Prototyping Projektdokumentation

Name: Leonardo Gaglio

E-Mail: gaglileo@students.zhaw.ch

URL der deployten Anwendung: <https://nexanote.netlify.app/>

# Einleitung

NexaNote ist eine moderne Web-Anwendung zur Organisation von Aufgaben, Notizen und Dateien an einem zentralen Ort. Die Grundidee besteht darin, Themen anzulegen, denen beliebig viele Aufgaben, Notizen oder Dateien zugeordnet werden können. Alternativ können Einträge auch ohne direkte Zuordnung zu einem Thema erstellt werden, um maximale Flexibilität zu gewährleisten. Die Anwendung bietet eine übersichtliche und benutzerfreundliche Oberfläche und eignet sich besonders für Studium, Projektarbeit oder private Organisation. Zusätzlich wurde eine Chat-Funktion integriert, die als Erweiterung dient und den Nutzern ermöglicht, Fragen zu stellen und Unterstützung innerhalb der Anwendung zu erhalten.

# Datenmodell

*Ein Bild, das Entwurf, Zeichnung, Diagramm, Lineart enthält.

KI-generierte Inhalte können fehlerhaft sein.*

Ein Thema kann beliebig viele Aufgaben, Notizen und Dateien enthalten, muss aber keine enthalten. Umgekehrt können Aufgaben, Notizen und Dateien entweder einem Thema zugeordnet sein oder auch ohne Thema existieren. Dadurch bleibt die Zuordnung optional und es können sowohl themenbezogene als auch „lose“ Elemente gespeichert werden.

# Beschreibung der Anwendung

## Startseite

Route: /

Beim Aufrufen der Webanwendung landet man zunächst auf die Hauptseite. Auf dieser Seite werden die wichtigsten Funktionen von NexaNote in Form von Kacheln präsentiert.

Ein Bild, das Text, Screenshot, Software, Computersymbol enthält.

KI-generierte Inhalte können fehlerhaft sein.

Für die Darstellung der einzelnen Features wird die Komponente FeatureTitle.svelte verwendet. Über die Kacheln gelangt man direkt zu den jeweiligen Bereichen der Plattform. Zusätzlich hebt eine eigene Kachel den Zugang zum AI Chat hervor. Die Sidebar ermöglicht die zentrale Navigation zwischen allen Hauptbereichen und ist dauerhaft über die gesamte App hinweg sichtbar. Sie ist als OffcanvasSidebar in +layout.svelte eingebunden.

Dateien:

* lib/components/FeatureTitle.svelte
* lib/components/OffcanvasSidebar.svelte
* routes/+page.svelte
* routes/+layout.svelte

## Topic-Übersicht

Route: /topics

Auf dieser Seite werden alle Themen (Topics) aufgelistet.

Ein Bild, das Text, Screenshot, Software, Computer enthält.

KI-generierte Inhalte können fehlerhaft sein.

Für die Darstellung der einzelnen Themen wird die Komponente TopicCard.svelte verwendet. Über das Dropdown-Menü «Kategorie» kann die Liste nach den Kategorien Modul, Projekt oder Übung gefiltert werden. Mit dem Button «Filter zurücksetzen» wird die Filterung aufgehoben und alle Themen angezeigt. Über den Button «Neues Thema» gelangt man zur Seite /topics/create zur Erstellung eines neuen Themas.

Mit dem + Button in jeder Themen-Karte gelangt man zur Seite /topics/create?parentId=ID um ein Unterthema zum bestehenden Topic zu erstellen. Über den Button «bearbeiten» in der Themen-Karte gelangt man direkt zur Detailseite des jeweiligen Themas /topics/[id].

Dateien:

* lib/components/TopicCard.svelte
* routes/topics/+page.svelte
* routes/topics/+page.server.js

## Topic Erstellen

Route: /topics/create

Auf dieser Seite wird ein Formular bereitgestellt, um ein neues Thema (Topic) zu erstellen.

Ein Bild, das Text, Screenshot, Software, Computersymbol enthält.

KI-generierte Inhalte können fehlerhaft sein.

Im Formular können Titel, Beschreibung, Kategorie (Modul, Projekt, Übung) sowie eine individuelle Farbe für das Thema gewählt werden.

Dateien:

* routes/topics/create/+page.svelte
* routes/topics/create/+page.server.js
* lib/db/topics.js

## Topic-Detailseite

Route: /topics/[id]

Hier werden alle Informationen zum gewählten Thema (Topic) angezeigt.

