Dokumentation

SE3-Projekt Stundenplan

1. Projektbeschreibung

Das Projekt ist ein webbasierter Stundenplan für Studierende, der die Verwaltung von Kursplänen unterstützen soll. Es bietet Funktionen wie Benutzerauthentifizierung, Auswahl von Stundenplänen, klickbare Termine und eine Profilansicht, sowie (später geplante) Hinzufügen von Kursen, Benachrichtigungen und Datenexporte.

Das Backend basiert auf dem Python-Framework Flask und verwendet eine MySQL-Datenbank zur Speicherung der Daten. Benutzer müssen sich per Login authentifizieren und können anschließend über geschützte API-Endpunkte mit dem System interagieren.

Das Frontend basiert auf dem React Web-Framework Next.js. Wir benutzen Nextauth für den Login- und Logout-Prozess und die Sessionverwaltung. Im Frontend können sich die Benutzer registrieren, anmelden und dann ihren Stundenplan auswählen und diesen im Kalender besichtigen. Jeder Termin im Kalender ist anklickbar und zeigt mehr Informationen an. Man kann einen neuen Stundenplan erstellen und löschen.

2. Wie wird das Projekt gestartet?

Das Projekt kann unterschiedlich gestartet werden:

- Zunächst müssen natürlich alle Bibliotheken sowie Dependencies installiert werden. Das kann ganz einfach mit make install gemacht werden. Dieses Script stellt sicher, dass alle Dependencies installiert sind und dass venv initialisiert wurde. Nachdem dann die Datenbank konfiguriert wurde und das Datenbank-Setup abgeschlossen wurde, kann man das Projekt im Root-Verzeichnis mit npm run dev starten. Dabei werden Frontend und Backend parallel gestartet.
- Falls auf die Benutzung von dem Makefile verzichtet wird, dann muss man venv und alle Dependencies selber installieren. Dazu gibt es eine Anleitung im README.

3. Hauptarchitektur & Prinzipien

Frontend:

- Modularisierung durch Components und vorgefertigte Components von shadcn (Production-Ready Components)
- Da wir Components benutzen, hat es am meisten Sinn gemacht das Styling mit Tailwind zu machen, da man dann die Stylings immer Component-basiert hat
- Jede Seite hat eine eigene Datei, in der Components dann wiederverwendet werden können (Next.js Page Router)

- API Calls zum Backend können sowohl in Components als auch Seiten gemacht werden (könnten auch in einem eigenen API Ordner angelegt werden)

Backend:

- Modularisierung über Flask Blueprints
  → Jede Route-Gruppe wie `auth`, `timetable`, `health` ist in einer eigenen Datei organisiert.

- JWT-Authentifizierung
  → Nur eingeloggte Nutzer können Stundenpläne sehen/bearbeiten (`@jwt_required()`)

- ORM mit SQLAlchemy
  → Datenbankabfragen sind objektorientiert, kein direktes SQL notwendig

- REST-Prinzipien
  → Die API folgt REST-Standards: `GET`, `POST`, `PUT`, `DELETE`

- Fehlerbehandlung mit Try/Except
  → Jede Route gibt im Fehlerfall eine JSON-Fehlermeldung mit Statuscode zurück

4. Wie wurden diese Prinzipien umgesetzt?

Frontend

- Jede UI-Einheit (Card, Button, Input, etc.) lebt als eigene React-Komponente unter `src/components/ui` und wird über shadcn/ui (production-ready) importiert.
- **Tailwind-Styling:** Jeder Component erhält seine Klassen direkt im JSX (`className="p-4 space-y-6"`), so bleibt Styling strikt lokal und übersichtlich.

Backend

- In `timetable.py` sieht man klar:
  - REST-Endpunkte (`/`, `/<id>`, `/active`, `/duplicate`)
  - Datenbankoperationen per `Model.query` und `db.session`
  - Sicherung durch Token mit `@jwt_required()`
- In `health.py`:
  - Einfache Monitoring-Logik
  - Prüfung der DB-Verbindung
  - JSON-Antwort mit Status & Timestamp
- Datenbank-Setup mit lokaler MySQL und API Anbindung via SQLAlchemy & Flask
- Error-Handling via error_handler.py, welcher Statuscodes zurückgibt

5. Bekannte Probleme (Known Issues)

Aufgrund eines groß angesetzten Projektumfangs konnten einige Funktionen bislang noch nicht umgesetzt werden.

