SRS.md – Anforderungen (Frontend) + Swagger-Anhang

ToDo-App – Frontend (LF 10a)

Dieses Dokument enthält nur die Anforderungen an das Frontend. Die vollständige Swagger/OpenAPI-Dokumentation ist als Anhang enthalten und gilt als verbindlicher API-Vertrag.

---

1. Ziel und Kontext

1.1 Ziel der Anwendung

Das Frontend ist eine moderne Single-Page-Application (SPA), die eine intuitive Benutzeroberfläche für die Verwaltung von Aufgaben (To-dos) bereitstellt. Es ermöglicht Mitarbeitern die Einsicht und Bearbeitung ihrer Aufgaben, Abteilungsleitern die Zuweisung und Verwaltung von Aufgaben an ihre Teams sowie Administratoren die Benutzerverwaltung. Die Anwendung kommuniziert über eine REST-API mit dem Backend und ist vollständig responsiv für Desktop, Tablet und Smartphone.

1.2 Rollen im System

Mitarbeiter:

- Können sich am System anmelden und ihre zugewiesenen Aufgaben einsehen
- Können Aufgaben als erledigt/unerledigt markieren
- Können Aufgaben nach Projekt und Priorität sortieren/filtern

Abteilungsleiter:

- Verfügen über alle Mitarbeiter-Rechte
- Können Aufgaben aller Mitarbeiter ihrer Abteilung einsehen
- Können neue Aufgaben erstellen und Mitarbeitern zuweisen
- Können neue Projekte anlegen und verwalten
- Können Prioritäten verwalten

Administrator:

- Können Benutzerkonten anlegen, bearbeiten und löschen
- Können Rollen an Benutzer vergeben und ändern
- Haben Zugriff auf die Benutzerverwaltung

1.3 Aktualisierung der Datenbestände

Da mehrere Clients auf das gleiche Backend zugreifen können ist dafür zu sorgen, dass ein Mechanismus vorhanden ist, der die Datenbestände auf dem Frontend aktuell hält. Es soll sichergestellt werden, dass Änderungen an Aufgaben, Projekten oder Prioritäten von einem Benutzer sofort bei anderen Benutzern sichtbar sind, ohne dass diese die Seite manuell aktualisieren müssen. Hierfür kann beispielsweise ein Polling-Mechanismus oder WebSockets verwendet werden, um Echtzeit-Updates zu ermöglichen.

2. Technischer Stack (für die KI-Umsetzung verbindlich)

Programmiersprache:

- TypeScript 5.x (strikte Typisierung für bessere Wartbarkeit und Fehlerprävention)

Framework / Architektur:

- React 18.x mit funktionalen Komponenten und React Hooks
- React Router v6 für clientseitiges Routing
- Context API oder Zustand für globales State Management (z.B. Auth-Context)
- Komponentenbasierte Architektur mit klarer Trennung von UI und Logik

Build-Tool / Projekt-Setup:

- Vite als Build-Tool und Development Server (schneller als CRA)
- ESLint + Prettier für Code-Qualität und Formatierung
- Ordnerstruktur:

```
src/
├── components/     # Wiederverwendbare UI-Komponenten
├── pages/          # Seitenkomponenten (Views)
├── hooks/          # Custom React Hooks
├── context/        # React Context Provider
├── services/       # API-Service Layer
├── types/          # TypeScript Interfaces/Types
├── utils/          # Hilfsfunktionen
└── assets/         # Statische Ressourcen
```

Styling / UI-Konzept:

- Tailwind CSS 3.x als Utility-First CSS Framework
- Mobile-First Responsive Design Ansatz
- Konsistente Design-Tokens für Farben, Abstände und Typografie
- Dark Mode Unterstützung (optional)
- Eigene Tailwind-Konfiguration für projektspezifische Design-Tokens

Weitere Libraries (Name + Zweck):

- **axios** (^1.6.x): HTTP-Client für API-Requests mit Interceptors für Token-Handling
- **react-router-dom** (^6.x): Clientseitiges Routing und Navigation
- **react-hook-form** (^7.x): Performante Formularverwaltung mit Validierung
- **zod** (^3.x): Schema-Validierung für Formulare und API-Responses
- **@tanstack/react-query** (^5.x): Server-State Management, Caching und automatische Refetches
- **clsx** oder **tailwind-merge**: Bedingte CSS-Klassen Zusammenführung
- **lucide-react** oder **heroicons**: Icon-Library (SVG-basiert, Tree-Shakeable)
- **date-fns** (^3.x): Datums-Formatierung und -Manipulation (leichtgewichtig)
- **react-hot-toast** oder **sonner**: Toast-Benachrichtigungen für Feedback

3. Funktionale Anforderungen (nur Anforderungen)

Hinweis: Die API ist im Swagger-Anhang beschrieben. In diesem Kapitel werden nur Anforderungen beschrieben: Was muss das Frontend können?

3.1 Authentifizierung

FR-01 Login Beschreibung (UI-Verhalten):

- Login-Formular mit Feldern für Benutzername und Passwort
- Submit-Button mit Loading-State während der Authentifizierung
- Nach erfolgreichem Login: JWT-Token im localStorage/sessionStorage speichern und Weiterleitung zum Dashboard
- Automatische Weiterleitung zum Dashboard bei bestehendem gültigem Token

Akzeptanzkriterien:

- ☐ Beide Felder (Benutzername, Passwort) sind Pflichtfelder
- ☐ Submit-Button ist nur aktiv, wenn beide Felder ausgefüllt sind
- ☐ Nach erfolgreicher Anmeldung wird der Benutzer zum rollenspezifischen Dashboard weitergeleitet
- ☐ JWT-Token wird sicher gespeichert und bei API-Requests im Authorization-Header mitgesendet
- ☐ Bei Seiten-Refresh bleibt der Benutzer eingeloggt (Token-Persistenz)

Fehlerverhalten (z. B. falsche Zugangsdaten / 401):

- Bei 401: Fehlermeldung "Benutzername oder Passwort falsch" unter dem Formular anzeigen
- Passwort-Feld wird geleert, Benutzername bleibt bestehen
- Bei Netzwerkfehler: Fehlermeldung "Verbindung zum Server fehlgeschlagen. Bitte versuchen Sie es später erneut."
- Bei Token-Ablauf: Automatischer Logout und Weiterleitung zur Login-Seite

