LazyNotes

Dokumentation

vorgelegt am 9. Februar 2024

Fakultät Wirtschaft

Studiengang Wirtschaftsinformatik

Kurs WWI2021F

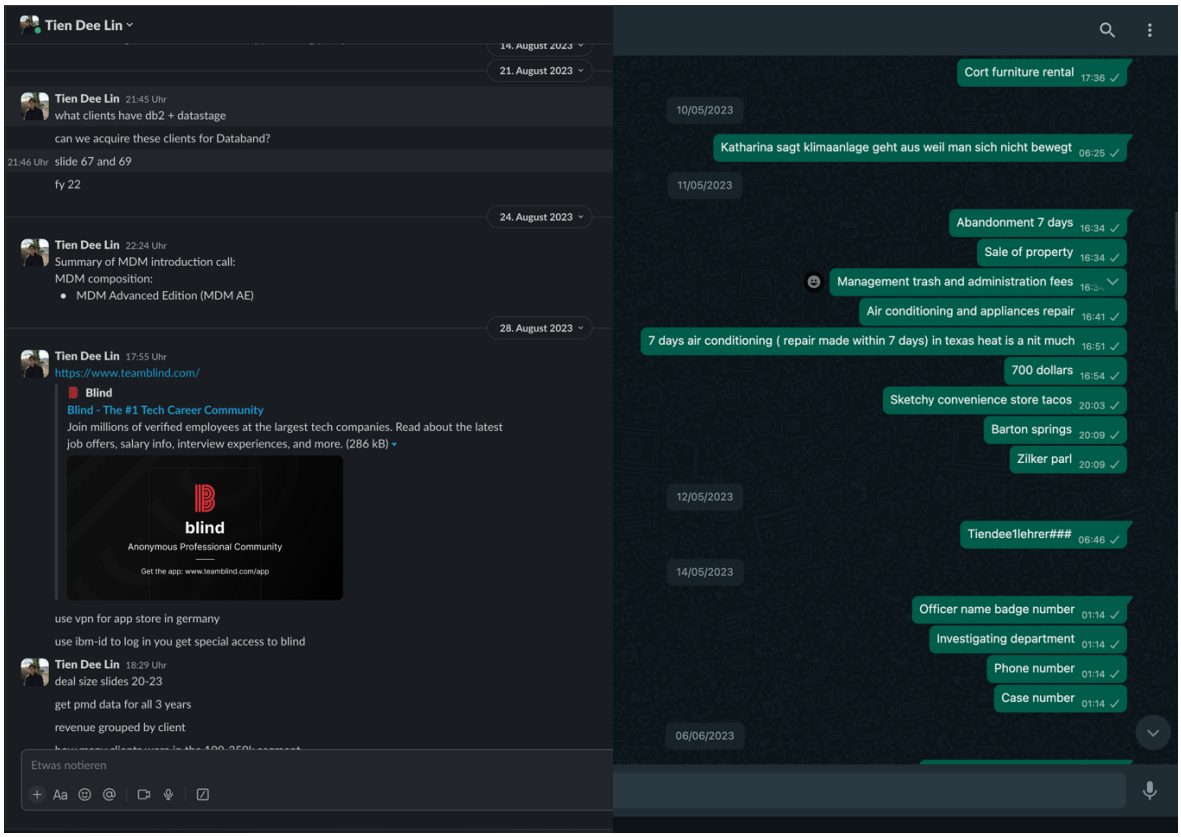
von

Tien Dee Lin

Inhaltsverzeichnis

1 Ziel und Funktionalität der Software

Notizen-Apps haben sich als unentbehrliche Werkzeuge im Alltag etabliert. Sie sind auf zahlreichen Geräten vorinstalliert und werden vielfältig genutzt. Trotzdem gibt es dabei eine Herausforderung: Organisation. Die Fähigkeit organisiert zu sein, variiert von Person zu Person erheblich. Viele machen sich regelmäßig Notizen – manche erfolgreich, andere weniger. Gerade in hektischen Phasen oder bei natürlicherweise weniger strukturierten Menschen scheint der Aufwand, Notizen sorgfältig und systematisch zu pflegen, oft zu groß. Ein Beispiel für dieses organisatorische Dilemma ist das Versenden von Nachrichten an sich selbst – sei es über WhatsApp, Slack oder andere häufig genutzte Messenger – in der Hoffnung, zum entscheidenden Moment die zündende Idee, die wichtige Information oder die notwendige Erinnerung wiederzufinden. Dies führt jedoch oft zu chaotischen und ungeordneten Notizsammlungen, die letztlich ihrem Zweck nicht dienen.

