Projektbeschreibung und Dokumentation für das MOCO UI LAP Projekt

1. Einführung

Dieses Projekt ist eine **Teamverwaltungsanwendung** für Teamleiter, die es ermöglicht, die Anwesenheitszeiten und Arbeitsorte ihrer Teammitglieder direkt über eine Weboberfläche zu verwalten. Die Anwendung ermöglicht es, Arbeitszeiten zu bearbeiten und den Arbeitsort (Homeoffice oder Büro) für jedes Teammitglied festzulegen. Daten werden über eine API-Verbindung abgerufen und aktualisiert.

Das Ziel dieses Projekts ist es, eine leicht bedienbare Lösung bereitzustellen, die Teamleitern die Organisation der Arbeitszeiten ihrer Teammitglieder erleichtert.

---

2. Anforderungen

Benutzeroberfläche (UI):

- **Teammitglieder-Auswahl**: Eine übersichtliche Liste der Teammitglieder wird angezeigt. Die Teamleiter können ein Mitglied auswählen, um dessen Arbeitszeiten und Arbeitsort zu bearbeiten.
- **Arbeitszeitbearbeitung**: Teamleiter können die Arbeitszeiten (Start- und Endzeiten) anpassen, indem sie die Zeiten direkt in der Tabelle bearbeiten.
- **Festlegung des Arbeitsorts**: Der Arbeitsort (Homeoffice oder Büro) kann über eine Checkbox ausgewählt werden. Wird die Checkbox aktiviert, wird das Teammitglied als im Homeoffice arbeitend markiert; andernfalls wird Büroarbeit angenommen.

Technologie-Stack und Tools:

- **Frontend-Framework**: Die Anwendung wurde mit **Angular 18** entwickelt, was eine moderne, reaktive Benutzeroberfläche bietet.
- **Angular Material**: Für die Gestaltung und das responsive Design der Benutzeroberfläche wurden Angular Material-Komponenten wie Tabellen, Buttons und Checkboxen verwendet.
- **Versionierung und Quellcodeverwaltung**: **Git** wurde verwendet, um den Quellcode zu verwalten und die Zusammenarbeit im Team zu erleichtern.
- **API-Integration**: Die Anwendung verwendet eine externe API, um Daten zu Teammitgliedern, Anwesenheitszeiten und Arbeitsorten abzurufen und zu aktualisieren. Der API-Schlüssel wird vom Benutzer eingegeben und im **localStorage** gespeichert.

---

3. Technische Details

Verwendete Technologien:

- **Angular 18**: Ein Framework, das für die Erstellung von Single Page Applications (SPA) verwendet wird. Angular bietet eine reaktive Programmierstruktur und erleichtert die Komponenten-basierte Entwicklung.
- **Angular Material**: Eine UI-Bibliothek, die vorgefertigte Komponenten wie Tabellen, Schaltflächen und Eingabefelder bietet. Diese Bibliothek wurde verwendet, um eine moderne und benutzerfreundliche Oberfläche zu erstellen.
- **LocalStorage**: Der API-Schlüssel des Benutzers wird sicher im **localStorage** des Browsers gespeichert, um persistente Sitzungen zu ermöglichen.

Hauptfunktionen:

- **Benutzerverwaltung**: Teamleiter können ihre Teammitglieder in einer Liste anzeigen, auswählen und deren Arbeitszeiten und Arbeitsort bearbeiten.
- **Zeiterfassung**: Die Arbeitszeiten der Teammitglieder können über Eingabefelder geändert werden. Die Änderungen werden in der API gespeichert.
- **Homeoffice-Option**: Der Arbeitsort kann über eine **Checkbox** festgelegt werden. Diese Option ermöglicht es, schnell festzulegen, ob das Teammitglied von zu Hause oder im Büro arbeitet.

---

4. Implementierung

4.1. Benutzerverwaltung

Die Benutzerverwaltung erfolgt über eine Liste, die alle Teammitglieder anzeigt. Die Teamleiter können ein Mitglied auswählen, um dessen Anwesenheitsdaten zu bearbeiten.

```
<div class="team-list">
  <h2 class="title">Teammitglieder</h2>
  <mat-list class="team-list-container">
    <mat-list-item *ngFor="let user of users" class="team-card"
(click)="selectUser(user)">
      <div class="team-info">
        <img mat-list-avatar [src]="user.avatar_url || 'default-
avatar.jpg'" alt="{{ user.firstname }}'s Avatar" />
        <div class="user-details">
          <span class="user-name">{{ user.firstname }} {{ user.lastname
}}</span>
          <span class="user-role">{{ user.tags || 'Teammitglied' }}</span>
        </div>
      </div>
    </mat-list-item>
  </mat-list>
</div>
```