---

3.2 Anforderungen Mitarbeiter

FR-02 Aufgabenübersicht (eigene Aufgaben) Beschreibung:

- Tabellarische oder Karten-Ansicht aller dem Mitarbeiter zugewiesenen Aufgaben
- Anzeige von: Titel, Projekt, Priorität, Status (erledigt/offen), Fälligkeitsdatum (falls vorhanden)
- Visuelle Unterscheidung zwischen offenen und erledigten Aufgaben (z.B. Durchstreichung, Farbcodierung)
- Prioritäten werden farblich codiert (z.B. Hoch=Rot, Mittel=Gelb, Niedrig=Grün)

Akzeptanzkriterien:

- ☐ Alle dem eingeloggten Benutzer zugewiesenen Aufgaben werden angezeigt
- ☐ Loading-State während des Ladens der Aufgaben
- ☐ Empty-State wird angezeigt, wenn keine Aufgaben vorhanden sind
- ☐ Aufgaben werden mit Projekt und Priorität angezeigt
- ☐ Liste aktualisiert sich automatisch nach Statusänderungen

FR-03 Aufgabe als erledigt/unerledigt markieren Beschreibung:

- Checkbox oder Toggle-Button zum Markieren einer Aufgabe als erledigt/unerledigt
- Sofortiges visuelles Feedback (Optimistic UI Update)
- Bei Fehler: Rollback des Status und Fehlermeldung

Akzeptanzkriterien:

- ☐ Klick auf Checkbox/Toggle ändert den Status der Aufgabe
- ☐ API-Request wird im Hintergrund ausgeführt
- ☐ Optimistic Update: UI zeigt sofort den neuen Status
- ☐ Bei API-Fehler: Rollback auf vorherigen Status + Toast-Meldung
- ☐ Erledigte Aufgaben werden visuell als erledigt dargestellt

FR-04 Sortieren/Filtern (Projekt, Priorität) Beschreibung:

- Dropdown oder Filterchips für Filterung nach Projekt und Priorität
- Sortieroptionen: Nach Priorität (absteigend/aufsteigend), Nach Erstellungsdatum
- Filter können kombiniert werden
- Aktive Filter werden visuell hervorgehoben

Akzeptanzkriterien:

- ☐ Filter nach Projekt funktioniert (Auswahl eines oder mehrerer Projekte)
- ☐ Filter nach Priorität funktioniert
- ☐ Sortierung ändert die Reihenfolge der Aufgaben
- ☐ Filter-Reset-Button setzt alle Filter zurück
- ☐ Filterung erfolgt clientseitig für schnelle UX (bei kleinen Datenmengen)

3.3 Anforderungen Abteilungsleiter

FR-05 Aufgabenübersicht (Team / alle Aufgaben) Beschreibung:

- Erweiterte Ansicht mit allen Aufgaben der Abteilung/des Teams
- Zusätzliche Spalte: Zugewiesener Mitarbeiter
- Filter nach Mitarbeiter zusätzlich zu Projekt und Priorität
- Gruppierungsmöglichkeit nach Mitarbeiter oder Projekt

Akzeptanzkriterien:

- ☐ Alle Aufgaben des Teams werden angezeigt (nicht nur eigene)
- ☐ Mitarbeiter-Name ist bei jeder Aufgabe sichtbar
- ☐ Filter nach Mitarbeiter ist verfügbar
- ☐ Abteilungsleiter sieht nur Aufgaben seiner zugeordneten Mitarbeiter
- ☐ Gleiche Sortier/Filter-Funktionen wie Mitarbeiter-Ansicht

FR-06 Neue Aufgabe erstellen und einem Mitarbeiter zuweisen Beschreibung:

- Formular oder Modal zur Erstellung einer neuen Aufgabe
- Dropdown zur Auswahl des Mitarbeiters (nur Mitarbeiter der eigenen Abteilung)

- Dropdown zur Auswahl von Projekt und Priorität
- Titel und optionale Beschreibung als Textfelder
- Optionales Fälligkeitsdatum

Akzeptanzkriterien:

- ☐ Formular zeigt alle erforderlichen Felder an
- ☐ Mitarbeiter-Dropdown zeigt nur zugewiesene Teammitglieder
- ☐ Projekt-Dropdown zeigt alle verfügbaren Projekte
- ☐ Priorität-Dropdown zeigt alle verfügbaren Prioritäten
- ☐ Nach erfolgreicher Erstellung: Schließen des Modals und Aktualisierung der Liste
- ☐ Erfolgs-Toast wird angezeigt

Pflichtfelder im UI:

- Titel (mind. 3 Zeichen)
- Zugewiesener Mitarbeiter
- Priorität
- Projekt

Validierung im UI:

- Titel: Mindestlänge 3 Zeichen, Maximallänge 100 Zeichen
- Alle Pflichtfelder müssen ausgefüllt sein
- Fälligkeitsdatum: Muss in der Zukunft liegen (falls angegeben)
- Echtzeit-Validierung mit visueller Fehleranzeige

FR-07 Projekte verwalten (anzeigen / anlegen / bearbeiten / löschen) Beschreibung:

- Separate Seite oder Tab für Projektverwaltung
- Tabellarische Auflistung aller Projekte mit Name und Beschreibung
- "Neues Projekt"-Button öffnet Formular/Modal
- Inline-Editing oder Edit-Modal für bestehende Projekte
- Löschen mit Bestätigungsdialog

Akzeptanzkriterien:

- ☐ Alle Projekte werden in einer Liste/Tabelle angezeigt
- ☐ Projekt kann mit Name und optionaler Beschreibung angelegt werden
- ☐ Projekt kann bearbeitet werden
- ☐ Projekt kann gelöscht werden (mit Warnung, falls Aufgaben zugeordnet)
- ☐ Löschbestätigung ist erforderlich
- ☐ Validierung: Projektname ist Pflichtfeld und muss eindeutig sein

FR-08 Prioritäten verwalten (anzeigen / anlegen / bearbeiten / löschen) Beschreibung:

- Verwaltungsseite für Prioritäten
- Anzeige von Name und optionaler Farbcodierung
- CRUD-Operationen für Prioritäten
- Drag-and-Drop für Reihenfolge (optional)

Akzeptanzkriterien:

- [ ] Alle Prioritäten werden aufgelistet
- [ ] Neue Priorität kann erstellt werden
- [ ] Priorität kann bearbeitet werden
- [ ] Priorität kann gelöscht werden (mit Warnung bei zugeordneten Aufgaben)
- [ ] Standard-Prioritäten (Hoch, Mittel, Niedrig) sind vordefiniert

---

3.4 Anforderungen Administrator

FR-09 Benutzerliste anzeigen Beschreibung:

- Tabellarische Übersicht aller Benutzer im System
- Spalten: Benutzername, Rolle, Status (aktiv/inaktiv)
- Suchfunktion nach Benutzername
- Filter nach Rolle

Akzeptanzkriterien:

- [ ] Alle Benutzer werden in einer Tabelle angezeigt
- [ ] Benutzername und Rolle sind sichtbar
- [ ] Suchfeld filtert die Liste
- [ ] Rollenfilter funktioniert
- [ ] Paginierung bei vielen Benutzern

FR-10 Benutzer anlegen Beschreibung:

- Modal oder Formular-Seite zum Anlegen neuer Benutzer
- Felder: Benutzername, Passwort, Passwort-Bestätigung, Rolle
- Rollenauswahl als Dropdown (Mitarbeiter, Abteilungsleiter, Administrator)
- Optionale Zuweisung zu einem Abteilungsleiter (für Mitarbeiter)

Akzeptanzkriterien:

- [ ] Formular zeigt alle erforderlichen Felder
- [ ] Rollenauswahl ist als Dropdown verfügbar
- [ ] Nach erfolgreicher Erstellung: Weiterleitung zur Benutzerliste
- [ ] Erfolgs-Toast wird angezeigt
- [ ] Bei Fehler (z.B. Benutzername existiert): Fehlermeldung wird angezeigt

Pflichtfelder im UI:

- Benutzername
- Passwort
- Passwort-Bestätigung
- Rolle

Validierung im UI:

- Benutzername: Mindestlänge 3 Zeichen, nur alphanumerische Zeichen

- Passwort: Mindestlänge 8 Zeichen, mindestens eine Zahl und ein Sonderzeichen
- Passwort-Bestätigung: Muss mit Passwort übereinstimmen
- Echtzeit-Validierung mit Inline-Fehlermeldungen

FR-11 Benutzer bearbeiten Beschreibung:

- Edit-Modal oder separate Seite zum Bearbeiten eines Benutzers
- Änderbare Felder: Rolle, Passwort (optional)
- Benutzername ist nicht änderbar (nur Anzeige)
- Passwortänderung nur wenn explizit gewünscht

Akzeptanzkriterien:

- ☐ Benutzer kann über Edit-Button bearbeitet werden
- ☐ Rolle kann geändert werden
- ☐ Passwort kann optional neu gesetzt werden
- ☐ Speichern aktualisiert den Benutzer und schließt das Modal
- ☐ Validierung wie bei Neuanlage

FR-12 Benutzer löschen Beschreibung:

- Löschen über Delete-Button in der Benutzerliste
- Bestätigungsdialog vor dem Löschen
- Eigenes Benutzerkonto kann nicht gelöscht werden

Akzeptanzkriterien:

- ☐ Delete-Button ist bei jedem Benutzer sichtbar (außer eigenem)
- ☐ Bestätigungsdialog erscheint ("Sind Sie sicher?")
- ☐ Nach Bestätigung wird der Benutzer gelöscht
- ☐ Liste wird automatisch aktualisiert
- ☐ Fehler bei Löschen von Benutzern mit zugeordneten Aufgaben wird angezeigt

---

4. Nicht-funktionale Anforderungen

Usability (intuitiv, verständliche UI):

- Konsistente UI-Patterns in der gesamten Anwendung
- Klare visuelle Hierarchie durch Tailwind-Typography
- Eindeutige Call-to-Action Buttons (Primary/Secondary Styling)
- Breadcrumb-Navigation für tiefere Seitenstrukturen
- Tooltips für Icons und nicht selbsterklärende Elemente
- Tastaturnavigation für alle interaktiven Elemente (Accessibility-Grundlagen)
- Aussagekräftige Fehlermeldungen mit Handlungsempfehlungen
- Konsistente Icon-Sprache (z.B. Mülleimer für Löschen, Stift für Bearbeiten)

Performance (Reaktionszeiten, Ladezeiten):

- Initial Load (First Contentful Paint): < 1,5 Sekunden
- Time to Interactive: < 3 Sekunden

- API-Response-Feedback: Sofortiger Loading-State (<100ms)
- Code-Splitting für Route-basiertes Lazy Loading
- Optimierte Bundle-Größe durch Tree-Shaking
- Caching von API-Responses mit React Query (Stale-While-Revalidate)
- Optimistic Updates für bessere gefühlte Performance
- Keine unnötigen Re-Renders durch React.memo und useMemo

Responsive Design (Smartphone/Tablet/Desktop):

- Mobile-First Entwicklungsansatz mit Tailwind Breakpoints
- Breakpoints: sm (640px), md (768px), lg (1024px), xl (1280px)
- Touch-freundliche Tap-Targets (min. 44x44px)
- Anpassung der Navigation: Hamburger-Menü auf Mobile, Sidebar auf Desktop
- Tabellen werden auf Mobile zu Card-Layout transformiert
- Formulare sind auf allen Gerätegrößen nutzbar
- Keine horizontale Scrollbar auf mobilen Geräten

Datenschutz/Sicherheit (Token-Handling, Fehlerverhalten, keine sensiblen Daten im Frontend):

- JWT-Token wird nur im localStorage gespeichert, nicht im Code
- Token wird automatisch bei jedem API-Request im Authorization-Header gesendet
- Bei 401-Response: Automatischer Logout und Token-Löschung
- Token-Refresh-Logik bei fast abgelaufenem Token (falls Backend unterstützt)
- Keine sensiblen Daten in URL-Parametern
- Passwörter werden niemals im Klartext angezeigt oder geloggt
- XSS-Schutz durch korrektes Escaping (React-Standardverhalten)
- CSRF-Schutz durch SameSite-Cookies oder Token-basierte Auth
- Keine sensiblen Fehlermeldungen an den Benutzer (Stack Traces etc.)