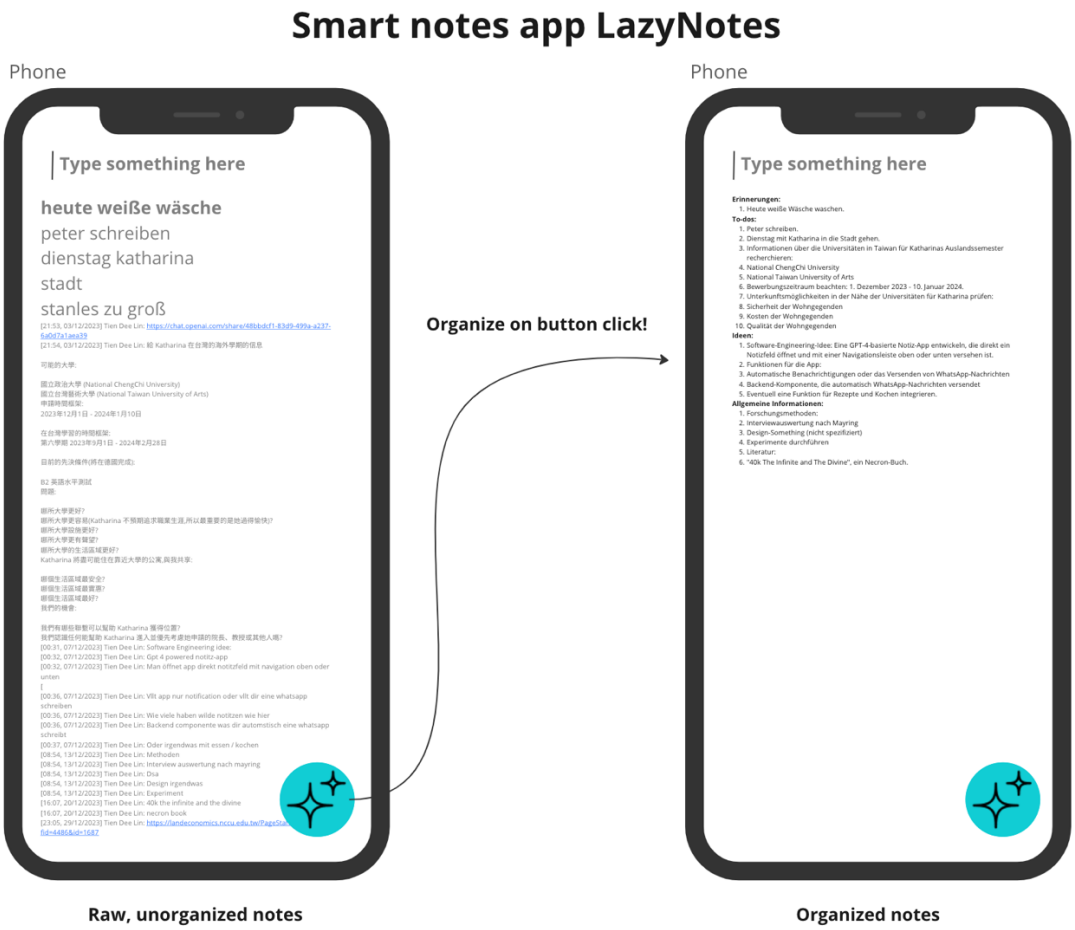


Wie im Beispiel aufgezeigt, sind die Notizen kaum wiederzuerkennen, und Informationen sind oft auch mit großem Aufwand nicht mehr nutzbar. Einziger Vorteil dieser Art der Notizführung ist die einfache und intuitive Benutzeroberfläche der Apps, die durch regelmäßige Verwendung vertraut sind. Doch das daraus resultierende Chaos kann unterschiedliche Konsequenzen nach sich ziehen, wie das Übersehen einer innovativen Idee, das Versäumen von Terminen oder gravierendere Auswirkungen.

LazyNotes zielt darauf ab, eine Notiz-App zu präsentieren, die Nutzern die vertraute Einfachheit und intuitive Bedienung bietet, jedoch mit einem klaren Vorteil: Sie organisiert die Notizen automatisch. So wird es Nutzern ermöglicht, ihre natürliche Neigung zur Aufwandsoptimierung in der Notizführung zu bewahren, ohne die organisatorische Leistungsfähigkeit zu beeinträchtigen. Die Effizienz in der täglichen Organisation wird mühelos gesteigert, ohne dass jemals wieder ein wertvoller Gedanke oder bedeutende Erinnerung verloren geht.

LazyNotes erreicht dies, indem es auf Knopfdruck die ungefilterten Notizen der Nutzer mit einem speziell angepassten Prompt über die OpenAI API an das GPT-4 Modell zur Verarbeitung übergibt. Die in die Zielsprache übersetzten, sprachlich optimierten und in Kategorien eingeteilte Notizen ersetzten im Notizfeld die rohen Notizen und der Nutzer kann diese bei Bedarf bearbeiten oder weitere rohe Notizen an beliebiger Stelle einfügen.

