerverhalten und Interessen.
   * Nutzung von Machine Learning oder regelbasierten Algorithmen zur Generierung relevanter Vorschläge (z.B. Erstellung individueller Lernpfade)
5. **Zahlungsabwicklung und Monetarisierung**
   * Unterstützung für verschiedene Zahlungsanbieter (z. B. Stripe, PayPal).
   * Abwicklung von einmaligen Zahlungen und abonnementbasierten Modellen (inklusive Rabattaktionen und Gutscheinen).
   * Verwaltung von Rechnungen und Zahlungsnachweisen gemäß gesetzlichen Vorschriften.

### 2.2.2 Non-functional Requirements (Product Specific Qualities)

Nicht-funktionale Anforderungen betreffen übergreifende architektonische Aspekte wie Skalierbarkeit, Sicherheit und Performance.

1. **Skalierbarkeit**
   * Die Plattform muss mit einer wachsenden Nutzerbasis und steigender (weltweiter) Kursanzahl skalieren können.
   * Unterstützung für horizontale Skalierung durch verteilte Microservices-Architektur.
   * Nutzung von Caching-Mechanismen (z. B. Redis) zur Reduzierung von Datenbankzugriffen.
2. **Sicherheit**
   * Schutz von Nutzerdaten durch Verschlüsselung (z. B. TLS für Übertragungen, AES für gespeicherte Daten).
   * Sichere Authentifizierung durch OAuth 2.0 und JWT.
   * Einhaltung von Datenschutzbestimmungen (z. B. DSGVO) durch Mechanismen wie Anonymisierung und explizite Nutzerzustimmung.
3. **Benutzerfreundlichkeit und Responsive Design**
   * Optimierung der UI/UX für verschiedene Geräte (Desktop, Tablet, Smartphone).
   * Implementierung von Barrierefreiheitsstandards (z. B. WCAG).
   * Minimierung der Ladezeiten durch asynchrones Laden von Inhalten.
   * Allgemeine Inklusion durch beispielsweise durch die Unterstützung für Sprachsynthese (Text-to-Speech) für barrierefreies Lernen.
4. **Verfügbarkeit und Ausfallsicherheit**
   * Nutzung von Cloud-Hosting mit hoher Verfügbarkeit (z. B. AWS, Azure).
   * Replikation von Datenbanken für Fehlertoleranz.
   * Implementierung von automatischen Backups und Disaster Recovery Mechanismen
5. **Low Total Cost of Ownership**
   * Optimierung der Infrastrukturkosten durch **dynamische Skalierung** mittels Kubernetes und Cloud-Services wie AWS Auto Scaling.
   * Minimierung von Datenbank- und Speicherkosten durch effiziente **Speicherverwaltung**, z. B. durch die Auslagerung älterer Inhalte in **Cold Storage**.
   * Reduzierung von Lizenzkosten durch **Einsatz von Open-Source-Technologien** (z. B. PostgreSQL, Apache Kafka), um eine langfristige Unabhängigkeit von proprietären Lösungen zu gewährleisten.
6. **Wartbarkeit und Nachvollziehbarkeit**
   * Die Architektur muss **klar strukturiert und modular aufgebaut** sein, um eine einfache Weiterentwicklung und Fehlerbehebung zu ermöglichen.
   * Alle wesentlichen Architekturentscheidungen werden **mittels Architecture Decision Records (ADR) dokumentiert**, um technische Entscheidungen nachvollziehbar zu machen.
   * Implementierung eines **Observability-Stacks** mit Logging, Monitoring und Tracing zur **Fehlersuche, Performance-Optimierung und Stabilitätsüberwachung**.

## 2.3 Scope, Assumptions and Boundary Conditions

**2.3.1 Scope**

Die Plattform dient als cloudbasierte Online-Lernumgebung, die Lehrenden ermöglicht, Kurse zu erstellen, zu verwalten und zu monetarisieren, während Lernende diese belegen, absolvieren und mit der Community interagieren können. Neben der Bereitstellung von Kursinhalten liegt der Fokus darauf, den Austausch zwischen Lehrenden und Lernenden zu fördern. Funktionen wie Kommentare, Bewertungen und Diskussionen sollen eine interaktive Lernumgebung schaffen. Zudem werden sowohl kostenfreie als auch kostenpflichtige Kurse unterstützt, wobei die Monetarisierung über Abonnements, Einzelverkäufe und transaktionsbasierte Gebühren erfolgt. In der ersten Entwicklungsphase konzentriert sich die Umsetzung auf die Webanwendung mit einer für mobile Endgeräte optimierten Benutzeroberfläche.