4.2. Anwesenheitsverwaltung

Die Anwesenheitsverwaltung wird in einer Tabelle umgesetzt. Teamleiter können Start- und Endzeiten für jedes Teammitglied bearbeiten. Zudem gibt es eine Checkbox zur Festlegung des Arbeitsorts (Homeoffice oder Büro).

```
<table mat-table [dataSource]="presences" class="mat-elevation-z8">
  <ng-container matColumnDef="date">
    <th mat-header-cell *matHeaderCellDef>Datum</th>
    <td mat-cell *matCellDef="let presence">{{ presence.date | date:
'dd.MM.yyyy' }}</td>
  </ng-container>

  <ng-container matColumnDef="from">
    <th mat-header-cell *matHeaderCellDef>Von</th>
    <td mat-cell *matCellDef="let presence">
      <ng-container *ngIf="!presence.isEditing; else editFrom">
        {{ presence.from }}
      </ng-container>
      <ng-template #editFrom>
        <input [(ngModel)]="presence.from" type="time" />
      </ng-template>
    </td>
  </ng-container>

  <ng-container matColumnDef="to">
    <th mat-header-cell *matHeaderCellDef>Bis</th>
    <td mat-cell *matCellDef="let presence">
      <ng-container *ngIf="!presence.isEditing; else editTo">
        {{ presence.to }}
      </ng-container>
      <ng-template #editTo>
        <input [(ngModel)]="presence.to" type="time" />
      </ng-template>
    </td>
  </ng-container>

  <ng-container matColumnDef="isHomeOffice">
    <th mat-header-cell *matHeaderCellDef>Home Office</th>
    <td mat-cell *matCellDef="let presence">
      <ng-container *ngIf="!presence.isEditing; else editHomeOffice">
        <mat-checkbox [checked]="presence.is_home_office" disabled></mat-
checkbox>
      </ng-container>
      <ng-template #editHomeOffice>
        <mat-checkbox [(ngModel)]="presence.is_home_office"></mat-checkbox>
      </ng-template>
    </td>
  </ng-container>
</table>
```

4.3. Authentifizierung und API-Schlüssel-Verwaltung

Die Authentifizierung erfolgt über die Eingabe eines API-Schlüssels, der vom Benutzer manuell in ein Textfeld eingegeben wird. Der API-Schlüssel wird im **localStorage** gespeichert, damit die Authentifizierung zwischen den Sitzungen erhalten bleibt.

```
<div class="profile-container">
  <img *ngIf="playingCaptain" [src]="profileImageUrl" alt="Profil"
class="profile-image" (click)="toggleApiKeyForm()" />

  <div class="api-key-form" *ngIf="isApiKeyFormVisible">
    <input type="text" placeholder="API-Key eingeben" [(ngModel)]="apiKey"
/>
    <button (click)="saveApiKey()">Speichern</button>
    <button (click)="logout()">Abmelden</button>
  </div>
</div>
```

---

5. Herausforderungen

5.1. API-Integration

Eine der größten Herausforderungen bestand darin, die API-Integration mit dynamischer Verwaltung des API-Schlüssels umzusetzen. Der Schlüssel wird im **localStorage** gespeichert, um sicherzustellen, dass die Benutzer über mehrere Sitzungen hinweg authentifiziert bleiben.

Lösung: Der API-Schlüssel wird nach erfolgreicher Eingabe im lokalen Speicher des Browsers abgelegt und bei jeder Anfrage automatisch an die API übergeben.

5.2. Benutzeroberfläche und Nutzererfahrung

Eine weitere Herausforderung bestand darin, eine benutzerfreundliche Oberfläche zu entwickeln, die es den Teamleitern ermöglicht, die Anwesenheitsdaten ihrer Teammitglieder einfach zu verwalten. Dies erforderte die Implementierung von **bearbeitbaren Tabellen** und einer **Checkbox-Logik** für den Arbeitsort (Homeoffice oder Büro).

---

6. Tests

6.1. Unit-Tests

Die wichtigsten Funktionen, wie die Bearbeitung der Anwesenheitszeiten und die Speicherung der Daten über die API, wurden mit **Unit-Tests** überprüft, um sicherzustellen, dass die Kernlogik fehlerfrei funktioniert.

6.2. Manuelles Testing

Manuelles Testing wurde eingesetzt, um sicherzustellen, dass die Benutzeroberfläche fehlerfrei funktioniert und die Benutzerfreundlichkeit gewährleistet ist. Besonders die **API-Schlüssel-Verwaltung** und die **Bearbeitung der Anwesenheitsdaten** wurden in verschiedenen Szenarien getestet.

---

7. Fazit

Dieses Projekt bietet eine **benutzerfreundliche Lösung** zur Verwaltung von Anwesenheitszeiten und Arbeitsorten für Teamleiter. Die Verwendung von **Angular** und **Angular Material** ermöglicht eine moderne, reaktionsschnelle Benutzeroberfläche, die den Teamleitern eine einfache Handhabung ermöglicht.

Verbesserungsmöglichkeiten:

- **Echtzeit-Aktualisierungen**: Die Implementierung von WebSockets oder ähnlichen Technologien könnte es ermöglichen, dass Änderungen in den Anwesenheitszeiten in Echtzeit für alle Teamleiter sichtbar werden.
- **Erweiterte Fehlerbehandlung**: Eine verbesserte Rückmeldung bei API-Fehlern und detaillierte Fehlerberichte würden die Benutzerfreundlichkeit weiter erhöhen.