2 UI-Konzept



Das zentrale Prinzip des UI-Konzepts ist die minimalistische und einfache Bedienung. Die App besteht aus einer einzigen Seite, die beim Öffnen direkt sichtbar ist. Die Hauptseite besteht aus einem Textfeld, welches die gesamte Seite ausfüllt und dem Nutzer ermöglicht, alle Notizen einzutragen. In der unteren rechten Ecke befindet sich ein Button, der beim Anklicken die Notizen aus dem Textfeld nimmt und sie mithilfe der OpenAI API (Gpt-4) formatiert und ordnet. Die organisierten Notizen werden, sobald sie von der OpenAI API zurückgesendet werden, direkt im Notizfeld angezeigt. Der Nutzer kann die strukturierten Notizen bearbeiten oder unstrukturierte Notizen an jeder Stelle hinzufügen. Die App bietet keine Funktionen zur manuellen Gestaltung der Notizen, da die Zielgruppe Nutzer sind, die weder Interesse noch Zeit haben, ihre Notizen selbst zu organisieren. Der Organisieren-Button bietet jedoch eine UI-Schnittstelle, um zusätzliche Funktionen zu implementieren. Zum Beispiel könnte durch längeres Drücken des Buttons eine Auswahl von Buttons erscheinen, die es ermöglichen, verschiedene Einstellungen wie die Sprachauswahl oder bevorzugte Notizkategorien anzupassen. Das Notizfeld als Zentrales Element findet man auch in etablierten Notiz-Apps (z.B. IOS „Notes“) wieder. Die Verwendung von Notizkarten oder anderen strukturierten UI-Elemente sind nicht vorgesehen, da sie die Komplexität und somit auch den Aufwand der Nutzung erhöhen.

3 Arbeitspakete

Die Entwicklung der App orientierte sich am Prinzip "Frontend First", um den Fokus auf die äußerst simple und unkomplizierte User-Experience zu legen, welches das USP für diese App gegenüber den anderen zahlreich vorhandenen Notiz-Apps ist. Dabei bildet das Frontend mit dem großen Textfeld als Hauptelement und das fortlaufende Speichern der Notizen im JavaScript Local Storage die Hauptaufgabe. Das Backend hingegen ist unkompliziert, es beschränkt sich auf einen Node.js-Server mit einem Endpunkt, der Anfragen mit optimiertem Prompt an die OpenAI API weiterleitet. Das zugehörige Prompt-Engineering stellt keine signifikante Herausforderung dar.

Arbeitspaket 1: Ideation + Erstellung der Mockups (1 Tag)

Aufgaben:

- Entwicklung der App-Idee inklusive Ziel, verwendete Services und Interface
- Erstellung von Mockups auf Miro
- Rücksprache mit Dozenten und Freigabe

Arbeitspaket 2: Einrichtung Entwicklungsumgebung (0.5 Tage)

Aufgaben:

- Verbindung mit DHBW GitLab herstellen, Repository clonen
- Ionic App und Node.js Backend initialisieren
- Git Verbindung verifizieren, erster Commit und Push

Arbeitspaket 3: Ionic-Frontend mit Textarea als Hauptseite (1 Tag)

Aufgaben:

- Fertigstellung der Homepage mit einer ion-textarea über die ganze Seite
- Notiz-Interface fertigstellen
- Fab Button als Organisieren-Button und mit richtigem Icon stylen
- Kontinuierliches Speichern der Notizen vom Textarea in Javascript Local Storage

Arbeitspaket 4: Node Backend mit Anbindung zur OpenAI API (1 Tag)

Aufgaben:

- Aufsetzen der index.js
- OpenAI API Key anlegen
- Verbindung mit GPT-4 Modell über OpenAI API herstellen + Testen in Postman

Arbeitspaket 5: Prompt-Engineering und Testing (GPT-4) (1 Tag)

Aufgaben:

- Test-Case mit unorganisierten Notizen erstellen
- Prompt-Engineering über OpenAI API für Organisation von Notizen
- Optimaler Prompt parametrisiert (für Sprache und Roh-Notizen) in index.js hinterlegen

Arbeitspaket 6: Anbindung Backend an Frontend (1.5 Tage)

Aufgaben:

- Erstellung des AI-Notes Services im Frontend
- Weitergabe der unsortierten Notizen aus Local Storage an Node Server
- Anzeige der sortierten Notizen auf der Homepage
- Implementierung von Entscheidungslogik für verschiedene Backend URLs (je nachdem, ob die App im Browser oder im Android Emulator läuft)
- Testen

Arbeitspaket 7: Finale Tests + Feinschliff (1 Tag)

Aufgaben:

- Letztes Styling der UI
- Code formatieren, Kommentare wo angebracht
- Finales Testen der App

Arbeitspaket 8: Dokumentation (3 Tage)