Nicht Bestandteil dieser Architekturbetrachtung sind spezifische Erweiterungen wie ein dediziertes Creator-Dashboard mit umfangreichen Analysefunktionen, native Mobile-Apps oder ein erweitertes Reporting-System für detaillierte Kurs- und Nutzerstatistiken. Diese Funktionen können zu einem späteren Zeitpunkt entwickelt werden, liegen jedoch außerhalb des initialen Entwicklungsumfangs. Ebenso werden Integrationen mit externen Plattformen oder Drittanbieter-Services für personalisierte Zertifikate oder Unternehmenslösungen nicht im Rahmen dieser Architektur berücksichtigt.

**2.3.2 Assumptions and Boundary conditions**

Die Plattform wird in einer Cloud-Umgebung gehostet, um eine hohe Verfügbarkeit und Skalierbarkeit zu gewährleisten. Es wird vorausgesetzt, dass die Nutzer über eine stabile Internetverbindung verfügen, um insbesondere Video- und interaktive Inhalte reibungslos abrufen zu können. Die Anwendung muss mit gängigen Webbrowsern wie Chrome, Firefox, Safari und Edge kompatibel sein und auf mobilen Endgeräten ebenso performant funktionieren wie auf Desktop-Computern. Durch den Einsatz eines Content Delivery Networks (CDN) soll eine schnelle Bereitstellung von Kursinhalten, insbesondere von Videodaten, sichergestellt werden.

Ein wesentlicher Aspekt ist die Einhaltung der Datenschutz-Grundverordnung (DSGVO) bei der Speicherung und Verarbeitung personenbezogener Daten. Dies umfasst transparente Datenschutzerklärungen, Mechanismen zur Einwilligungserklärung der Nutzer sowie die Möglichkeit zur Datenlöschung. Personenbezogene Daten werden bevorzugt innerhalb der EU gehostet, um regulatorische Anforderungen zu erfüllen. Falls zukünftig KI-gestützte Funktionen implementiert werden, die auf Nutzerdaten basieren, müssen diese datenschutzkonform verarbeitet und dokumentiert werden. Zusätzlich muss die Plattform barrierefrei gemäß den Web Content Accessibility Guidelines (WCAG) gestaltet sein, um eine inklusive Nutzung zu ermöglichen. Dazu gehören die Unterstützung von Screenreadern, alternative Texte für Bilder sowie eine kontrastreiche und intuitive Benutzeroberfläche. Um eine nachhaltige und effiziente Wartung zu ermöglichen, wird die Architektur so konzipiert, dass neue Funktionen modular ergänzt werden können, ohne bestehende Systeme maßgeblich zu beeinflussen.

## 2.5 Architecture Drivers

Die Architektur der Plattform wird von mehreren entscheidenden Faktoren beeinflusst, die maßgeblich ihre Struktur, Technologien und Designentscheidungen bestimmen. Die folgenden Treiber haben den größten Einfluss auf die Architektur und werden nach ihrer Bedeutung begründet

### 2.5.1 Scalability

Da die Plattform eine große Anzahl von Nutzern und Kursen unterstützen muss, ist eine skalierbare Architektur essenziell. Durch eine Microservices-Architektur, containerisierte Anwendungen und Cloud-basierte Lösungen wird sichergestellt, dass sich die Plattform flexibel erweitern lässt. Dies ermöglicht nicht nur eine hohe Verfügbarkeit und Performance, sondern auch Anpassungsmöglichkeiten für Entwickler, um die Software sinnvoll und effizient zu erweitern. Eine fehlende Skalierbarkeit könnte zu Leistungseinbußen oder Systemausfällen führen, insbesondere bei steigender Nutzeranzahl oder hohen Zugriffsraten auf Videoinhalte.

### 2.5.2 Security

Der Schutz von Nutzerdaten und Zahlungsinformationen ist ein zentraler Aspekt der Architektur. Die Plattform muss strenge Datenschutz- und Sicherheitsrichtlinien, wie die DSGVO und PCI-DSS, einhalten, um Missbrauch und Datenlecks zu verhindern. Dies erfordert eine sichere Authentifizierung (z. B. OAuth 2.0, JWT), Verschlüsselung (TLS, AES) und Angriffsschutzmaßnahmen (z. B. gegen SQL-Injections, XSS, DDoS-Attacken). Da die Plattform auch monetarisierte Inhalte bietet, ist die Sicherheit der Zahlungsabwicklung besonders kritisch. Ohne eine robuste Sicherheitsarchitektur könnte das Vertrauen der Nutzer erheblich beeinträchtigt werden, was direkte Auswirkungen auf die Plattformnutzung und den wirtschaftlichen Erfolg hätte.