6. Was würde man anders machen?

Ursprünglich haben wir React Router benutzt, entschieden uns dann aber für einen Wechsel zu Next.js, da sich die ursprüngliche Lösung als zu komplex für unsere Anforderungen erwies. Rückblickend hätte eine realistische Projektplanung sowie eine gründlichere Vorabrecherche zum tatsächlichen Entwicklungsaufwand geholfen, solche Herausforderungen frühzeitig zu erkennen und besser zu adressieren.

7. Individuelle Reflektion

Was habe ich gelernt?

- Robin: Die Erstellung und Implementierung von API Endpoints war neu für mich. Außerdem habe ich gelernt, lokale Datenbanken mit localhost Applikationen zu verbinden.
- Tim: Ich habe gelernt, APIs mit Flask ins Frontend zu integrieren, da ich das vorher nie mit einem Python Backend gemacht habe.
- Laura: Ich habe gelernt, mit Python zu programmieren und API Endpoints zu definieren.
- Leo: Die Auseinandersetzung mit Python und die Zusammenarbeit in einem größeren Team waren für mich neu; besonders das Konzept von Blueprints, Routen und HTTP-Methoden war anfangs schwer zu verstehen
- Luca: Noch genauer verstanden wie man Styles in CSS schreibt, wie man API Routen definiert und Errorhandling für API Requests
- Jannes: Ich habe gelernt, wie Python Syntax funktioniert und die Unterschiede zu Java kennengelernt. Außerdem, wie man eine relationale MySQL-Datenbank für Kurse, Dozenten, Räume und Zeiten modelliert und einrichtet. Neue Git
- Yazan: Ich habe ein bisschen über Python und React-Komponenten gelernt. Außerdem kann ich jetzt besser mit Git arbeiten.

Wo habe ich noch Lücken?

- Robin: Bei der Verbindung von Backend und Frontend.
- Tim: Mir fällt es schwer, im Code und beim Arbeiten konsistent zu bleiben.
- Laura: In der Erstellung von Frontends.
- Leo: In Themen, mit denen ich weniger intensiv gearbeitet habe, habe ich noch Lücken, und es ist manchmal eine Herausforderung, die Motivation konstant aufrechtzuerhalten
- Luca: Komplettes Backend zusammenhängend zu verstehen welche Datei genau was macht. Um Probleme und Lösungsfindung zu optimieren
- Jannes: Selber, durch längeres Überlegen die Lösung zu finden, da ich schnell jemand anderes Frage oder im Internet nach einer Lösung suche. Wie man mit Flask die Datenbank mit dem Frontend verbindet, habe ich anfangs nicht verstanden.
- Yazan: In Python ist mir die Datenbankanbindung (z. B. mit SQLAlchemy) noch nicht ganz klar. Daten aus einer API zu holen und richtig zu verarbeiten fand ich etwas kompliziert.

Wo bin ich über meinen Schatten gesprungen?

- Robin: Auch weil das Thema neu für mich war, fand ich es schwierig, die API Endpoints richtig zu implementieren.
- Tim: Letztendlich arbeitet man im Studium mit Kommilitonen zusammen, die alle einen unterschiedlichen Wissensstand haben. Das muss man berücksichtigen und war zuerst schwierig zu begreifen.
- Laura: Das Arbeiten mit so vielen für mich neuen Themen und gleichzeitig mit so einer großen Gruppe war schwer unter einen Hut zu bekommen und einen Anschluss zu finden. Am Ende hat alles dann ganz gut funktioniert, also habe ich es geschafft, dort über meinen Schatten zu springen.
- Leo: Ich bin über meinen Schatten gesprungen, indem ich mich trotz der vielen neuen Themen nicht habe stressen lassen, sondern mich Schritt für Schritt in die Materie eingearbeitet habe.
- Luca: Python genauer zu verstehen und meine Kenntnisse von Java auf Python zu übertragen
- Jannes: Bei der Erstellung von Inhalten, von denen ich anfangs nicht wusste, wie ich sie umsetzen soll, da ich es noch nie in der Praxis gemacht habe (z.B Implementation der Datenbank).
- Yazan: Ich habe versucht, Probleme selbst zu lösen, statt sofort jemanden um Hilfe zu bitten. Ich habe Python besser verstanden, als ich vorher gedacht hätte.