-.ChipsyTok.-

Inhalte

[**1. Ausgangslage** 1](#_Toc193113189)

[**2. Zielsetzung und Anforderungen** 2](#_Toc193113190)

[2.1 Funktionale Anforderungen 2](#_Toc193113191)

[2.2 Nichtfunktionale Anforderungen 2](#_Toc193113192)

[2.3 User-Stories und Detailabläufe 3](#_Toc193113193)

[**ChipsyPost erstellen & teilen** 3](#_Toc193113194)

[**Chat / Nachrichten** 3](#_Toc193113195)

[**3. Aufgabenverteilung** 4](#_Toc193113196)

[3.1 Rollen und Zuständigkeiten 4](#_Toc193113197)

[3.2 Arbeitspakete 4](#_Toc193113198)

[**4. Zeitplan 5**](#_Toc193113199)

[**5. Implementierung 6**](#_Toc193113200)

[5.1 Technologie-Stack 6](#_Toc193113201)

[5.2 Struktur und Vorgehen 7](#_Toc193113202)

[**6. Offene Punkte / Risiken 7**](#_Toc193113203)

[**7. Fazit / Ausblick 8**](#_Toc193113204)

# 

# **1. Ausgangslage**

Die mit den Modulen 106, 294, 322 und 165 verknüpften Kompetenzen sollen in Form eines **projektorientierten Praxisteils** demonstriert werden. Das geplante Projekt heisst **ChipsyTok** und ermöglicht das Teilen von unterhaltsamen Bild-Posts (ChipsyPosts) mit kurzen, mehrsprachigen Texten.

* Die Anwendung soll die **Mehrsprachigkeit** als zentrales Feature aufweisen.
* Die Deadline für die fertige Lösung ist **Mitte Juni**.
* In der vorangegangenen Lektion wurden **erste Ideen** gesammelt, der grobe Funktionsumfang festgelegt und erste UI-Skizzen erstellt.

# **2. Zielsetzung und Anforderungen**

**Ziel:**  
Mit „ChipsyTok“ wollen wir eine unterhaltsame Plattform realisieren, die eine Mischung aus Social-Media-Interaktion, Mehrsprachigkeit und leichten Chat-Funktionen bietet.

## 2.1 Funktionale Anforderungen

1. **Registrierung und Login**
   * Nutzerinnen und Nutzer müssen sich registrieren können (mind. E-Mail & Passwort).
   * Sichere Aufbewahrung der Passwörter (Hashing).
2. **Mehrsprachigkeit**
   * Umschalten der Inhalte (z. B. Deutsch & Englisch) über einen Sprachschalter.
   * Dynamisches Laden der Texte abhängig von der gewählten Sprache (z. B. i18n-Library).
3. **ChipsyPosts (Bilder)**
   * Anzeige eines Feeds mit vorhandenen Bildern.
   * Like-Funktion (inkl. Anzeige der Like-Anzahl).
   * Eigene Bilder hochladen, kurze Bildbeschreibung in mehreren Sprachen eingeben.
4. **Sharing / Weiterleiten**
   * Möglichkeit, einen ChipsyPost an andere Nutzerinnen und Nutzer direkt zu senden oder in deren Feed zu teilen.
5. **Chat / Nachrichten**
   * Grundlegender Chat zwischen befreundeten Usern.
   * Freundschaftsanfragen: Chat nur bei gegenseitiger Annahme.
6. **Profilverwaltung**
   * Benutzerprofile mit Profilbild und Kurzinfo.
   * Anpassung der Sprache, ggf. Einstellung weiterer Optionen.

