Handout

1. **Projektnamen**
   Deskflex

2. **Beschreibung**
   Ein System zur Verwaltung von Büroplätzen, das die Zuweisung von Arbeitsplätzen erleichtert und die Transparenz fördert.

3. **Aufwandsstatistiken**
   a. **Arbeitsstunden pro Person und Hauptbeitrag pro Person**
      - **Filpe Jannik**
        25 bis 30h
        Doku
      - **Mürle Moritz**
        9h
        Doku
      - **Niederer Christoph**
        44 bis 48h
        CI/CD + Frontend + Doku
      - **Rother Leon**
        15 bis 18h
        Doku + Tests
      - **Zagola Hagen**
        70h
        Backend + Frontend

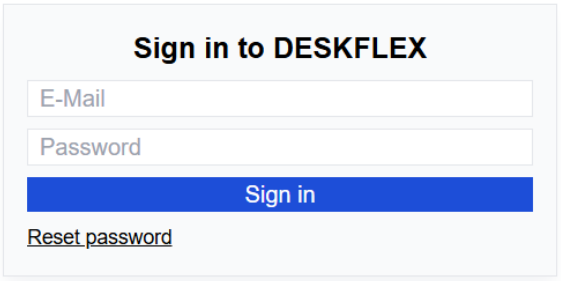
   b. **Arbeitsstunden pro Workflow**
      Requirement Analyse: 30h
      Projektmanagment: 25h
      Entwicklung: 70h

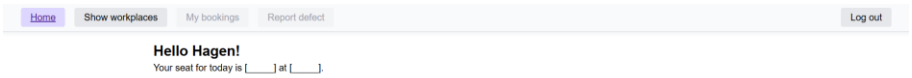
   c. **Arbeitsstunden pro Phase**
      Vorarbeit: 20h
      Entwicklungsphase: 70h
      Projektmanagmentsphase: 25h
      Evaluation: 10h

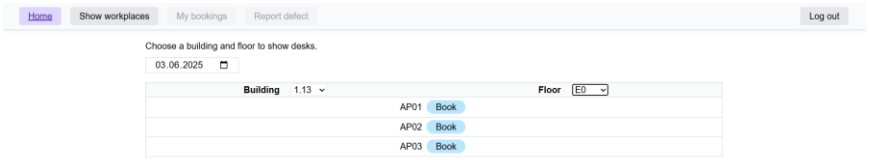
4. **Highlights eurer Demo (Beschreibung + Screenshots)**
   Unser Projekt **Deskflex** bietet eine intuitive Webanwendung zur flexiblen Arbeitsplatzbuchung in Unternehmen. Die Demo zeigt den typischen Ablauf für Benutzer:innen – vom Login bis zur Buchung eines Platzes – mit Fokus auf Benutzerfreundlichkeit und Übersichtlichkeit.
   Login und Übersichtsseite
   Nach dem erfolgreichen Login gelangt man auf eine persönliche Übersichtsseite, auf der z. B. aktuelle Buchungen angezeigt werden. Diese dient als zentraler Einstiegspunkt in die Anwendung.
   Navigationsmenü & Buchung
   Über die Menüleiste kann der Bereich **"Platz buchen"** aufgerufen werden.
   Nutzer:innen werden auf eine separate Seite weitergeleitet, auf der sie die folgenden Filteroptionen haben:
   - **Datumsauswahl** (Kalender)

- **Gebäudeauswahl** (Dropdown)
- **Etagen-/Floor-Auswahl** (Dropdown)

Platzübersicht und Buchung
Basierend auf den gewählten Kriterien wird eine übersichtliche **Tabelle mit verfügbaren Arbeitsplätzen** angezeigt. In dieser Tabelle sind buchbare Plätze entsprechend gekennzeichnet, sodass man mit einem Klick eine Buchung vornehmen kann.







---

5. Higlights eueres Projekts:

a. Architektur

- Architekturstil: Client-Server-Architektur
  - Argumente:
  - Klare Trennung zwischen Frontend (Vue.js) und Backend (ASP.NET), wodurch die Entwicklung und Wartung modularisiert wird.
  - Ermöglicht einfache Erweiterbarkeit und Skalierbarkeit.

- Datenbankdesign: Relationale Datenbank (SQLite)
  - Argumente:
  - Leichtgewichtig und einfach zu implementieren für kleinere Projekte.
  - Gute Integration mit dem ASP.NET-Framework.

b. Software Tools/Plattform/Technik/Libaries

- Entwicklungsumgebung: Visual Studio Code, Visual Studio
- Programmiersprachen: C# (Backend), JavaScript (Frontend)
- Frameworks:
  - ASP.NET Core für das Backend
  - Vue.js für das Frontend
  - TailwindCSS für das Styling
- Datenbank: SQLite
- Versionskontrolle: Git (GitHub Repository)

c. Datenbank Design

Für die Anwendung wurde eine relationale SQLite-Datenbank verwendet. Die zentrale Struktur besteht aus den Tabellen **Buildings**, **Floors** und **Desks**, welche logisch miteinander verknüpft sind.
Ein **Building** kann mehrere **Floors** besitzen, und jeder **Floor** enthält mehrere **Desks**. Die Relationen wurden über Foreign Keys modelliert.
Die Datenbank ist einfach gehalten, um eine klare und performante Abfrage von verfügbaren Arbeitsplätzen nach Gebäude und Stockwerk zu ermöglichen.

d. Testing

Zur Qualitätssicherung wurden gezielt **Unit Tests** geschrieben. Diese testen einzelne Komponenten isoliert, insbesondere im Backend. Dadurch konnte sichergestellt werden, dass zentrale Funktionen wie Datenbankzugriffe und Logikmethoden erwartungsgemäß arbeiten. Die Unit Tests wurden regelmäßig durch die CI/CD Pipeline ausgeführt, um Regressionen frühzeitig zu erkennen und die Stabilität der Anwendung zu gewährleisten.

e. Metriken

Für die Bewertung der Codequalität wurden verschiedene Metriken herangezogen:

- **Code Smells**: Verdächtige Stellen im Code wurden identifiziert, z. B. überflüssige Bedingungen oder unklare Verantwortlichkeiten.
- **Cyclomatic Complexity**: Komplexe Methoden wurden auf einfache Rückgabe- und Kontrollstrukturen hin optimiert.
- **Code Duplication**: Wiederholter Code wurde reduziert, um Wartbarkeit und Klarheit zu verbessern.

Diese Metriken halfen dabei, die Lesbarkeit und Wartbarkeit des Codes gezielt zu steigern.

f. CI/CD

Ein GitHub-Actions-basierter CI/CD-Workflow wurde eingerichtet. Bei jedem Push wird automatisch ein Build durchgeführt und eine statische Codeanalyse ausgeführt. So wird sichergestellt, dass Änderungen keine offensichtlichen Fehler verursachen und grundlegende Qualitätsrichtlinien eingehalten werden. Dieser Prozess fördert eine kontinuierliche Integration und Qualitätssicherung im Projektverlauf. Die Tests werden durchgeführt und Artefakte erstellt.