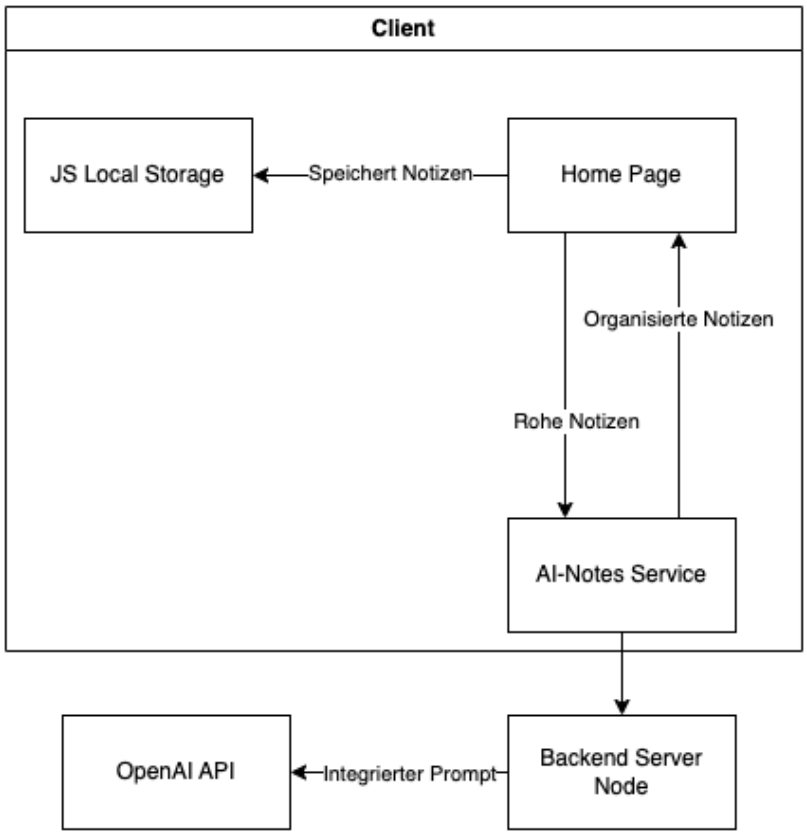
Aufgaben:

- Ziel und Funktionalität beschreiben
- UI-Konzept vorstellen
- Arbeitspakete vorstellen
- Umsetzung der App darlegen

Da dieses Projekt eine Einzelabgabe darstellt, wurden alle Arbeitspakete von dem einzigen Teammitglied Tien Dee Lin bearbeitet.

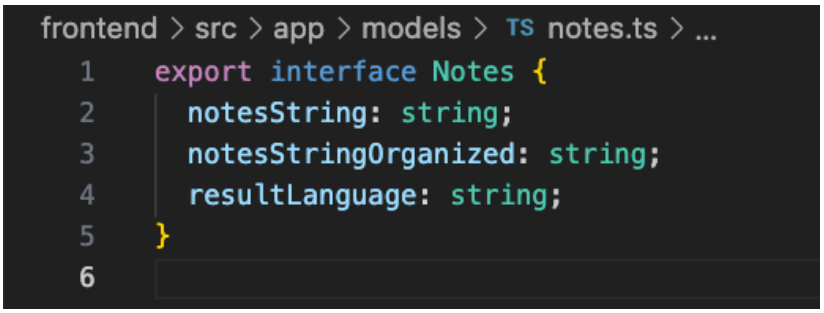
4 Umsetzung

4.1 Architektur



Die LazyNotes App besteht aus einem Ionic-Frontend mit einem „AI-Notes“ Service und einem Node-Backend, welches die OpenAI API aufruft. Die vom Nutzer eingegebenen Notizen werden kontinuierlich in den JavaScript Local Storage gespeichert. Die rohen Notizen werden (nach Drücken des Buttons) an den AI-Notes Service (im Code „ainotes.service“) übergeben, der je nachdem wie die App ausgeführt wurde, die richtige Server-URL auswählt und die rohen Notizen an den Node Server übergibt. Der Node-Server stellt die Anfrage an die OpenAI API mit den rohen Notizen in einen parametrisierten Prompt. Der Node Server selektiert aus der Antwort die organisierten Notizen und gibt diese an den AI-Notes Service zurück, der dann auf der Homepage anstelle der unorganisierten Notizen angezeigt wird.

4.2 Interface



Die LazyNotes App verwendet die „Notes“-Interface, die die aktuellen (meist unorganisierten) Notizen („notesString“), die zuletzt organisierten Notizen („notesStringOrganized“) und die Sprache, in der die organisierten Notizen sein sollen („resultLanguage“), enthält.

