Handout – Breakpoint

Teammitglieder

- Nico Reimer
- Dennis Gordeev
- Tim Haug

Sprints (2 Wochen Zyklus)

Sprint 1: Planung und Initialisierung

- Projektidee brainstormen und Anforderungen sammeln: Alle (4h Meeting)
- React Grundlagen lernen: Alle (~6h)
- Erstellen eines groben Projektplans und Milestones setzen: Alle (2h Meeting)
- Erstellen des GitHub Repositories und Dokumentation einrichten: Nico (3h)
- Einrichten von Node.js und Testen der Entwicklungsumgebung: Dennis (4h)

  Gesamt: Nico 15h, Dennis 16h, Tim 12h

Sprint 2: Anforderungsanalyse und Dokumentation

- Erstellen des Software Requirement Specification (SRS): Alle (2h Meeting)
- Use Case Diagramm und erste UML-Diagramme erstellen: Nico + Tim (2h)
- Recherche zu MariaDB und Aufbau der Datenbankstruktur: Dennis (3h)
- Diskussion und Dokumentation der Architekturstile: Alle (3h Meeting)
- Einführung in GitHub für Dokumentation: Alle (2h Meeting)
- Korrigieren und Überarbeiten der ersten Diagramme: Nico (1h)

  Gesamt: Nico 10h, Dennis 10h, Tim 9h

Sprint 3: Klassendesign und Kernfunktionalität

- Erstellen eines ersten Klassendiagramms: Dennis + Nico (2h)
- Implementierung des Grundlayouts mit React: Tim + Dennis (5h)
- Erstellen der API-Schnittstellen und Backend-Grundlagen: Nico (3h)
- Debugging von Layoutproblemen im Frontend: Dennis + Tim (2h)
- Erstellen und Testen erster einfacher Datenbankabfragen: Nico (4h)
- Feedbackrunde und Anpassungen am Klassendiagramm: Alle (3h Meeting)

  Gesamt: Nico 12h, Dennis 12h, Tim 10h

Sprint 4: Erweiterte Features und Architekturentscheidungen

- Implementierung der Sidebar und Kategorie-Logik: Tim + Dennis (4h)
- Erstellen eines Wiki-Editor-Prototyps: Dennis (5h)
- Verbessern der Datenbankintegration: Nico + Tim (3h)
- Fehleranalyse und Bugfixing in der API: Nico + Dennis (4h)
- Architekturentscheidungen finalisieren und dokumentieren: Alle (4h Meeting)
- Integration von Animationen und Hover-Effekten: Tim (5h)

  Gesamt: Nico 11h, Dennis 17h, Tim 16h

Sprint 5: Präsentation und Feinschliff

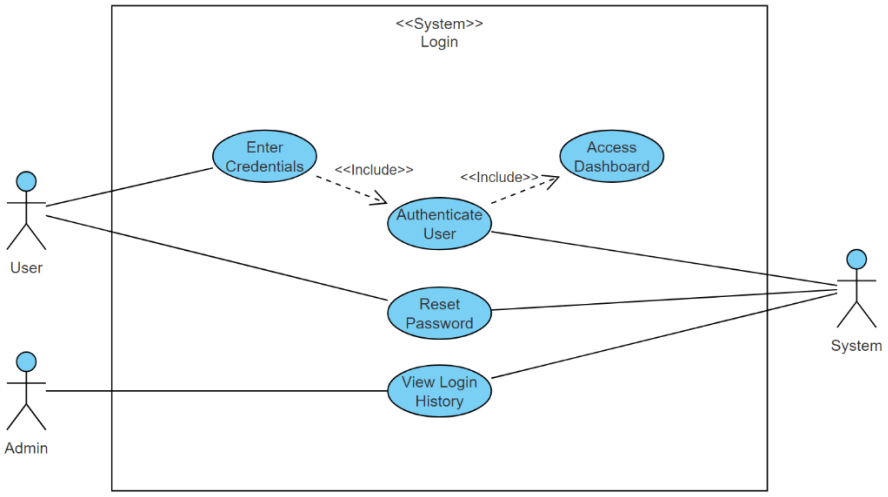
- Erstellen der Präsentation (Folien): Nico (2h), Dennis (1h)
- Handout für die Halbzeitpräsentation vorbereiten: Tim (2h)
- Testen und Debuggen des gesamten Systems: Alle (7h auf 2 Tage)
- Integration von letzten Design-Elementen (Farbanpassungen, Animationen): Tim (3h)
- Erstellen der aktuellen Use-Case Diagramme: Nico (2h)
- Feedbackrunde für Präsentation und Projektstruktur: Alle (2h)