### 2.5.3 Ease of use

Eine intuitive Benutzeroberfläche ist essenziell, um die Plattform für Lehrende und Lernende gleichermaßen zugänglich zu machen. Die Architektur muss so gestaltet sein, dass schnelle Ladezeiten, eine klare Navigation und ein durchdachtes UI/UX-Design gewährleistet sind. Eine schlechte Benutzerfreundlichkeit kann dazu führen, dass Nutzer die Plattform meiden oder ihre Kurse abbrechen. Ein responsives Design stellt sicher, dass die Plattform auf allen Geräten reibungslos funktioniert. Zusätzlich sind Barrierefreiheitsfunktionen erforderlich, um eine inklusive Nutzung zu ermöglichen.

### 2.5.4 Integration

Die Plattform muss nahtlos mit externen Zahlungssystemen und Drittanbieter-APIs interagieren können. Eine modulare Architektur mit gut definierten Schnittstellen (z. B. REST oder GraphQL) erleichtert die Integration von Diensten wie Stripe oder PayPal für sichere Transaktionen. Zusätzlich kann die Anbindung externer Lernmanagementsysteme (LMS), Zertifizierungsstellen oder KI-gestützter Empfehlungssysteme die Funktionalität der Plattform erweitern. Fehlende Integrationsmöglichkeiten könnten die Wettbewerbsfähigkeit der Plattform einschränken und die Nutzererfahrung negativ beeinflussen.

# Architecture Overview

Die Architektur von Skillshare basiert auf einer modularen Microservices-Architektur, um Skalierbarkeit, Flexibilität und Wartbarkeit zu gewährleisten. Sie folgt einem Cloud-nativen Ansatz, bei dem Services unabhängig entwickelt, bereitgestellt und skaliert werden können. Die zentrale Plattform besteht aus mehreren Schichten, die über APIs miteinander interagieren.

**3.1 Architecture Concept**

Die Architektur ist in folgende Hauptkomponenten unterteilt:

**1 Frontend**

* Entwickelt mit React.js oder Angular für eine moderne, responsive Benutzeroberfläche.
* Kommuniziert über REST- oder GraphQL-APIs mit dem Backend.
* Bietet eine Progressive Web App (PWA) für eine nahtlose mobile Nutzererfahrung.
* Implementiert Client-seitiges Caching, um die Performance zu optimieren.

**2 Backend-Microservices**

* Implementiert mit Spring Boot (Java) oder Node.js (Express.js).
* Besteht aus mehreren Microservices, die dedizierte Funktionen erfüllen (z. B. Kursverwaltung, Zahlungsabwicklung, Nutzerverwaltung).
* Nutzt Containerisierung (Docker) und Orchestrierung (Kubernetes) für Skalierbarkeit.

**3 Database & Storage**

* PostgreSQL für relationale Daten wie Nutzer- und Zahlungsinformationen.
* MongoDB für flexible und skalierbare Speicherung von Kursinhalten.
* Redis als Caching-Lösung zur Performance-Optimierung.
* AWS S3 oder Azure Blob Storage für die Speicherung und Bereitstellung von Videos.
* Content Delivery Network (CDN) zur globalen und schnellen Bereitstellung großer Medieninhalte.
* Automatische Backups und Replikation zur Datensicherung und Ausfallsicherheit.