Dieses Interface kann um weitere Attribute erweitert werden, um die App um weitere Funktionen zu erweitern. Wie zum Beispiel eine Liste an Kategorien, in die der Nutzer die Notizen sortiert haben möchte. Hier muss jedoch abgewogen werden, ob vorgegebene Notiz-Kategorien nicht die volle Entfaltung des KI-Modells im Hintergrund behindern, da das Modell womöglich passendere und verständlichere Kategorien findet als der Nutzer. Zudem kann es sein, dass die Nutzer aus der Zielgruppe aufgrund ihrer Veranlagung zur Aufwandoptimierung die vorgegebenen Kategorien suboptimal definieren und diese auch nicht pflegen oder aktualisieren, was zur verschlechterten User-Experience führt.

4.3 Homepage

Die Homepage besteht aus einem „ion-textarea“ und einem „ion-fab-button“. Wichtig im „ion-textarea“ ist, dass der „autoGrow“ Parameter auf „true“ gesetzt ist, damit das Textfeld bei mehrzeiligem Input den Text wie gewohnt vertikal anzeigt. Ansonsten bleibt die Größe des Textfeldes auf ca. 2 Zeilen beschränkt und weitere Zeilen werden verdeckt und müssen durch manuelles Runterscrollen im Textfeld angezeigt werden. Zudem kann man im home.page.scss mit „—highlight-color-focused: none“ und „—highlight-color-valid: none“ den default vorhandenen blauen highlight-Unterstrich unter dem Textfeld entfernen welcher erscheint, wenn man in das Textfeld reinklickt. Jedoch muss dann die Cursor-Farbe manuell gesetzt werden, dies wird mit „carat-color“ erreicht.

Die „ion-textarea“ ist über two-way-binding mit dem „notesString“ Attribut des Notiz-Objektes synchronisiert. Das heißt, dass das „notesObject.notesString“ stets die aktuellen Notizen des Textfeldes beeinhaltet. Mit „(ngModelChange)=“storeLocalStoreNotes()“ wird die „storeLocalStoreNotes()“ Funktion immer aufgerufen wenn das Textfeld manipuliert wird z.B. Notizen hinzugefügt oder gelöscht werden.

Die „storeLocalStoreNotes()“ Funktion speichert bei Aufruf die aktuellen Notizen im Textfeld („notesString“) in den Javascript Local Storage. Der localStorage erlaubt Applikationen Daten

in Key-Value-Paare lokal im Browser zu speichern, sodass die Daten nicht verloren werden, wenn die App/Seite neu geladen wird. In der LazyNotes App wird bei jedem Initialisieren der Seite (zum Beispiel Neuladen) die „getLocalStoredNotes()“ Funktion aufgerufen, welche die im localStorage gespeicherten Notizen in das Attribut für die aktuellen Notizen „notesObject.notesString“ lädt. Über das Two-Way-Binding werden diese Notizen auch direkt im Textfeld angezeigt und können bearbeitet werden.

Nach dem Drücken des Organisieren-Buttons wird ein Loading-Screen angezeigt, der den Nutzer an allen Interaktionen hindert. Dieser Screen wird aufgelöst, wenn die organisierten Notizen zurückgegeben und im Textfeld dargestellt wurden. Dies hindert den Nutzer daran, während dem Organisieren-Prozess Notizen in das Textfeld einzutragen und diese verloren gehen, da das Textfeld mit den organisierten Notizen komplett überschrieben wird.

4.4 „ainotes“ Service

Wenn der Nutzer den „ion-fab-button“ („organize-notes“ Button) klickt, wird die „sendNotestoAI()“ Funktion aufgerufen, die die „organizeNotes()“ Funktion des ainotes-Services aufruft und dabei das aktuelle Notiz-Objekt als Übergabeparameter übergibt.

Der ainotes-Service ist Teil der Ionic App und stellt die Verbindung zwischen der UI und dem Node.js Server im Backend dar. „organizeNotes()“ ist die einzige Funktion in dem Service und sendet einen http Post-Request an dem Node.js Server (auf dem Pfad /askGPT), der mit der OpenAI API „verbunden“ ist. Bevor der Request gestellt wird, überprüft die „checkPlatform()“ Funktion ob die App mit Cordova, d.h. in unserem Fall im Android Emulator, gestartet wurde oder im Browser. Wenn die App im Browser läuft, wird die im Node Server definierte Server-URL localhost und Port 8080 gewählt. Wenn die App im Android Emulator läuft, muss die Hostadresse 10.0.2.2 mit Port 8080 gewählt werden, da bei Ausführung der App im Emulator der „localhost“ den „localhost“ des emulierten Handys adressieren würde und nicht den localhost des Computers. Der Android Emulator mappt per Default den localhost des Host-Rechners auf die Handy-Interne IP-Adresse „10.0.2.2“.