*(Diese Anforderungen können je nach Projektfortschritt oder Feedback erweitert werden.)*

## 2.2 Nichtfunktionale Anforderungen

1. **Performance**
   * Die Anwendung soll bei ~20 gleichzeitigen Nutzenden stabil und flüssig sein.
2. **Usability**
   * **Responsives** Web-Design: Nutzbar auf Desktop, Tablets und Smartphones.
   * Intuitive Navigationsstruktur (leichte Einarbeitung).
3. **Sicherheit**
   * Passwörter werden verschlüsselt (Hash + Salt).
   * Datenbankzugriffe geschützt (kein SQL-Injection etc.).
4. **Skalierbarkeit**
   * Einfache Einbindung weiterer Sprachen (z. B. Französisch, Spanisch).
   * Möglichkeit, das Datenbankmodell für grosse Datenmengen zu erweitern.
5. **Wartbarkeit**
   * Klare **Trennung** von Frontend und Backend.
   * Dokumentation des Codes, sinnvolle Commit-Messages, Code-Reviews.
6. **Erweiterbarkeit** (optional)
   * Nutzung einer Cloud zum Speichern von Bildern (z. B. AWS S3 oder ähnliches).
   * Spätere Einführung von Videofunktionen (derzeit „Won’t have“, könnte aber in Version 2.0 relevant werden).

## 2.3 User-Stories und Detailabläufe

1. **Registrierung & Login**

**User-Story:** „Als neuer Nutzer möchte ich mich registrieren können, damit ich ChipsyTok vollumfänglich verwenden kann.“

* + **Akzeptanzkriterien:**
    - Formular mit E-Mail, Passwort (min. 8 Zeichen).
    - Anzeige von Fehlermeldungen (z. B. bei bereits existierender E-Mail).
    - Gesicherte Sessions nach erfolgreichem Login.

1. **Mehrsprachige Ansicht**

**User-Story:** „Als Nutzer möchte ich schnell die Sprache umstellen können, um die Inhalte in meiner bevorzugten Sprache zu sehen.“

* + **Akzeptanzkriterien:**
    - Ein Klick auf einen Sprachschalter oder ein Dropdown ändert sofort alle sichtbaren Texte.
    - Neue Posts enthalten Übersetzungen in allen hinterlegten Sprachen.

# **ChipsyPost erstellen & teilen**

**User-Story:** „Als eingeloggter Nutzer kann ich eigene Bilder hochladen und kurze Beschreibungen in verschiedenen Sprachen verfassen.“

* + **Akzeptanzkriterien:**
    - Upload per Dateiauswahl (Bildformate: JPG, PNG, …).
    - Eingabefelder für die Bildbeschreibung pro Sprache.
    - Sichtbarkeit der hochgeladenen Posts in meinem Feed und beim Teilen in den Feeds anderer.

# **Chat / Nachrichten**

**User-Story:** „Als Nutzer möchte ich meine Freunde direkt anschreiben können, um mich mit ihnen über die Inhalte auszutauschen.“

* + **Akzeptanzkriterien:**
    - Freundschaftsverwaltung (Anfrage versenden, Bestätigung).
    - Message-Fenster oder Chat-Bereich.
    - Anzeige neuer Nachrichten (ggf. Benachrichtigung).

(Weitere Stories können hier ergänzt werden, z. B. Profilbearbeitung, Benachrichtigungen, etc.)

# **3. Aufgabenverteilung**

In diesem Abschnitt wird dargestellt, wer welche Teilaufgaben übernimmt und bis wann sie erledigt sein sollen.

## 3.1 Rollen und Zuständigkeiten

* **Alessio Huber (UI-Verantwortlicher)**
  + Entwurf und Umsetzung des UI/UX-Designs.
  + Erstellung responsiver Layouts.
  + Zusammenarbeit mit Frontend (Kush) hinsichtlich Styling und Komponentenanordnung.
* **Jamie Poeffel (Backend-Verantwortlicher)**
  + Datenbankanbindung (SQL oder NoSQL).
  + Erstellung und Absicherung der API-Routen (z. B. für Bilder, Chat, Registrierung).
  + Implementierung von Authentifizierung und Session-Management.
* **Kush Srigiri (Frontend-Verantwortlicher)**
  + Nuxt.js-Konfiguration und Komponentenstruktur.
  + Integration der i18n-Library für Mehrsprachigkeit.
  + Anbindung an das Backend (API-Calls), Validierung der Benutzereingaben.