---

5. UI/UX-Vorgaben

Screens/Ansichten (Liste + kurze Beschreibung):

Screen	Route	Beschreibung	Zugänglich für
Login	`/login`	Anmeldeformular mit Benutzername/Passwort	Alle (unauthentifiziert)
Dashboard	`/` oder `/dashboard`	Übersicht mit wichtigsten KPIs und Schnellzugriff	Alle (authentifiziert)
Meine Aufgaben	`/tasks`	Liste der eigenen Aufgaben mit Filter/Sortierung	Alle (authentifiziert)
Aufgaben-Detail	`/tasks/:id`	Detailansicht einer Aufgabe	Alle (authentifiziert)
Team-Aufgaben	`/team/tasks`	Alle Aufgaben des Teams	Abteilungsleiter

Screen	Route	Beschreibung	Zugänglich für
Neue Aufgabe	`/tasks/new`	Formular zum Erstellen einer Aufgabe	Abteilungsleiter
Projekte	`/projects`	Projektverwaltung (CRUD)	Abteilungsleiter
Prioritäten	`/priorities`	Prioritätsverwaltung (CRUD)	Abteilungsleiter
Benutzerverwaltung	`/admin/users`	Liste aller Benutzer	Administrator
Benutzer anlegen	`/admin/users/new`	Formular für neuen Benutzer	Administrator
Benutzer bearbeiten	`/admin/users/:id/edit`	Bearbeitungsformular	Administrator

Navigation (wie wechselt man zwischen den Screens?):

- **Desktop**: Seitenleiste (Sidebar) links mit Icon + Text-Navigation
- **Tablet**: Ausklappbare Sidebar mit Icons
- **Mobile**: Bottom-Navigation oder Hamburger-Menü oben
- Navigationseinträge werden rollenbasiert angezeigt/ausgeblendet
- Aktiver Navigationsbereich ist visuell hervorgehoben
- Breadcrumbs auf Detailseiten zur Orientierung
- Logo/App-Name führt immer zum Dashboard

Zustände (Loading, Empty, Error – wie werden sie angezeigt?):

Loading-State:

- Skeleton-Loader für Tabellen und Listen (Tailwind animate-pulse)
- Spinner für Buttons während API-Calls
- Overlay-Spinner für modale Aktionen

Empty-State:

- Zentrierte Illustration oder Icon
- Aussagekräftige Nachricht ("Keine Aufgaben vorhanden")
- Call-to-Action falls passend ("Erste Aufgabe erstellen")

Error-State:

- Inline-Fehler bei Formularfeldern (rot umrandet, Text darunter)
- Toast-Notifications für API-Fehler (unten rechts, auto-dismiss)
- Vollbild-Fehlerseite für 500er-Fehler mit Retry-Button
- 404-Seite bei nicht gefundenen Ressourcen

5.1 Farben

- Primärfarbe: #5F6463
- Sekundärfarbe: #9D9B9D
- Akzentfarbe: #FF4845

Logos/Icons:

Verpflichtend: Das Icon der Tiese GmbH (siehe Datei logo.png) muss in der App auf weißem Hintergrund verwendet werden.

---

6. Abnahmekriterien

Beschreibe, wann das Frontend als „fertig" gilt (prüfbare Kriterien in Textform):

Funktional:

- [ ] Alle funktionalen Anforderungen (FR-01 bis FR-12) sind implementiert und testbar
- [ ] Alle Akzeptanzkriterien der einzelnen FRs sind erfüllt
- [ ] Login/Logout-Flow funktioniert fehlerfrei
- [ ] Rollenbasierte Zugriffssteuerung funktioniert (Mitarbeiter, Abteilungsleiter, Admin)
- [ ] Alle CRUD-Operationen für Aufgaben, Projekte, Prioritäten und Benutzer sind möglich
- [ ] Filter- und Sortierfunktionen arbeiten korrekt

Technisch:

- [ ] Anwendung baut fehlerfrei mit `npm run build`
- [ ] Keine TypeScript-Fehler oder ESLint-Warnungen (außer dokumentierte Ausnahmen)
- [ ] Alle API-Endpoints aus der Swagger-Dokumentation sind integriert
- [ ] Token-Handling ist implementiert (Speicherung, Auto-Logout bei 401)
- [ ] Environment-Variablen für API-URL konfiguriert

UI/UX:

- [ ] Responsive Design funktioniert auf Mobile (375px), Tablet (768px) und Desktop (1280px)
- [ ] Alle Loading-, Empty- und Error-States sind implementiert
- [ ] Konsistentes Styling mit Tailwind CSS
- [ ] Farben und Icons entsprechen den Vorgaben
- [ ] Navigation ist auf allen Gerätegrößen nutzbar

Performance:

- [ ] Lighthouse Performance Score > 80
- [ ] Bundle-Größe (gzipped) < 500KB
- [ ] First Contentful Paint < 2s auf 3G-Verbindung

Dokumentation:

- [ ] README.md mit Installationsanleitung ist vorhanden
- [ ] Environment-Setup ist dokumentiert
- [ ] Verfügbare npm-Scripts sind beschrieben

Anhang A – Swagger/OpenAPI (verbindlicher API-Vertrag)