Damit LazyNotes im Emulator auch eine Verbindung zu dieser Adresse herstellen kann muss in LazyNotes/frontend/App/config.xml in Zeile 24 folgender Ausdruck stehen, um http-Kommunikation für die App zu ermöglichen:

```
22        <preference name="Scheme" value="http" />
23        <edit-config file="app/src/main/AndroidManifest.xml" mode="merge" target="/manifest/application" xmlns:android=
24            <application android:usesCleartextTraffic="true" />
25        </edit-config>
26        <resource-file src="resources/android/xml/network_security_config.xml" target="app/src/main/res/xml/network_sec
```

Dazu muss noch in der frontend/resources/android/xml/network_security_config.xml in Zeile 5 die Adresse 10.0.2.2 für http-Kommunikation freigeschaltet werden.

frontend > resources > android > xml > network_security_config.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config cleartextTrafficPermitted="true">
        <domain includeSubdomains="true">localhost</domain>
        <domain includeSubdomains="true">10.0.2.2</domain>
    </domain-config>
</network-security-config>
```

4.5 Node.js Server und Rückgabe der organisierten Notizen

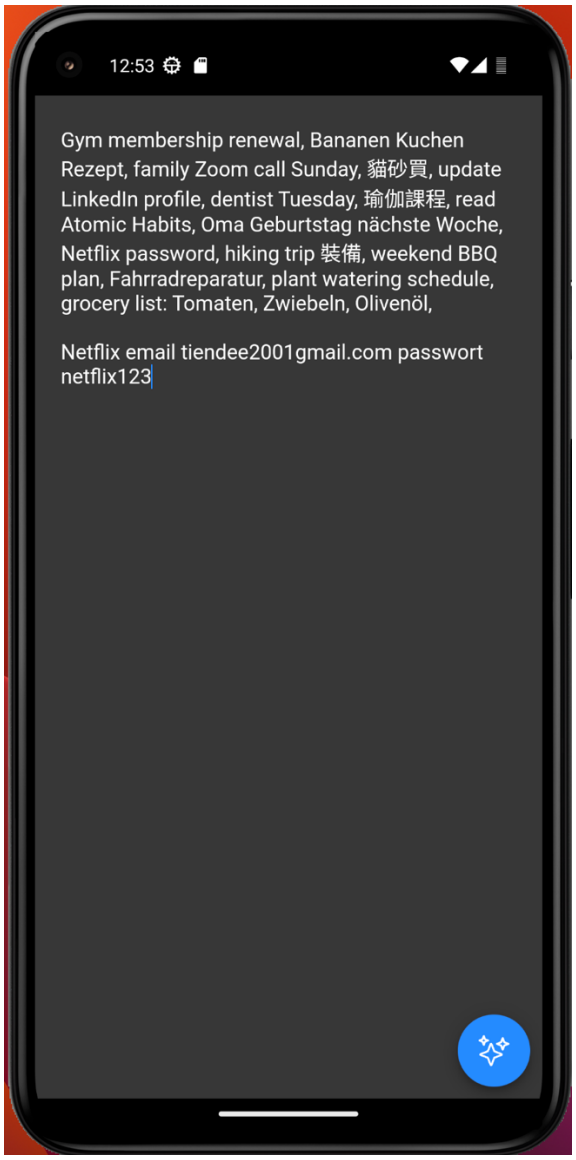
Der Node-Server im Backend wird lokal mit node index gestartet. Es läuft nicht in einem Container. Dieser Server bietet einen post-Endpoint bei dem Pfad /askGPT an, der von dem ainotes-Service aufgerufen wird und ein Notiz-Objekt im Request annimmt. Der Node-Server instanziiert eine Verbindung zur OpenAI mit einem festgelegten API-Key, welches in diesem Fall hart-kodiert ist, da für diesen Key ein fester Geldbetrag für API-Anfragen hinterlegt ist und der Dozent diese zum Testen der App benötigt.

In dem Pfad /askGPT wird ein Request an die OpenAI API geschickt, welches aus einem bereits optimierten Anweisungsprompt zur Notizorganisation besteht. Die rohen Notizen und die Sprache, in der die organisierten Notizen sein sollen, werden aus dem Request entnommen und an dafür vorgesehene Stellen im Anweisungsprompt eingeführt. Die organisierten Notizen werden aus der Response der API entnommen, in das „notesStringOrganized“ Attribut des Notiz-Objektes gespeichert und das gesamte Notiz-Objekt an den ainotes-Service zurückgegeben. Der ainotes-Service gibt das Notiz-Objekt wiederum an die aufrufende Methode („sendNotesToAI()“) im home.page.ts zurück, wo die organisierten Notizen aus der Response zuerst in das Attribut „notesStringOrganized“ des Notiz-Objektes der Homepage und dann in das „notesString“ des selben Objektes gespeichert wird.

Die organisierten Notizen werden doppelt gespeichert, da der „notesString“ ständig die aktuellen Notizen nach Änderungen des Nutzers darstellt und nur „notesStringOrganized“ bis zum nächsten Organisieren unverändert bleibt. Diese Struktur ist insbesondere für spätere Entwicklungszyklen notwendig, da es sein kann, dass der Nutzer die Notizen so lange nicht aktualisiert, dass die von OpenAI organisierte Notizen-Struktur aufgrund der stark veränderten und unorganisierten Notizen von der letzten organisierten Struktur stark abweicht. Um die letzte organisierte Struktur z.B. Kategorien wiederherzustellen, weil z.B. der Nutzer sich daran gewöhnt hat und diese bevorzugt, muss dem Request an die OpenAI API der letzte organisierte Zustand mit übergeben werden. Somit erreicht man über einen längeren Zeitraum eine geringere Varianz in der organisierten Notizstruktur und der Nutzer kann sich an z.B. stets ähnliche Kategorien gewöhnen.

4.6 Funktionsbeispiel

Rohe Notizen:



Organisierte und übersetzte Notizen, Original-Ausdrücke werden in Klammern beibehalten, um eventuellen Informationsverlust bei der Übersetzung zu vermeiden:


12:54
--To-Dos--
- Renew gym membership
- Update LinkedIn profile
- Schedule dental appointment on Tuesday
- Read Atomic Habits
- Prepare for grandma's birthday next week (Oma Geburtstag nächste Woche)
- Repair bike (Fahrradreparatur)
- Plan for weekend BBQ
- Create plant watering schedule
--Reminders--
- Family Zoom call on Sunday
- Buy cat litter (貓砂買)
- Yoga class (瑜伽課程)
- Prepare equipment for hiking trip (hiking trip 裝備)
--Recipes--
- Banana cake recipe (Bananen Kuchen Rezept)
--Login Information--
- Netflix: Email: tiendee2001@gmail.com, Password: netflix123
--Grocery List--
- Tomatoes (Tomaten)
- Onions (Zwiebeln)
- Olive oil (Olivenöl)

4.7 OpenAI API

Die OpenAI API ermöglicht Entwicklern den Zugriff auf fortschrittliche KI-Modelle von OpenAI. Durch die API können Anwendungen Anfragen an die KI senden und deren Antworten, wie hier in LazyNotes, direkt verwenden.

Um die OpenAI API zu nutzen, muss zuerst die OpenAI library installiert werden. Dazu navigiert man in den Ordner des Node-Servers und installiert es z.B. mit npm:

$ npm install --save openai

Als nächsten Schritt muss man einen OpenAI API-Key anlegen. Dafür muss ein OpenAI Account angelegt werden, diesen mit einer festen Geldmenge beladen und einen API-Key erzeugen: https://platform.openai.com/api-keys

Wenn das Setup abgeschlossen ist, wird im nächsten Schritt in der index.js die OpenAI Library importiert und eine Verbindung instanziiert. OpenAI empfiehlt aus Sicherheitsgründen den API-Key in den Umgebungsvariablen zu setzen. In diesem Projekt wird aus den genannten Gründen der Demonstration der API-Key hartcodiert:

```
const openai = new OpenAI({
    apiKey: "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",
});
```

Für den Anwendungsfall von LazyNotes wird eine sprachliche Antwort (organisierte Notizen) basierend auf einem sprachlichen Input (rohe Notizen + Anweisung) benötigt. Dafür eignet sich die „chat.completion“ Funktion der API, welches basierend auf einer Konversation eine Antwort liefert:

```
    const response = await openai.chat.completions.create({
        messages: [
            { role: "system", content: "You are a helpful assistant." },
            {role: "user", content: prompt}],
        model: "gpt-4",
        temperature: 0.7
      });

    notes.notesStringOrganized = response.choices[0].message.content;
    res.json(notes);
```

Neben der Konversations-Funktionalität (chat.completions) bietet die API weitere Funktionen an, wie zum Beispiel Transkriptionen von Audio-Dateien, Übersetzung von Text-Dateien oder die Erstellung von Embeddings (openai.embeddings.create). Die hier genutzte Funktion „chat.completions“ ist der Funktionsweise von ChatGPT am ähnlichsten.

Die API-Anfrage mit „chat.completions“ benötigt mindestens 2 Attribute im Request-Body: „messages“ und „model“.

„messages“ ist ein Array, das die vorangegangene Konversation enthält. Im ersten Objekt des Arrays wird die Anweisung übergeben, wie sich das System zu verhalten hat (role: „system“). Hier wird das System angewiesen, ein hilfreicher Assistent zu sein. Darauf folgt der Input des Nutzers mit einer Frage oder Anweisung (role: „user“). Im Fall von LazyNotes besteht die Anweisung aus dem zuvor definierten Prompt mit Roh-Notizen, Zielsprache und Organisationsanweisungen. Die übergebene Konversation kann beliebig lang sein, wobei eine Unterhaltung zwischen dem Nutzer und dem System simuliert werden kann. Diese vorangegangene Unterhaltung zu übergeben, hilft dem KI-Modell seine Aufgabe besser zu verstehen und sich selbst und somit die finale Antwort auf die gegebene Situation anzupassen.

„model“ ist ein string, der definiert, welches Modell zur Bearbeitung der Anfrage genutzt werden soll. Beispielsweise kann man wählen zwischen gpt-3.5, gpt-4-turbo-preview, gpt-4-vision-preview und viele weitere. OpenAI veröffentlich regelmäßig neue Modelle die besser sind und auf aktuellere Daten trainiert sind.

„temperature“ ist ein optionaler Parameter zwischen 0 und 2, der festlegt, wie kreativ das Modell in der Antwort sein soll. 0 ist am unkreativsten / fokussiert und deterministisch während 2 am kreativsten und zufälligsten ist. Nach dem Testen hat sich ergeben, dass für diesen Zweck gpt-4 mit einer Temperatur von 0.7 die besten Ergebnisse liefert. Es gibt viele weitere Attribute, die gesetzt werden können. Weitere Informationen sind in der Dokumentation verfügbar: https://platform.openai.com/docs/api-reference/chat/create

Das response-Objekt der „chat.completions“ Funktion hat folgende Struktur:


```
{
    "id": "chatcmpl-8qKhOSz6JUw2ld0NfHWmFGICj5Vlc",
    "object": "chat.completion",
    "created": 1707483314,
    "model": "gpt-4-0613",
    "choices": [
        {
            "index": 0,
            "message": {
                "role": "assistant",
                "content": "--To-Dos--\n- Kaugummis kaufen"
            },
            "logprobs": null,
            "finish_reason": "stop"
        }
    ],
    "usage": {
        "prompt_tokens": 181,
        "completion_tokens": 11,
        "total_tokens": 192
    },
    "system_fingerprint": null
}
```

Um das generierte Ergebnis des Modells, in diesem Fall die organisierten Notizen, zu erhalten, muss man im response-Objekt das erste Element des choices-Arrays selektieren und auf den content-Key des message-Objektes zugreifen:

```
notes.notesStringOrganized = response.choices[0].message.content;
res.json(notes);
```

Das Attribut „id“ weist jeder Anfrage einen Identifier zu. Das „usage“ Objekt zeigt, wie viele Tokens im Prompt und in der Antwort verwendet werden. Dies ist wichtig, um nachzuvollziehen, wie viel die API-Nutzung in der App kostet. Die Kosten belaufen sich bei gpt-4 Modellen auf 1-3 Cent pro 1000 Tokens, detaillierte Informationen sind erhältlich auf der OpenAI-Seite: https://openai.com/pricing#language-models

Detaillierte Informationen zu jedem Element im response-Objekt sind erhältlich in der offiziellen OpenAI Dokumentation: https://platform.openai.com/docs/api-reference/chat/object

Hinweis: ChatGPT ist für das Prompt-Engineering in der API-Nutzung nicht geeignet, da es andere Funktionsweisen hat und Einstellungen wie das gewünschte Modell oder Temperatur nicht verfügbar sind. Für das Prompt-Engineering zur API-Nutzung eignet sich nur die API, das heißt, dass die Inferencing-Kosten des Prompt-Engineering ebenfalls im Pay-Per-Use Modell in den Entwicklungskosten einberechnet werden müssen. Das OpenAI-Premium Abonnement für 20 Euro im Monat enthält kein kostenloses API-Inferencing.

5 Anleitung zum Starten der App

Terminal:

1. **Git Respository clonen:**

   $ git clone git@git.dhbw-stuttgart.de:software-engineering-wwi2021f/assignment-gruppe-6.git

2. **In den Backend-Ordner wechseln:**
   $ cd assginment-gruppe-6/backend

3. **Node Packages des Servers installieren:**
   $ npm install

4. **Node Server starten:**
   $ node index

5. **Neues Terminal öffnen**

6. **In den frontend-Ordner navigieren, z.B:**
   $ cd desktop/ assginment-gruppe-6/frontend

7. **Node Packages des Frontends installieren:**
   $ npm install

Zum Starten der App im Browser:

$ ionic serve

Zum Starten der App im Android Emulator:

1. Sicherstellen, dass das emulierte Handy gestartet und verfügbar ist
2. $ ionic cordova run android

Zum Testen der App kann dieser unorganisierte Notiz-String verwendet werden:

Gym membership renewal, Bananen Kuchen Rezept, family Zoom call Sunday, 貓砂買, update LinkedIn profile Login Netflix email: frank.mueller123@gmail.com passwort: passwort123

dentist Tuesday, 瑜伽課程, read Atomic Habits, Oma Geburtstag nächste Woche, Netflix password, hiking trip 裝備, weekend BBQ plan, Fahrradreparatur, plant watering schedule, grocery list: Tomaten, Zwiebeln, Olivenöl

Heute Abend Müll rausbringen!!!! Kokosmilch Kaufen