  Gesamt: Nico 13h, Dennis 10h, Tim 14h

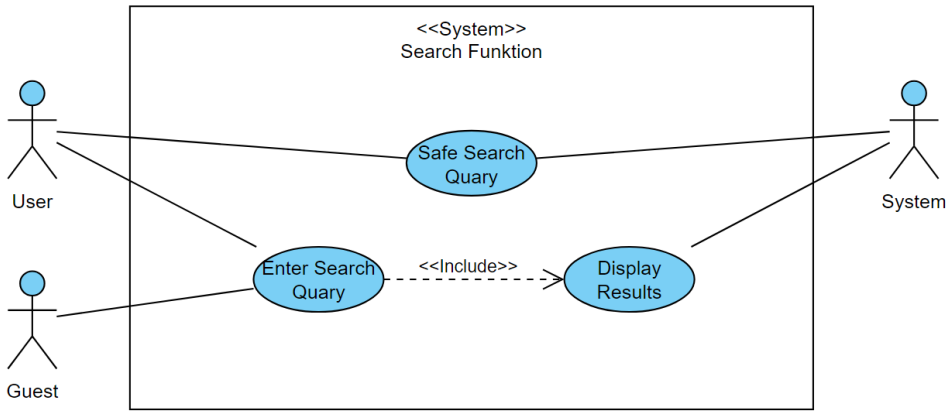
Gesamte Arbeitszeiten:

- Nico: ~61h
- Dennis: ~65h
- Tim: ~61h

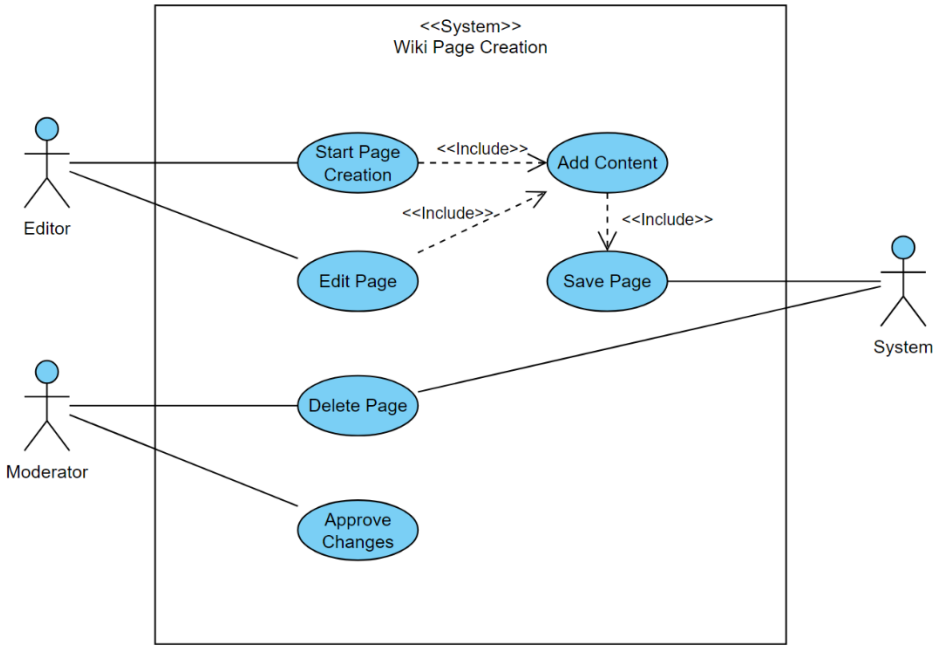
Use-Case-Diagram Login:



Use-Case-Diagram Search Function:



Use-Case-Diagram Wiki Page Creation:



Architekturstile/-entscheidungen:

1. Layered Architecture (Schichtenarchitektur):

- Beschreibung: Klare Trennung zwischen Datenbank, Backend und Frontend.
- Argument: Diese Struktur ermöglicht eine bessere Wartbarkeit und Skalierbarkeit des Systems, da jede Schicht unabhängig voneinander entwickelt und getestet werden kann.

2. MVC (Model-View-Controller) im Backend:

   - Beschreibung: Das Backend wurde nach dem MVC-Muster aufgebaut, um die Trennung von Datenmodell, Logik und Benutzeroberfläche zu gewährleisten.
   - Argument: Diese Struktur reduziert die Komplexität der Codebasis und erleichtert zukünftige Erweiterungen.

3. Komponentenbasiertes Design im Frontend:

   - Beschreibung: Verwendung von React-Komponenten zur Modularisierung des Frontends.
   - Argument: Diese Architektur ermöglicht die Wiederverwendbarkeit von Elementen wie Sidebar, Header und Tiles, was die Entwicklungszeit verkürzt und den Wartungsaufwand minimiert.

4. RESTful API zur Kommunikation zwischen Frontend und Backend:

   - Beschreibung: Die Kommunikation zwischen Frontend und Backend erfolgt über klar definierte API-Endpunkte.
   - Argument: Diese Methode gewährleistet eine lose Kopplung zwischen den Systemen, wodurch sich das Backend oder Frontend unabhängig voneinander anpassen lässt.

5. MariaDB als Datenbank:

   - Beschreibung: MariaDB wurde für die Datenbank ausgewählt, um Daten zuverlässig zu speichern und abzufragen.
   - Argument: Die Datenbank bietet hohe Leistung und Skalierbarkeit, die für ein wachsendes Wiki-Projekt erforderlich sind.

Software Tools/Platforms/Techniken:

1. Frontend:

   - Technologie: ReactJS
     - Beschreibung: Ermöglicht die Erstellung eines responsiven und dynamischen Frontends. Der Fokus liegt auf Dark Mode und intuitivem Design.

- Argument: React-Komponenten ermöglichen eine modulare Entwicklung und reduzieren den Entwicklungsaufwand.

2. Backend:

- Technologie: Node.js
  - Beschreibung: Verarbeitung von API-Anfragen und Verbindung zur Datenbank.
  - Argument: Bietet eine schnelle und skalierbare Plattform für serverseitige Anwendungen.

3. Datenbank:

- Technologie: MariaDB
  - Beschreibung: Speicherung und Verwaltung von Wiki-Daten.
  - Argument: Ermöglicht komplexe Abfragen und bietet eine zuverlässige Lösung für relationale Daten.

4. Entwicklungstools:

- IDE: Visual Studio Code
  - Beschreibung: Primäre Entwicklungsumgebung.
  - Argument: Bietet integrierte Unterstützung für React und Node.js sowie Erweiterungen für Produktivitätssteigerung.
- Versionskontrolle: GitHub
  - Beschreibung: Verwaltung des Projektcodes und Zusammenarbeit im Team.
  - Argument: Ermöglicht kollaboratives Arbeiten und Versionsmanagement.

5. Dokumentationstools:

- visual-paradigm.com: Erstellung von UML-Diagrammen wie Use Case, Klassendiagrammen und Sequenzdiagrammen.
- sequencediagram.org: Erstellung von Sequenzdiagrammen.
- GitHub: Dokumentierung der Arbeitspakete und Sprints.

6. Design und Animation:

- CSS: Verwendung moderner CSS-Techniken für Animationen, Dark Mode und Responsive Design.
- Icons: Minimalistische SVG-Icons zur Verbesserung der Benutzeroberfläche.