Base-URL(s) aus Swagger: https://lf9server.onrender.com/


```
{
  "openapi": "3.0.0",
  "info": {
    "title": "Task Management API",
    "version": "1.0.0",
    "description": "A simple Task Management API application made with Express and
documented with Swagger"
  },
  "paths": {
    "/auth/login": {
      "post": {
        "summary": "Authenticate user and return a JWT token",
        "tags": [
          "Authentication"
        ],
        "requestBody": {
          "required": true,
          "content": {
            "application/json": {
              "schema": {
                "type": "object",
                "required": [
                  "username",
                  "password"
                ],
                "properties": {
                  "username": {
                    "type": "string",
                    "description": "User's username"
                  },
                  "password": {
                    "type": "string",
                    "description": "User's password"
                  }
                }
              }
            }
          }
        },
        "responses": {
          "200": {
            "description": "Authentication successful, returns JWT token and user
object",
            "content": {
              "application/json": {
                "schema": {
                  "type": "object",
                  "properties": {
                    "message": {
                      "type": "string",
                      "example": "Login successful"
```

```
                    },
                    "token": {
                      "type": "string",
                      "description": "JWT authentication token"
                    },
                    "user": {
                      "type": "object",
                      "properties": {
                        "id": {
                          "type": "integer"
                        },
                        "username": {
                          "type": "string"
                        },
                        "role": {
                          "type": "string"
                        }
                      }
                    }
                  }
                }
              }
            }
          },
          "401": {
            "description": "Invalid credentials",
            "content": {
              "application/json": {
                "schema": {
                  "type": "object",
                  "properties": {
                    "message": {
                      "type": "string",
                      "example": "Invalid username or password"
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "/auth/register": {
      "post": {
        "summary": "Create a new user (Admin only)",
        "tags": [
          "Authentication"
        ],
        "security": [
          {
            "bearerAuth": []
          }
        ],
```

```
        "requestBody": {
          "required": true,
          "content": {
            "application/json": {
              "schema": {
                "type": "object",
                "required": [
                  "username",
                  "password",
                  "role"
                ],
                "properties": {
                  "username": {
                    "type": "string",
                    "description": "New user's username"
                  },
                  "password": {
                    "type": "string",
                    "description": "New user's password"
                  },
                  "role": {
                    "type": "string",
                    "description": "Role of the new user (e.g., Administrator,
Abteilungsleiter, Mitarbeiter)",
                    "enum": [
                      "Administrator",
                      "Abteilungsleiter",
                      "Mitarbeiter"
                    ]
                  }
                }
              }
            }
          }
        },
        "responses": {
          "201": {
            "description": "User created successfully",
            "content": {
              "application/json": {
                "schema": {
                  "type": "object",
                  "properties": {
                    "message": {
                      "type": "string",
                      "example": "User created successfully"
                    },
                    "userId": {
                      "type": "integer",
                      "example": 4
                    }
                  }
                }
              }
```

```
            }
          },
          "400": {
            "description": "Invalid input"
          },
          "401": {
            "description": "Unauthorized"
          },
          "403": {
            "description": "Forbidden, only Administrators can create users"
          },
          "409": {
            "description": "Username already exists"
          }
        }
      }
    },
    "/priorities": {
      "get": {
        "summary": "Retrieve a list of all priorities",
        "tags": [
          "Priorities"
        ],
        "security": [
          {
            "bearerAuth": []
          }
        ],
        "responses": {
          "200": {
            "description": "A list of priorities.",
            "content": {
              "application/json": {
                "schema": {
                  "type": "array",
                  "items": {
                    "type": "object",
                    "properties": {
                      "id": {
                        "type": "integer",
                        "description": "The priority ID.",
                        "example": 1
                      },
                      "name": {
                        "type": "string",
                        "description": "The priority name.",
                        "example": "High"
                      }
                    }
                  }
                }
              }
            }
          },
```

```
            "401": {
              "description": "Unauthorized"
            }
          }
        },
        "post": {
          "summary": "Create a new priority",
          "tags": [
            "Priorities"
          ],
          "security": [
            {
              "bearerAuth": []
            }
          ],
          "requestBody": {
            "required": true,
            "content": {
              "application/json": {
                "schema": {
                  "type": "object",
                  "required": [
                    "name"
                  ],
                  "properties": {
                    "name": {
                      "type": "string",
                      "description": "The name of the priority.",
                      "example": "Urgent"
                    }
                  }
                }
              }
            }
          },
          "responses": {
            "201": {
              "description": "Priority created successfully.",
              "content": {
                "application/json": {
                  "schema": {
                    "type": "object",
                    "properties": {
                      "id": {
                        "type": "integer",
                        "description": "The ID of the newly created priority.",
                        "example": 4
                      },
                      "name": {
                        "type": "string",
                        "description": "The name of the newly created priority.",
                        "example": "Urgent"
                      }
                    }
```

```
                }
              }
            }
          },
          "400": {
            "description": "Invalid input"
          },
          "401": {
            "description": "Unauthorized"
          },
          "403": {
            "description": "Forbidden, only Abteilungsleiter can create
priorities"
          }
        }
      }
    },
    "/priorities/{id}": {
      "get": {
        "summary": "Retrieve a single priority by ID",
        "tags": [
          "Priorities"
        ],
        "security": [
          {
            "bearerAuth": []
          }
        ],
        "parameters": [
          {
            "in": "path",
            "name": "id",
            "required": true,
            "description": "Numeric ID of the priority to retrieve.",
            "schema": {
              "type": "integer"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "A single priority.",
            "content": {
              "application/json": {
                "schema": {
                  "type": "object",
                  "properties": {
                    "id": {
                      "type": "integer",
                      "description": "The priority ID.",
                      "example": 1
                    },
                    "name": {
                      "type": "string",
```

```
                    "description": "The priority name.",
                    "example": "High"
                  }
                }
              }
            }
          }
        },
        "401": {
          "description": "Unauthorized"
        },
        "404": {
          "description": "Priority not found."
        }
      }
    },
    "put": {
      "summary": "Update a priority by ID",
      "tags": [
        "Priorities"
      ],
      "security": [
        {
          "bearerAuth": []
        }
      ],
      "parameters": [
        {
          "in": "path",
          "name": "id",
          "required": true,
          "description": "Numeric ID of the priority to update.",
          "schema": {
            "type": "integer"
          }
        }
      ],
      "requestBody": {
        "required": true,
        "content": {
          "application/json": {
            "schema": {
              "type": "object",
              "required": [
                "name"
              ],
              "properties": {
                "name": {