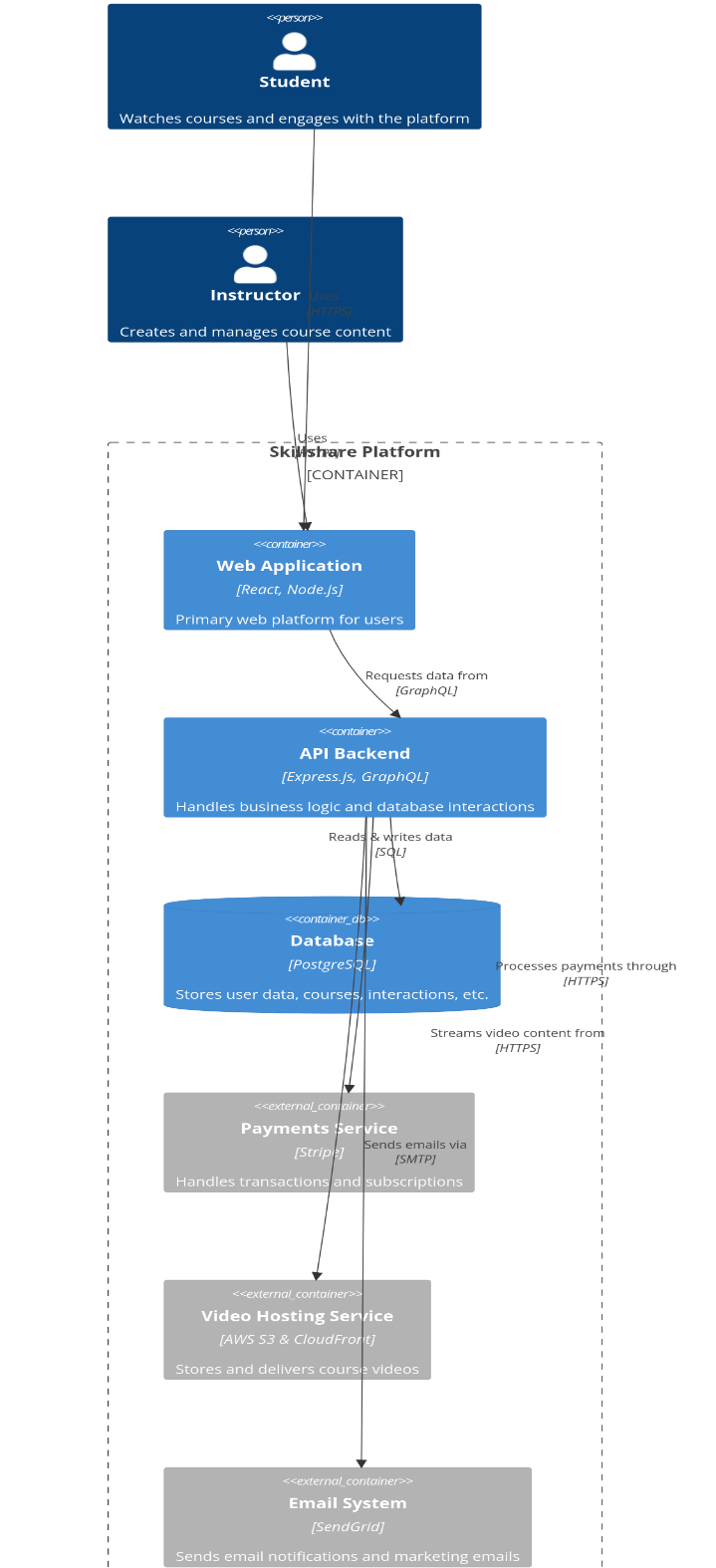
**4 APIs & External Services**

* Anbindung an Zahlungsdienstleister (Stripe, PayPal).
* Einbindung von Machine Learning-Diensten für personalisierte Kursempfehlungen.
* OAuth 2.0 & JWT für sichere Authentifizierung.