Ein Bild, das Text, Screenshot, Software, Computersymbol enthält.

KI-generierte Inhalte können fehlerhaft sein.

Die Seite ist in verschiedene Bootstrap-Karten unterteilt und verwendet ein Grid-Layout für eine übersichtliche Darstellung. Es gibt folgende Funktionen:

* Bearbeiten und Löschen: Das aktuelle Topic kann direkt bearbeitet oder gelöscht werden.
* Aufgabenliste: Aufgaben können erstellt, aufgelistet und gelöscht werden.
* Notizen: Notizen mit Titel und Inhalt können hinzugefügt und entfernt werden.
* Dateiupload: Dateien lassen sich hochladen und sind mit dem Topic verknüpft sichtbar.
* Kommentare: Kommentare und Feedback können eingetragen und angezeigt werden.

Dateien:

* routes/topics/[id]/+page.svelte
* routes/topics/[id]/+page.server.js
* lib/db/objects.js

## Aufgaben-Übersicht

Route: /tasks

Auf dieser Seite werden alle Aufgaben in einer übersichtlichen Kartenansicht dargestellt.

Ein Bild, das Text, Screenshot, Software, Computersymbol enthält.

KI-generierte Inhalte können fehlerhaft sein.

Neue Aufgaben können direkt über das Formular oben erstellt werden. Für die Darstellung einzelner Aufgaben wird die Komponente TaskCard.svelte verwendet. Beim Erstellen einer Aufgabe sind Titel und Fälligkeitsdatum Pflichtfelder, das Anlegen erfolgt direkt auf dieser Seite.

Über den Button «Details» gelangt man zur Detailansicht einer Aufgabe unter /tasks/[id]. Mithilfe einer Checkbox kann die Liste nach offenen Aufgaben gefiltert werden.

Dateien:

* routes/tasks/+page.svelte
* routes/tasks/+page.server.js
* lib/components/TaskCard.svelte
* lib/db/tasks.js

## Aufgaben-Detailseite

Route: /tasks/[id]

Auf dieser Seite können die Details einer Aufgabe angezeigt, bearbeitet oder gelöscht werden.

Ein Bild, das Text, Screenshot, Software, Computersymbol enthält.

KI-generierte Inhalte können fehlerhaft sein.

Titel und Fälligkeitsdatum bleiben Pflichtfelder. Der Status kann per Checkbox auf „Erledigt“ gesetzt werden. Zusätzlich können Kommentare zur Aufgabe hinzugefügt und angezeigt werden. Über den Button «Zurück zur Übersicht» gelangt man zur Aufgaben-Übersicht /tasks.

Dateien:

* routes/tasks/[id]/+page.svelte
* routes/tasks/[id]/+page.server.js
* lib/db/tasks.js

## Notizen-Übersicht

Route: /notes

Auf dieser Seite werden alle Notizen in einer Kartenansicht dargestellt.

Ein Bild, das Text, Screenshot, Software, Display enthält.

KI-generierte Inhalte können fehlerhaft sein.

Für die Anzeige der einzelnen Notizen wird die Komponente NoteCard.svelte verwendet. Neue Notizen können über den Button «Neue Note» erstellt werden. Bestehende Notizen können über «Anzeigen» geladen werden, um Details zu bearbeiten oder sie zu löschen.

Dateien:

* routes/notes/+page.svelte
* routes/notes/+page.server.js
* lib/components/NoteCard.svelte
* lib/db/notes.js

## Notiz erstellen

Route: /notes/create

Ein Bild, das Text, Screenshot, Software, Computersymbol enthält.

KI-generierte Inhalte können fehlerhaft sein.

Auf dieser Seite kann eine neue Notiz über ein Formular erstellt werden. Der Titel ist ein Pflichtfeld. Nach dem Speichern wird zur Übersicht weitergeleitet.

Dateien:

* routes/notes/create/+page.svelte
* routes/notes/create/+page.server.js
* lib/db/notes.js

## Notizen-Detailseite

Route: /notes/[id]

Auf dieser Seite werden die Details einer einzelnen Notiz angezeigt.

Ein Bild, das Text, Screenshot, Software, Computersymbol enthält.

KI-generierte Inhalte können fehlerhaft sein.

Titel und Inhalt können bearbeitet und gespeichert werden. Die Notiz kann ausserdem gelöscht werden, danach erfolgt eine Weiterleitung zur Übersicht.