                  "type": "string",
                  "description": "The new name of the priority.",
                  "example": "Very High"
                }
              }
            }
```

```
            }
          }
        },
        "responses": {
          "200": {
            "description": "Priority updated successfully.",
            "content": {
              "application/json": {
                "schema": {
                  "type": "object",
                  "properties": {
                    "message": {
                      "type": "string",
                      "example": "Priority updated successfully"
                    }
                  }
                }
              }
            }
          },
          "400": {
            "description": "Invalid input"
          },
          "401": {
            "description": "Unauthorized"
          },
          "403": {
            "description": "Forbidden, only Abteilungsleiter can update
priorities"
          },
          "404": {
            "description": "Priority not found."
          }
        }
      },
      "delete": {
        "summary": "Delete a priority by ID",
        "tags": [
          "Priorities"
        ],
        "security": [
          {
            "bearerAuth": []
          }
        ],
        "parameters": [
          {
            "in": "path",
            "name": "id",
            "required": true,
            "description": "Numeric ID of the priority to delete.",
            "schema": {
              "type": "integer"
            }
```

```
          }
        ],
        "responses": {
          "200": {
            "description": "Priority deleted successfully.",
            "content": {
              "application/json": {
                "schema": {
                  "type": "object",
                  "properties": {
                    "message": {
                      "type": "string",
                      "example": "Priority deleted successfully"
                    }
                  }
                }
              }
            }
          },
          "401": {
            "description": "Unauthorized"
          },
          "403": {
            "description": "Forbidden, only Abteilungsleiter can delete
priorities"
          },
          "404": {
            "description": "Priority not found."
          }
        }
      }
    },
    "/projects": {
      "get": {
        "summary": "Retrieve a list of all projects",
        "tags": [
          "Projects"
        ],
        "security": [
          {
            "bearerAuth": []
          }
        ],
        "responses": {
          "200": {
            "description": "A list of projects.",
            "content": {
              "application/json": {
                "schema": {
                  "type": "array",
                  "items": {
                    "type": "object",
                    "properties": {
                      "id": {
```

```
                    "type": "integer",
                    "description": "The project ID.",
                    "example": 1
                  },
                  "name": {
                    "type": "string",
                    "description": "The project name.",
                    "example": "Website Relaunch"
                  }
                }
              }
            }
          }
        }
      },
      "401": {
        "description": "Unauthorized"
      }
    }
  },
  "post": {
    "summary": "Create a new project",
    "tags": [
      "Projects"
    ],
    "security": [
      {
        "bearerAuth": []
      }
    ],
    "requestBody": {
      "required": true,
      "content": {
        "application/json": {
          "schema": {
            "type": "object",
            "required": [
              "name"
            ],
            "properties": {
              "name": {
                "type": "string",
                "description": "The name of the project.",
                "example": "New Feature Development"
              }
            }
          }
        }
      }
    },
    "responses": {
      "201": {
        "description": "Project created successfully.",
        "content": {
```

```
              "application/json": {
                "schema": {
                  "type": "object",
                  "properties": {
                    "id": {
                      "type": "integer",
                      "description": "The ID of the newly created project.",
                      "example": 4
                    },
                    "name": {
                      "type": "string",
                      "description": "The name of the newly created project.",
                      "example": "New Feature Development"
                    }
                  }
                }
              }
            }
          },
          "400": {
            "description": "Invalid input"
          },
          "401": {
            "description": "Unauthorized"
          },
          "403": {
            "description": "Forbidden, only Abteilungsleiter can create projects"
          }
        }
      }
    },
    "/projects/{id}": {
      "get": {
        "summary": "Retrieve a single project by ID",
        "tags": [
          "Projects"
        ],
        "security": [
          {
            "bearerAuth": []
          }
        ],
        "parameters": [
          {
            "in": "path",
            "name": "id",
            "required": true,
            "description": "Numeric ID of the project to retrieve.",
            "schema": {
              "type": "integer"
            }
          }
        ],
        "responses": {
```

```
        "200": {
          "description": "A single project.",
          "content": {
            "application/json": {
              "schema": {
                "type": "object",
                "properties": {
                  "id": {
                    "type": "integer",
                    "description": "The project ID.",
                    "example": 1
                  },
                  "name": {
                    "type": "string",
                    "description": "The project name.",
                    "example": "Website Relaunch"
                  }
                }
              }
            }
          }
        },
        "401": {
          "description": "Unauthorized"
        },
        "404": {
          "description": "Project not found."
        }
      }
    },
    "put": {
      "summary": "Update a project by ID",
      "tags": [
        "Projects"
      ],
      "security": [
        {
          "bearerAuth": []
        }
      ],
      "parameters": [
        {
          "in": "path",
          "name": "id",
          "required": true,
          "description": "Numeric ID of the project to update.",
          "schema": {
            "type": "integer"
          }
        }
      ],
      "requestBody": {
        "required": true,
        "content": {
```

```
            "application/json": {
              "schema": {
                "type": "object",
                "required": [
                  "name"
                ],
                "properties": {
                  "name": {
                    "type": "string",
                    "description": "The new name of the project.",
                    "example": "Website Redesign"
                  }
                }
              }
            }
          }
        },
        "responses": {
          "200": {
            "description": "Project updated successfully.",
            "content": {
              "application/json": {
                "schema": {
                  "type": "object",
                  "properties": {
                    "message": {
                      "type": "string",
                      "example": "Project updated successfully"
                    }
                  }
                }
              }
            }
          },
          "400": {
            "description": "Invalid input"
          },
          "401": {
            "description": "Unauthorized"
          },
          "403": {
            "description": "Forbidden, only Abteilungsleiter can update projects"
          },
          "404": {
            "description": "Project not found."
          }
        }
      },
      "delete": {
        "summary": "Delete a project by ID",
        "tags": [
          "Projects"
        ],
        "security": [
```

```
            {
              "bearerAuth": []
            }
          ],
          "parameters": [
            {
              "in": "path",
              "name": "id",
              "required": true,
              "description": "Numeric ID of the project to delete.",
              "schema": {
                "type": "integer"
              }
            }
          ],
          "responses": {
            "200": {
              "description": "Project deleted successfully.",
              "content": {
                "application/json": {
                  "schema": {
                    "type": "object",
                    "properties": {
                      "message": {
                        "type": "string",
                        "example": "Project deleted successfully"
                      }
                    }
                  }
                }
              }
            },
            "401": {
              "description": "Unauthorized"
            },
            "403": {
              "description": "Forbidden, only Abteilungsleiter can delete projects"
            },
            "404": {
              "description": "Project not found."
            }
          }
        }
      },
      "/tasks": {
        "get": {
          "summary": "Retrieve a list of all tasks",
          "tags": [
            "Tasks"
          ],
          "security": [
            {
              "bearerAuth": []
            }
```

```
        ],
        "responses": {
          "200": {
            "description": "A list of tasks.",
            "content": {