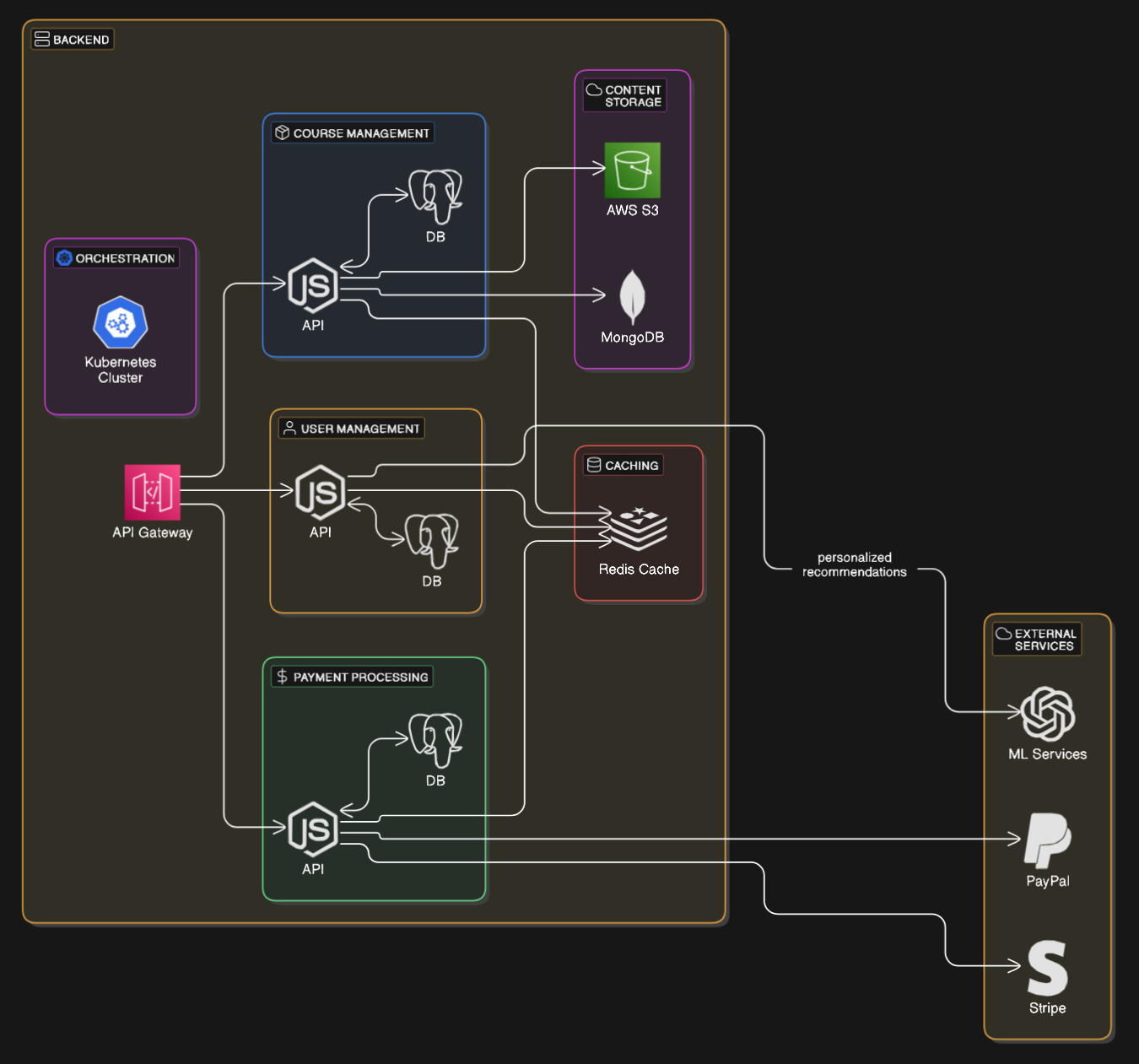
**3.2 Systemoverview & Diagramm**

Die folgende C4-Darstellung (Kontext- und Komponentenebene) zeigt die Architektur von Skillshare:

**Frontend:**



**Backend:**



Das Kontextdiagramm stellt die wichtigsten Systeminteraktionen dar:

* Nutzer (Lehrende & Lernende) greifen über das Frontend (Web & Mobile App) auf die Plattform zu.
* Die Anfragen werden über den API Gateway an die jeweiligen Microservices weitergeleitet.
* Der Authentifizierungsservice überprüft Zugriffsrechte.
* Kurs- und Nutzerdaten werden in PostgreSQL und MongoDB gespeichert.
* Videoinhalte werden über AWS S3/CDN bereitgestellt.
* Externe Systeme wie Zahlungsdienste (Stripe) oder Empfehlungsalgorithmen sind angebunden.

Das Komponentenmodell zeigt die wichtigsten Microservices:

* Kursmanagement-Service: Verwaltung von Kursen, Lektionen, Kategorien.
* Nutzerverwaltung: Registrierung, Login, Rollen & Berechtigungen.
* Zahlungsservice: Abwicklung von Zahlungen und Abonnements.
* Empfehlungssystem: Generierung personalisierter Kursvorschläge.