Dateien:

* routes/notes/[id]/+page.svelte
* routes/notes/[id]/+page.server.js
* lib/db/notes.js

# Erweiterungen

## Aufgaben-Filter

Auf der Seite /tasks können über eine Checkbox erledigte Aufgaben ausgeblendet werden, sodass nur noch offene Aufgaben angezeigt werden.

Ein Bild, das Text, Screenshot, Software, Computersymbol enthält.

KI-generierte Inhalte können fehlerhaft sein.

Dies wird über eine Checkbox im UI und eine clientseitige Filterung des Aufgaben-Arrays in der Svelte-Komponente realisiert. Die Filterfunktion prüft das Status-Feld completed jeder Aufgabe.

Dateien :

* lib/db/tasks.js
* routes/tasks/+page.svelte
* routes/tasks/+page.server.js

## Unterthemen (Hierarchie-Unterstützung)

Das Anlegen eines Unterthemas (Child-Topic) erfolgt direkt über den + Button in der Themenübersicht. Beim Klick auf diesen Button in einer TopicCard wird die Seite /topics/create mit dem entsprechenden parentId-Parameter aufgerufen (z. B. /topics/create?parentId=...).

Ein Bild, das Text, Screenshot, Software, Webseite enthält.

KI-generierte Inhalte können fehlerhaft sein.

Dadurch wird das neue Thema beim Speichern automatisch als Unterthema dem gewählten Parent-Topic zugeordnet und später als Child-Topic in der Themenübersicht angezeigt.

Ein Bild, das Text, Screenshot, Software, Computersymbol enthält.

KI-generierte Inhalte können fehlerhaft sein.

Dateien:

* lib/db/topics.js
* routes/topics/create/+page.svelte
* routes/topics/create/+page.server.js

## Themen-Filter

Auf der Seite /topics können die angezeigten Themen über ein Dropdown-Menü nach Kategorie (z. B. Modul, Projekt oder Übung) gefiltert werden.

Ein Bild, das Text, Screenshot, Software, Computer enthält.

KI-generierte Inhalte können fehlerhaft sein.

Die Filterung erfolgt clientseitig in der Svelte-Komponente, indem das geladene Topics-Array anhand des Feldes type gefiltert wird. Die Datenbank liefert stets alle Topics zurück; das Filtern übernimmt das Frontend nach Auswahl im Dropdown

Dateien:

* lib/db/topics.js
* routes/topics/+page.svelte
* routes/topics/+page.server.js

## Dateien upload

Route: /files

Auf dieser Seite können Dateien hochgeladen, angezeigt und heruntergeladen werden.

Ein Bild, das Text, Screenshot, Software, Webseite enthält.

KI-generierte Inhalte können fehlerhaft sein.

Für jede Datei werden Titel, File-Name, Typ und Upload-Datum angezeigt. Der Upload erfolgt direkt über ein Formular. Nach dem Absenden wird die Datei im Verzeichnis /static/uploads auf dem Server gespeichert und die zugehörigen Metadaten (wie Name, Typ, Grösse) werden in der Datenbank hinterlegt.

Für die Anzeige und Verwaltung der Dateien wird die Komponente FileCard.svelte verwendet. Jede Datei kann über einen Button heruntergeladen oder direkt angesehen werden (je nach Dateityp und Browserunterstützung).

Die Umsetzung erfordert neben der Interaktion mit der Datenbank auch den direkten Umgang mit dem Dateisystem (fs-Modul von Node.js) zum Speichern und Bereitstellen der Dateien auf dem Server.

Dateien

* routes/files/+page.svelte
* routes/files/+page.server.js
* lib/components/FileCard.svelte
* lib/db/files.js

## Gemini Chat

Route: /chat

Auf dieser Seite kann direkt mit dem Gemini KI-Modell gechattet werden.

Ein Bild, das Text, Screenshot, Display, Software enthält.

KI-generierte Inhalte können fehlerhaft sein.

Die Anbindung erfolgt serverseitig: Die Chat-Komponente sendet Nutzereingaben an den Server-Endpunkt /chat/api, der als Proxy die Anfrage mit dem API-Key an die Google Gemini API weiterleitet. Der Nachrichtenverlauf wird nur im Browser angezeigt und nicht dauerhaft gespeichert. Die gesamte Kommunikation mit der Google API erfolgt über den-Endpoint /chat/api. Das Modell sowie alle Parameter werden im Servercode festgelegt. Die sichere Speicherung des API-Keys erfolgt auf Netlify als Environment Variable.

Dateien:

* routes/chat/+page.svelte
* routes/chat/api/+server.js
* lib/components/Chat.svelte