*(Alle helfen sich gegenseitig, falls es zeitliche Engpässe gibt. Regelmässige Updates im Team.)*

## 3.2 Arbeitspakete

Eine mögliche Aufteilung der Aufgaben:

| **Arbeitspaket** | **Beschreibung** | **Verantwortlich** | **Deadline** |
| --- | --- | --- | --- |
| **Datenbankstruktur** | Tabellen/Collections für User, Posts, Chat. Beziehungen festlegen (z. B. User-Post). | Jamie | 05.05. (Beispiel) |
| **UI-Design (Mockups)** | Erste Layout-Entwürfe (Startseite, Profil, Feed). | Alessio | 10.05. |
| **Nuxt.js Grundgerüst** | Projekt einrichten, Routing, Struktur, i18n-Plugin einbinden. | Kush | 12.05. |
| **Auth & Registrierung** | Backend-Routen zum Registrieren/Einloggen, Passwort-Hashing, Session mgmt. | Jamie | 15.05. |
| **Upload-Funktion & Feed** | Frontend: Bild-Upload, Feed-Ansicht. Backend: Route für Upload/Download, DB-Speicherung. | Kush / Jamie | 20.05. |
| **UI-Feinschliff** | Feinheiten im Layout, responsives Design, evtl. Animations. | Alessio | 25.05. |
| **Mehrsprachige Post-Beschreibung** | Eingabe mehrsprachiger Texte, dynamisches Umschalten im Frontend, DB-Anpassungen. | Kush (Front.), Jamie (DB) | 30.05. |
| **Chat-Funktion** | Nachrichtenmodell, API-Route, evtl. WebSocket/Realtime oder Polling. | Jamie + (Front: Kush) | 05.06. |
| **Likes / Teilen** | Button „Like“ & „Teilen“ (Frontend+Backend), Zähler anzeigen, Weiterleitungslogik. | Alessio / Kush | 08.06. |
| **Finales Testing & Bugfixing** | Umfangreiche Tests, Fehlerbehebung, ggf. Usability-Verbesserungen. | Alle | 14.06. |
| **Abgabe & Präsentation** | Dokumentation finalisieren, Demo vorbereiten. | Alle | **16.06.** |

*(Die Daten sind Beispiele und können an euren realen Ablauf angepasst werden, solange das Gesamtziel 16. Juni im Blick bleibt.)*

# **4. Zeitplan**

Ein **möglichst detaillierter Zeitplan** hilft, bis zum 16. Juni alle relevanten Funktionen zu implementieren und ausreichend zu testen.

1. **KW 1 – KW 2 (Anfang bis Mitte Mai)**
   * Datenmodell definieren, Grundlegendes UI-Design, Projekt in Nuxt.js aufsetzen.
   * Registrierung, Login, erste Datenbankanbindung.
2. **KW 3 – KW 4 (Mitte bis Ende Mai)**
   * Bild-Upload & Feed-Funktion.
   * Mehrsprachigkeits-Funktion integrieren (i18n).
   * UI-Finishing (Responsivität, Grundfarben, Schriftarten).
3. **KW 5 – KW 6 (Anfang bis Mitte Juni)**
   * Chat-Funktion (Freundschaftsanfragen, Nachrichten).
   * Like-/Teilen-Funktion, Optimierung der DB-Abfragen.
   * Intensives Testing (manuelle Tests, ggf. einfache Unit-Tests).
4. **Letzte Woche vor 16.06.**
   * **Finale Fehlerbehebung**, Doku-Abschluss, ggf. kleine Features abrunden.
   * Präsentation vorbereiten: Live-Demo aller Kernfunktionen.