              "application/json": {
                "schema": {
                  "type": "array",
                  "items": {
                    "type": "object",
                    "properties": {
                      "id": {
                        "type": "integer",
                        "description": "The task ID.",
                        "example": 1
                      },
                      "title": {
                        "type": "string",
                        "description": "The task title.",
                        "example": "Implement user authentication"
                      },
                      "description": {
                        "type": "string",
                        "description": "The task description.",
                        "example": "Set up JWT-based authentication for user
login."
                      },
                      "dueDate": {
                        "type": "string",
                        "format": "date",
                        "description": "The due date of the task.",
                        "example": "2024-12-31"
                      },
                      "done": {
                        "type": "boolean",
                        "description": "Whether the task is completed.",
                        "example": false
                      },
                      "priority_id": {
                        "type": "integer",
                        "description": "The ID of the associated priority.",
                        "example": 1
                      },
                      "project_id": {
                        "type": "integer",
                        "description": "The ID of the associated project.",
                        "example": 1
                      },
                      "user_id": {
                        "type": "integer",
                        "description": "The ID of the user assigned to the task.",
                        "example": 1
                      }
                    }
```

```
                    }
                  }
                }
              }
            },
            "401": {
              "description": "Unauthorized"
            }
          }
        },
        "post": {
          "summary": "Create a new task",
          "tags": [
            "Tasks"
          ],
          "security": [
            {
              "bearerAuth": []
            }
          ],
          "requestBody": {
            "required": true,
            "content": {
              "application/json": {
                "schema": {
                  "type": "object",
                  "required": [
                    "title"
                  ],
                  "properties": {
                    "title": {
                      "type": "string",
                      "description": "The title of the task.",
                      "example": "Design database schema"
                    },
                    "description": {
                      "type": "string",
                      "description": "The description of the task.",
                      "example": "Create ER diagrams and define table structures."
                    },
                    "dueDate": {
                      "type": "string",
                      "format": "date",
                      "description": "The due date of the task.",
                      "example": "2024-10-15"
                    },
                    "done": {
                      "type": "boolean",
                      "description": "Whether the task is completed.",
                      "example": false
                    },
                    "priority_id": {
                      "type": "integer",
                      "description": "The ID of the associated priority.",
```

```
                    "example": 2
                  },
                  "project_id": {
                    "type": "integer",
                    "description": "The ID of the associated project.",
                    "example": 1
                  },
                  "user_id": {
                    "type": "integer",
                    "description": "The ID of the user assigned to the task.
(Optional for Mitarbeiter, required for Abteilungsleiter/Administrator when
assigning to others)",
                    "example": 1
                  }
                }
              }
            }
          }
        },
        "responses": {
          "201": {
            "description": "Task created successfully.",
            "content": {
              "application/json": {
                "schema": {
                  "type": "object",
                  "properties": {
                    "id": {
                      "type": "integer",
                      "description": "The ID of the newly created task.",
                      "example": 5
                    },
                    "title": {
                      "type": "string",
                      "description": "The title of the newly created task.",
                      "example": "Design database schema"
                    }
                  }
                }
              }
            }
          },
          "400": {
            "description": "Invalid input"
          },
          "401": {
            "description": "Unauthorized"
          },
          "403": {
            "description": "Forbidden, only Abteilungsleiter can create tasks"
          }
        }
      }
    },
```

```
    "/tasks/{id}": {
      "get": {
        "summary": "Retrieve a single task by ID",
        "tags": [
          "Tasks"
        ],
        "security": [
          {
            "bearerAuth": []
          }
        ],
        "parameters": [
          {
            "in": "path",
            "name": "id",
            "required": true,
            "description": "Numeric ID of the task to retrieve.",
            "schema": {
              "type": "integer"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "A single task.",
            "content": {
              "application/json": {
                "schema": {
                  "type": "object",
                  "properties": {
                    "id": {
                      "type": "integer",
                      "description": "The task ID.",
                      "example": 1
                    },
                    "title": {
                      "type": "string",
                      "description": "The task title.",
                      "example": "Implement user authentication"
                    }
                  }
                }
              }
            }
          },
          "401": {
            "description": "Unauthorized"
          },
          "404": {
            "description": "Task not found."
          }
        }
      },
      "put": {
```

```
        "summary": "Update a task by ID",
        "tags": [
          "Tasks"
        ],
        "security": [
          {
            "bearerAuth": []
          }
        ],
        "parameters": [
          {
            "in": "path",
            "name": "id",
            "required": true,
            "description": "Numeric ID of the task to update.",
            "schema": {
              "type": "integer"
            }
          }
        ],
        "requestBody": {
          "required": true,
          "content": {
            "application/json": {
              "schema": {
                "type": "object",
                "properties": {
                  "title": {
                    "type": "string",
                    "description": "The new title of the task.",
                    "example": "Implement user authentication (completed)"
                  },
                  "description": {
                    "type": "string",
                    "description": "The new description of the task.",
                    "example": "Set up JWT-based authentication for user login and
integrate with frontend."
                  },
                  "dueDate": {
                    "type": "string",
                    "format": "date",
                    "description": "The new due date of the task.",
                    "example": "2024-09-30"
                  },
                  "done": {
                    "type": "boolean",
                    "description": "Whether the task is completed.",
                    "example": true
                  },
                  "priority_id": {
                    "type": "integer",
                    "description": "The ID of the associated priority.",
                    "example": 3
                  },
```

```
                  "project_id": {
                    "type": "integer",
                    "description": "The ID of the associated project.",
                    "example": 1
                  },
                  "user_id": {
                    "type": "integer",
                    "description": "The ID of the user assigned to the task.
(Optional for Mitarbeiter, required for Abteilungsleiter/Administrator when
assigning to others)",
                    "example": 1
                  }
                }
              }
            }
          }
        },
        "responses": {
          "200": {
            "description": "Task updated successfully.",
            "content": {
              "application/json": {
                "schema": {
                  "type": "object",
                  "properties": {
                    "message": {
                      "type": "string",
                      "example": "Task updated successfully"
                    }
                  }
                }
              }
            }
          },
          "400": {
            "description": "Invalid input"
          },
          "401": {
            "description": "Unauthorized"
          },
          "403": {
            "description": "Forbidden, only Abteilungsleiter can update tasks"
          },
          "404": {
            "description": "Task not found."
          }
        }
      },