# Architecture Decisions

## 4.1 Major Decisions, Challenges and Risks

### Technology Decisions

* **Frontend**: Die Wahl fiel auf **React** gegenüber Angular. React wurde aufgrund seiner Flexibilität, hohen Performance und der großen Entwickler-Community bevorzugt. Es ermöglicht eine komponentenbasierte Architektur, die die Wiederverwendbarkeit von Code fördert und eine reaktive Benutzeroberfläche unterstützt. Angular, obwohl gut für große Unternehmensanwendungen geeignet, wurde aufgrund seiner höheren Komplexität und geringeren Flexibilität nicht gewählt. Die Konsequenz dieser Entscheidung ist eine leichtere Wartung und schnellere Entwicklung von neuen Features, allerdings erfordert React eine stärkere Disziplin bei der Zustandsverwaltung.
* **Backend**: Im Backend wurde **Spring Boot** (Java) gegenüber Node.js ausgewählt. Spring Boot bietet eine robuste und sichere Umgebung, die sich besonders gut für Microservices eignet. Es unterstützt die Integration mit relationalen Datenbanken und bietet umfangreiche Sicherheitsfeatures. Node.js, obwohl performant bei I/O-lastigen Anwendungen, wurde aufgrund seiner geringeren Eignung für transaktionssichere Anwendungen nicht gewählt. Die Konsequenz dieser Entscheidung ist eine höhere Stabilität und Sicherheit, allerdings mit einem höheren Ressourcenbedarf im Vergleich zu Node.js.
* **Datenbanken**: Für die Datenhaltung wurde eine Kombination aus **PostgreSQL** (relational) und **MongoDB** (NoSQL) gewählt. PostgreSQL eignet sich ideal für strukturierte Daten wie Nutzerprofile und Zahlungsinformationen, während MongoDB flexibel und skalierbar für unstrukturierte Daten wie Kursinhalte und Kommentare ist. Eine ausschließliche Nutzung von PostgreSQL oder MongoDB wurde verworfen, da dies entweder zu starrer Struktur oder mangelnder Datenintegrität geführt hätte. Die Konsequenz dieser Entscheidung ist eine optimierte Datenverwaltung, allerdings mit erhöhter Komplexität bei der Synchronisation zwischen den Datenbanken.
* **Zahlungsabwicklung**: **PayPal** wurde gegenüber Stripe bevorzugt, da es eine einfachere Integration und bessere Unterstützung für abonnementbasierte Geschäftsmodelle bietet. Beide Anbieter wurden in Betracht gezogen, aber PayPal erwies sich als besser geeignet für die Anforderungen der Plattform. Die Konsequenz dieser Entscheidung ist eine schnellere Implementierung, allerdings mit einer Abhängigkeit von PayPal als externem Dienstleister.
* **Skalierung und Bereitstellung**: **Kubernetes** wurde für die Orchestrierung von Microservices gewählt, da es automatische Skalierung und effizientes Management von Containern ermöglicht. Klassische VM-basierte Deployments wurden aufgrund ihrer geringeren Flexibilität und höheren Wartungskosten verworfen. Die Konsequenz dieser Entscheidung ist eine hohe Skalierbarkeit und Verfügbarkeit, allerdings mit einer steilen Lernkurve für das Entwicklungsteam.

### Major Architecture Decisions

* **Microservices-Architektur**: Die Plattform basiert auf einer **Microservices-Architektur**, die eine hohe Skalierbarkeit, Modularität und Fehlertoleranz bietet. Diese Entscheidung ermöglicht es, einzelne Services unabhängig voneinander zu skalieren und zu warten. Eine monolithische Architektur wurde verworfen, da sie langfristig zu Skalierungsproblemen und höherer Komplexität bei der Wartung geführt hätte.
* **API-Gateway**: Die Plattform nutzt ein **API-Gateway** als zentrale Schnittstelle zwischen Frontend und Backend. Dies vereinfacht die Kommunikation, verbessert die Sicherheit und reduziert die Komplexität für das Frontend. Eine direkte Kommunikation zwischen Frontend und Microservices wurde aufgrund der höheren Komplexität und potenziellen Sicherheitsrisiken verworfen.
* **Containerisierung und Cloud-Hosting**: Die Plattform wird mit **Docker** containerisiert und in einer **Cloud-Umgebung** (AWS oder Azure) gehostet. Dies ermöglicht eine automatisierte Bereitstellung, hohe Verfügbarkeit und bedarfsgerechte Skalierung. On-Premise-Hosting wurde aufgrund der höheren Kosten und geringeren Flexibilität nicht gewählt.

### Risks

* **Skalierbarkeit**: Ein plötzlicher Nutzeranstieg könnte die Skalierbarkeit der Microservices herausfordern. Um dies zu bewältigen, werden Load-Balancing-Techniken und Auto-Scaling-Strategien mit Kubernetes implementiert.
* **Sicherheit**: Die Speicherung und Verarbeitung von Nutzerdaten und Zahlungsinformationen birgt Sicherheitsrisiken. Diese werden durch den Einsatz von OAuth 2.0, JWT-Authentifizierung und TLS-Verschlüsselung minimiert. Zudem werden regelmäßige Sicherheitsaudits und Penetrationstests durchgeführt.
* **Technologische Abhängigkeiten**: Die Abhängigkeit von externen Diensten wie PayPal oder Cloud-Anbietern (AWS, Azure) könnte zu Problemen führen, falls diese Dienste ausfallen oder sich ändern. Es wurden jedoch Verträge und Backup-Strategien implementiert, um diese Risiken zu minimieren.
* **Komplexität der Microservices**: Die Microservices-Architektur erhöht die Komplexität der Plattform, insbesondere bei der Verwaltung und Orchestrierung der Services. Durch den Einsatz von Kubernetes und einem API-Gateway wird diese Komplexität jedoch beherrschbar gehalten.