# **5. Implementierung**

## 5.1 Technologie-Stack

* **Frontend:**
  + Nuxt.js (Vue.js-basiertes Framework)
  + HTML5, CSS3 (ggf. Tailwind, Bootstrap oder Vuetify)
  + i18n-Library (z. B. vue-i18n oder Nuxt i18n-Plugin)
* **Backend:**
  + Node.js / Express (oder ein ähnliches Framework)
  + REST-API-Schnittstellen für User-Management, Posts, Chat
  + Authentifizierung (z. B. JWT oder Session-Cookies)
* **Datenbank:**
  + Wahlweise **SQL** (z. B. MySQL, PostgreSQL) oder **NoSQL** (z. B. MongoDB)
  + Sammlung für User-Daten, Posts, Chat-Nachrichten
* **Sonstige Werkzeuge:**
  + Git (Versionierung), GitHub/GitLab (Code Hosting, Issues)
  + Möglicherweise Docker (Containerisierung von Backend + DB)
  + npm oder yarn für Paketverwaltung

## 5.2 Struktur und Vorgehen

* **Repository-Aufteilung:**
  + Separate Ordner/Repos für Frontend (Nuxt) und Backend (Node/Express).
* **Code-Reviews & Stand-Ups:**
  + Wöchentliche Kurzmeetings (10–15 Min), um den Status abzugleichen.
  + Code-Review durch mindestens eine andere Person, um Fehler früh zu erkennen.
* **Testing & Qualitätssicherung:**
  + Manuelle Klicktests (z. B. Registrierung, Bild-Upload, Chat).
  + Evtl. Unit-Tests (Jest) oder Integrationstests bei ausreichend Zeit.
* **Deployment (optional):**
  + Lokales Deployment zum Test.
  + Optionale Cloud-Bereitstellung (z. B. Heroku, Vercel), wenn Zeit und Ressourcen es erlauben.

# **6. Offene Punkte / Risiken**

1. **Bildspeicherung:**
   * Lokal im Projekt oder externes Cloud-Storage? Datenmengen könnten gross werden.
2. **Chat in Echtzeit?**
   * Entscheidung zwischen WebSockets (komplexer, aber „echter“ Echtzeit-Chat) und Polling (einfacher, aber weniger elegant).
3. **Mehrsprachige Datensätze:**
   * Aufwand beim Erstellen neuer ChipsyPosts steigt (mehrere Texte einpflegen).
   * Gut strukturierte DB nötig (pro Post eine Tabelle/Collection für Übersetzungen?).
4. **Zeitmanagement:**
   * Mögliche Verzögerungen durch andere schulische Projekte oder Feiertage.
   * Regelmässige Kontrolle der Deadlines, Puffer einplanen.

# **7. Fazit / Ausblick**

Mit diesem Dokument ist der **grundlegende Projektplan** für ChipsyTok definiert. Dank der **Aufgabenverteilung** weiss jeder im Team, wofür er verantwortlich ist. Der **Zeitplan** bis zum 16. Juni ist realistisch, erfordert aber konsequentes Arbeiten und regelmässige Abstimmung.

**Nächste Schritte**

* **Erstellung der Datenbankstruktur** (Jamie) & Bestätigung durch das Team.
* **Grundlegende UI-Skizzen** verfeinern (Alessio).
* **Nuxt.js-Projekt initialisieren** (Kush), erste i18n-Integration.

Wenn diese Meilensteine erreicht sind, kann mit den Kernfunktionen (Bilder-Upload, mehrsprachige Post-Beschreibung, Chat) begonnen werden. Eventuelle Extrafunktionen (Analytics, Profilgestaltung, Hashtags, etc.) können bei ausreichend Zeit ebenfalls integriert werden.

**Ziel:**  
Ein stabiler **MVP** (Minimum Viable Product) mit allen Must-have-Funktionen sollte spätestens **eine Woche vor die Sommerferien** stehen, damit genug Zeit für abschliessende Tests und Anpassungen bleibt. So kann am Stichtag eine funktionsfähige, ansprechende und mehrsprachige Web-App präsentiert werden.