      "delete": {
        "summary": "Delete a task by ID",
        "tags": [
          "Tasks"
        ],
        "security": [
```

```
            {
              "bearerAuth": []
            }
          ],
          "parameters": [
            {
              "in": "path",
              "name": "id",
              "required": true,
              "description": "Numeric ID of the task to delete.",
              "schema": {
                "type": "integer"
              }
            }
          ],
          "responses": {
            "200": {
              "description": "Task deleted successfully.",
              "content": {
                "application/json": {
                  "schema": {
                    "type": "object",
                    "properties": {
                      "message": {
                        "type": "string",
                        "example": "Task deleted successfully"
                      }
                    }
                  }
                }
              }
            },
            "401": {
              "description": "Unauthorized"
            },
            "403": {
              "description": "Forbidden, only Abteilungsleiter can delete tasks"
            },
            "404": {
              "description": "Task not found."
            }
          }
        }
      },
      "/tasks/{id}/done": {
        "patch": {
          "summary": "Mark a task as done or undone",
          "tags": [
            "Tasks"
          ],
          "security": [
            {
              "bearerAuth": []
            }
```

```
        ],
        "parameters": [
          {
            "in": "path",
            "name": "id",
            "required": true,
            "description": "Numeric ID of the task to update.",
            "schema": {
              "type": "integer"
            }
          }
        ],
        "requestBody": {
          "required": true,
          "content": {
            "application/json": {
              "schema": {
                "type": "object",
                "required": [
                  "done"
                ],
                "properties": {
                  "done": {
                    "type": "boolean",
                    "description": "The new status of the task (true for done,
false for undone).",
                    "example": true
                  }
                }
              }
            }
          }
        },
        "responses": {
          "200": {
            "description": "Task status updated successfully.",
            "content": {
              "application/json": {
                "schema": {
                  "type": "object",
                  "properties": {
                    "message": {
                      "type": "string",
                      "example": "Task status updated successfully"
                    }
                  }
                }
              }
            }
          },
          "400": {
            "description": "Invalid input"
          },
          "401": {
```

```
            "description": "Unauthorized"
          },
          "404": {
            "description": "Task not found."
          }
        }
      }
    },
    "/users": {
      "get": {
        "summary": "Retrieve a list of all users",
        "tags": [
          "Users"
        ],
        "security": [
          {
            "bearerAuth": []
          }
        ],
        "responses": {
          "200": {
            "description": "A list of users.",
            "content": {
              "application/json": {
                "schema": {
                  "type": "array",
                  "items": {
                    "type": "object",
                    "properties": {
                      "id": {
                        "type": "integer",
                        "description": "The user ID.",
                        "example": 1
                      },
                      "username": {
                        "type": "string",
                        "description": "The user's username.",
                        "example": "admin"
                      },
                      "role": {
                        "type": "string",
                        "description": "The user's role.",
                        "example": "Administrator"
                      }
                    }
                  }
                }
              }
            }
          },
          "401": {
            "description": "Unauthorized"
          },
          "403": {
```

```
            "description": "Forbidden, only Administrators oder Abteilungsleiter
können auf diese Ressource zugreifen"
          }
        }
      }
    },
    "/users/{id}": {
      "get": {
        "summary": "Retrieve a single user by ID",
        "tags": [
          "Users"
        ],
        "security": [
          {
            "bearerAuth": []
          }
        ],
        "parameters": [
          {
            "in": "path",
            "name": "id",
            "required": true,
            "description": "Numeric ID of the user to retrieve.",
            "schema": {
              "type": "integer"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "A single user by ID.",
            "content": {
              "application/json": {
                "schema": {
                  "type": "object",
                  "properties": {
                    "id": {
                      "type": "integer"
                    },
                    "username": {
                      "type": "string"
                    },
                    "role": {
                      "type": "string"
                    }
                  }
                }
              }
            }
          },
          "401": {
            "description": "Unauthorized"
          },
          "403": {
```

```
            "description": "Forbidden, only Administrators can access this
resource"
          },
          "404": {
            "description": "User not found."
          }
        }
      },
      "put": {
        "summary": "Update a user by ID",
        "tags": [
          "Users"
        ],
        "security": [
          {
            "bearerAuth": []
          }
        ],
        "parameters": [
          {
            "in": "path",
            "name": "id",
            "required": true,
            "description": "Numeric ID of the user to update.",
            "schema": {
              "type": "integer"
            }
          }
        ],
        "requestBody": {
          "required": true,
          "content": {
            "application/json": {
              "schema": {
                "type": "object",
                "properties": {
                  "username": {
                    "type": "string"
                  },
                  "password": {
                    "type": "string"
                  },
                  "role": {
                    "type": "string",
                    "enum": [
                      "Administrator",
                      "Abteilungsleiter",
                      "Mitarbeiter"
                    ]
                  }
                }
              }
            }
          }
```

```
        },
        "responses": {
          "200": {
            "description": "User updated successfully.",
            "content": {
              "application/json": {
                "schema": {
                  "type": "object",
                  "properties": {
                    "message": {
                      "type": "string"
                    }
                  }
                }
              }
            }
          },
          "400": {
            "description": "Invalid input"
          },
          "401": {
            "description": "Unauthorized"
          },
          "403": {
            "description": "Forbidden, only Administrators can access this
resource"
          },
          "404": {
            "description": "User not found."
          },
          "409": {
            "description": "Username already exists."
          }
        }
      },
      "delete": {
        "summary": "Delete a user by ID",
        "tags": [
          "Users"
        ],
        "security": [
          {
            "bearerAuth": []
          }
        ],
        "parameters": [
          {
            "in": "path",
            "name": "id",
            "required": true,
            "description": "Numeric ID of the user to delete.",
            "schema": {
              "type": "integer"
            }
```

```
          }
        ],
        "responses": {
          "200": {
            "description": "User deleted successfully.",
            "content": {
              "application/json": {
                "schema": {
                  "type": "object",
                  "properties": {
                    "message": {
                      "type": "string"
                    }
                  }
                }
              }
            }
          },
          "401": {
            "description": "Unauthorized"
          },
          "403": {
            "description": "Forbidden, only Administrators can access this
resource"
          },
          "404": {
            "description": "User not found."
          }
        }
      }
    }
  },
  "components": {
    "securitySchemes": {
      "bearerAuth": {
        "type": "http",
        "scheme": "bearer",
        "bearerFormat": "JWT"
      }
    }
  },
  "tags": [
    {
      "name": "Authentication",
      "description": "User authentication management"
    },
    {
      "name": "Priorities",
      "description": "Priority management"
    },
    {
      "name": "Projects",
      "description": "Project management"
    },
```

```
    {
      "name": "Tasks",
      "description": "Task management"
    },
    {
      "name": "Users",
      "description": "User management (Administratoren; Abteilungsleiter dürfen
die Benutzerliste lesen)"
    }
  ],
  "servers": [
    {
      "url": "http://localhost:3000"
    }
  ]
}
```