## 

## 4.2 Security Concept

Die Architektur der Skillshare-Plattform implementiert mehrere Sicherheitsmaßnahmen, um die Software vor Angriffen und Missbrauch zu schützen. Diese Maßnahmen umfassen den Schutz von Kommunikationskanälen, Datenspeichern sowie die Sicherstellung von Authentifizierung, Autorisierung und Protokollierung.

### 4.2.1 Protection against attacks

* **Sichere Authentifizierung**: Die Anmeldung erfolgt über OAuth 2.0 in Kombination mit JSON Web Tokens (JWT), um eine sichere Verwaltung von Nutzeranmeldungen und Tokens zu gewährleisten. Optional wird eine Zwei-Faktor-Authentifizierung (2FA) angeboten, um zusätzliche Sicherheit für Nutzerkonten zu gewährleisten.
* **Datenverschlüsselung**: Die gesamte Datenübertragung zwischen Clients und Servern wird mit TLS 1.3 verschlüsselt, um Man-in-the-Middle-Angriffe zu verhindern. Gespeicherte Daten, insbesondere sensible Informationen wie Nutzerdaten und Zahlungsinformationen, werden mit AES-256 verschlüsselt, um unbefugten Zugriff zu verhindern.
* **Angriffserkennung und -abwehr**: Zur frühzeitigen Erkennung von Bedrohungen werden Intrusion Detection Systeme (IDS) eingesetzt, die verdächtige Aktivitäten überwachen und entsprechende Alarme auslösen. Regelmäßige Sicherheitsaudits und Penetrationstests werden durchgeführt, um Schwachstellen zu identifizieren und zu beheben.
* **Rollenbasierte Zugriffskontrolle (RBAC)**: Zugriffsrechte werden streng geregelt und rollenbasiert vergeben. Nur autorisierte Personen haben Zugriff auf bestimmte Daten und Funktionen. Dies gewährleistet, dass Nutzer nur die Bereiche der Plattform nutzen können, für die sie berechtigt sind.
* **DDoS-Schutz**: Der Einsatz von Web Application Firewalls (WAFs) und Rate-Limiting-Techniken schützt die Plattform vor Distributed Denial-of-Service (DDoS)-Angriffen. Diese Maßnahmen verhindern, dass die Plattform durch übermäßige Anfragen überlastet wird und stellen die Verfügbarkeit der Dienste sicher.
* **Protokollierung und Monitoring**: Alle sicherheitsrelevanten Aktivitäten werden protokolliert, um eine Nachverfolgung von Angriffen oder verdächtigen Aktivitäten zu ermöglichen. Ein zentrales Monitoring-System überwacht die Plattform in Echtzeit und alarmiert das Sicherheitsteam bei verdächtigen Aktivitäten.

### 4.2.2 Data Privacy Protection

Die Plattform wurde entwickelt, um die **Datenschutz-Grundverordnung (DSGVO)** und andere relevante Datenschutzgesetze einzuhalten. Dies umfasst Maßnahmen zum Schutz personenbezogener Daten vor Diebstahl und Missbrauch sowie die Gewährleistung der Rechte der Nutzer.

* **Rechte der Nutzer**: Nutzer können ihre Daten gemäß der DSGVO korrigieren oder löschen lassen. Eine Datenexport-Funktion ermöglicht es Nutzern, ihre gespeicherten Daten herunterzuladen, um Transparenz zu gewährleisten.
* **Datenminimierung und Pseudonymisierung**: Personenbezogene Daten werden, wenn möglich, anonymisiert oder pseudonymisiert gespeichert, um das Risiko von Datenmissbrauch zu minimieren. Die Plattform folgt dem Prinzip der Datenminimierung, d.h., es werden nur die Daten erhoben, die für die Bereitstellung der Dienste erforderlich sind.
* **Backup und Wiederherstellung**: Automatisierte Backups und Notfallwiederherstellungsverfahren schützen vor Datenverlust und gewährleisten die Verfügbarkeit der Plattform im Ernstfall.
* **Sichere Drittanbieter-Integration**: Externe APIs oder Cloud-Dienste werden nur mit DSGVO-konformen Anbietern integriert. Datenverarbeitungsvereinbarungen (DPA) mit externen Dienstleistern stellen die Einhaltung der Datenschutzvorschriften sicher.
* **Sicherheitsupdates**: Regelmäßige Software-Updates und Patch-Management-Prozesse schließen bekannte Sicherheitslücken zeitnah.
* **Zwei-Faktor-Authentifizierung (2FA)**: Nutzer können ihr Konto durch eine optionale 2FA zusätzlich absichern, um den Schutz vor unbefugtem Zugriff zu erhöhen.

# 5 Summary

Die Skillshare-ähnliche Lernplattform wurde als cloudbasierte Lösung entwickelt, um eine skalierbare, sichere und benutzerfreundliche Umgebung für die Erstellung und Nutzung von Online-Kursen zu schaffen. Ziel des Projekts ist es, durch ein hybrides Monetarisierungsmodell langfristige Rentabilität zu erreichen und gleichzeitig eine breite Nutzerbasis anzusprechen. Die Plattform setzt auf einen modernen Technologie-Stack, der Flexibilität und Leistung gewährleistet.

Das Frontend wurde mit React.js entwickelt, um eine responsive Benutzeroberfläche für Web- und Mobile-Anwendungen zu bieten, während das Backend aus Microservices besteht, die mit Spring Boot (Java) implementiert sind. Diese Services werden mithilfe von Docker containerisiert und über Kubernetes orchestriert, was eine effiziente Skalierung und Fehlertoleranz ermöglicht. Für die Datenspeicherung kommen PostgreSQL für strukturierte Daten wie Nutzerprofile und Zahlungen sowie MongoDB für dynamische Kursinhalte zum Einsatz. Redis optimiert die Performance durch Caching, während AWS S3 oder Azure Blob Storage die zuverlässige Bereitstellung von Videoinhalten sicherstellen. Externe Integrationen umfassen Zahlungsdienstleister wie PayPal, OAuth 2.0/JWT für sichere Authentifizierung sowie KI-basierte Empfehlungssysteme.

Die Plattform generiert Einnahmen durch ein diversifiziertes Modell. Kernstück ist das Freemium-Subscription-Modell, bei dem Nutzer kostenlos auf Basisfunktionen zugreifen können, während Premium-Abonnements exklusive Kurse, Offline-Zugriff und Zertifikate bieten. Zusätzlich werden Transaktionsgebühren (15–20 % Provision) auf Verkäufe von Creatorn erhoben, etwa für Masterclasses oder Live-Workshops. Weitere Einnahmequellen umfassen B2B-Partnerschaften, bei denen Unternehmen Lizenzgebühren für interne Schulungen zahlen, sowie Zusatzdienste wie gebührenpflichtige Zertifizierungen, Premium-Support und ein Marktplatz für physische Produkte. Diese Kombination stabilisiert den Cashflow und reduziert die Abhängigkeit von einzelnen Erlösströmen.

Die Microservices-Architektur gewährleistet Modularität und Skalierbarkeit. Ein API-Gateway fungiert als zentrale Schnittstelle, vereinfacht die Kommunikation zwischen Frontend und Backend und übernimmt Sicherheitsaufgaben wie Authentifizierung und DDoS-Schutz. Die Cloud-basierte Infrastruktur (AWS/Azure) ermöglicht automatisches Skalieren und hohe Verfügbarkeit. Sicherheit steht im Fokus: Datenübertragungen werden mit TLS 1.3 verschlüsselt, gespeicherte Daten mit AES-256. Die Plattform erfüllt DSGVO-Anforderungen durch Pseudonymisierung, Nutzerdatenlöschung und transparente Datenschutzerklärungen. Regelmäßige Penetrationstests und eine Web Application Firewall (WAF) schützen vor Angriffen.

Durch Kubernetes und Cloud Auto-Scaling passt sich die Infrastruktur dynamisch an Nutzerspitzen an. Kosteneffizienz wird durch Open-Source-Tools und optimierte Ressourcennutzung erreicht. Netzwerkeffekte – etwa virales Wachstum über Creator-Netzwerke und lokalisierte Inhalte – stärken die Marktposition und minimieren Wettbewerbsrisiken. Insgesamt bietet die Plattform eine solide Grundlage für den langfristigen Erfolg, indem sie Skalierbarkeit, Sicherheit und Benutzerfreundlichkeit in Einklang bringt.