SEP Projektmappe

**Projektmappe des Projektes**

SchachEP

**Dokumentation des Projektes**

Gruppe I:

Ayoub Ouamar

Seher Tezer

Jonas Giesen

Kaan Hisiroglu

Yusuf Cobandir

Mario Mai

**Hinweis**

An vielen Stellen findet Ihr im Dokument folgendes Kästchen:

*Dies ist eine Hilfestellung.*

Diese Kästen dienen dazu, Euch kurze Informationen über Ziele und Inhalte der jeweiligen Abschnitte zu geben. Auch die Beispiele und Templates dienen dazu, euch bei der Dokumentation eures Projektes zu unterstützen. **Sowohl die Kästchen als auch die Beispiele und Templates sind spätestens zur finalen Abgabe der Projektmappe vollständig zu entfernen.** Betrachtet dieses Dokument bitte nicht als Aufgabe, die man von oben nach unten abarbeiten soll; es soll vielmehr als durchgängige Dokumentation eurer Projektarbeit dienen und fortlaufend erweitert bzw. angepasst werden, sodass am Ende des SEPs der Entwicklungsprozess Eurer Software vollständig dokumentiert ist.

Das SEP-Team wünscht Euch  
**Viel Erfolg**

Inhalt

[Projektbeschreibung 4](#_Toc148525188)

[Zyklus I 7](#_Toc148525189)

[Spezifikationsplanung 7](#_Toc148525190)

[User-Stories 8](#_Toc148525191)

[Papierprototypen 9](#_Toc148525192)

[Strukturdiagramme (Komponenten- und Klassendiagramme) 9](#_Toc148525193)

[Verhaltensdiagramme (Kommunikationsdiagramme) 9](#_Toc148525194)

[Funktionalitätsplanung 10](#_Toc148525195)

[Systemtests 11](#_Toc148525196)

[Zyklus II 13](#_Toc148525197)

[Spezifikationsplanung 13](#_Toc148525198)

[User-Stories 13](#_Toc148525199)

[Papierprototypen 14](#_Toc148525200)

[Strukturdiagramme (Komponenten- und Klassendiagramme) 14](#_Toc148525201)

[Verhaltensdiagramme (Kommunikationsdiagramme) 14](#_Toc148525202)

[Funktionalitätsplanung 15](#_Toc148525203)

[Modultests 16](#_Toc148525204)

[Systemtests 16](#_Toc148525205)

[Zyklus III 17](#_Toc148525206)

[Spezifikationsplanung 17](#_Toc148525207)

[User-Stories 17](#_Toc148525208)

[Papierprototypen 18](#_Toc148525209)

[Strukturdiagramme (Komponenten- und Klassendiagramme) 18](#_Toc148525210)

[Verhaltensdiagramme (Kommunikationsdiagramme) 18](#_Toc148525211)

[Funktionalitätsplanung 19](#_Toc148525212)

[Modultests 20](#_Toc148525213)

[Systemtests 20](#_Toc148525214)

[Nutzerhandbuch 21](#_Toc148525215)

[Technische Anforderungen 21](#_Toc148525216)

[Installationsanleitung 21](#_Toc148525217)

[Bedienungsanleitung 21](#_Toc148525218)

# Projektbeschreibung

Einleitung

Das SchachEP soll ein System werden, dass den Anwendern das Spielen von Schachpartien ermöglicht. Dabei ermöglicht das Systeme es Schachpartien mit Freunden oder Computer-Gegnern zu spielen. Außerdem verfügt es über ein Punktesystem und einem Leaderboard. Ferner beinhaltet es SinglePlayer Schachpuzzle sowie die Möglichkeit spannende Schachspiele zu streamen.

Zyklus 1

(Allgemeines) Im ersten Zyklus soll zunächst die Grundstruktur des Systems implementiert werden. Dem System soll eine Service-Architektur (bestehend aus mindestens einem Frontend-Service und einem BackendService) zu Grunde liegen. Sämtliche Daten des Systems sollen persistent gespeichert werden.

(Registrierung eines Nutzers) Nutzer sollen die Möglichkeit haben sich zu registrieren und somit ein Profil zu erstellen. Ein Nutzerprofil soll aus dem Vor- und Nachnamen des Nutzers, einer gültigen E-Mail-Adresse, dem Geburtsdatum des Nutzers, einem Passwort, Punkte (vergleichbar mit ELO) und einem optionalen Profilbild bestehen. Jeder Spieler startet mit 500 Punkten. Ein Nutzer soll in der Lage sein, sich sein Profil anzuschauen. Ein Nutzer soll ebenfalls die Möglichkeit haben, sich im System mittels seiner Zugangsdaten einzuloggen. Der Login-Vorgang soll durch eine Zwei-Faktor-Authentifizierung gesichert werden, bei der, neben dem Passwort, ein zufällig generierter Sicherheitscode eingegeben werden muss, der per E-Mail an den Nutzer versendet wird. Für die Abnahmen: Es soll ein „super“ Sicherheitscode existieren, der ebenfalls akzeptiert wird.

(Freundesliste) Nutzer sollen die Möglichkeit haben, andere Nutzer als Freunde hinzuzufügen. Eine Freundesliste soll für jeden Nutzer angelegt werden, in der er seine Freunde sehen und verwalten kann. Der Nutzer soll in der Lage sein, Freundschaftsanfragen zu senden und anzunehmen oder abzulehnen. Wenn ein Nutzer eine Freundschaftsanfrage erhält, soll er darüber per E-Mail benachrichtigt werden. Die Freundesliste soll öffentlich oder privat einstellbar sein, so dass andere Nutzer sehen können, wer in der Liste enthalten ist, oder ob sie privat ist. Ein Nutzer kann sich das Profil von jedem seiner Freunde anschauen.

(Schachpartie erstellen) Ein Benutzer kann eine Schachpartie mit einem eindeutigen Namen und einen Timer erstellen und einen weiteren Benutzer (entweder ein Benutzer, welcher durch eine Suchfunktion gefunden wurde oder einen Freund) einladen. Der eingeladene Benutzer kann die Einladung annehmen oder ablehnen. Nimmt der eingeladene Benutzer die Einladung an, können beide Spieler einen Platzhalter sehen. Jeder Spieler kann die Schachpartie ausblenden (nicht diese verlassen!) und nachträglich wieder beitreten.

Zyklus 2

(Chat) Nutzer sollen die Möglichkeit haben, mit ihren Freunden über das System zu chatten. Hierfür soll eine Chat-Funktion implementiert werden, der es den Nutzern ermöglicht, private oder Gruppenchats zu erstellen und Nachrichten zu senden. Die Nutzer sollen in der Lage sein, Nachrichten in Echtzeit zu senden und zu empfangen. Außerdem sollen die Nutzer die Möglichkeit haben, Nachrichten zu löschen oder zu bearbeiten, solange der Empfänger sie noch nicht gelesen hat.

(Schachpartie spielen) Eine erstellte Schachpartie soll nun spielbar werden. Dazu sollen ein Sachbrett mit allen Schachfiguren sichtbar sein, sobald eine Schachpartie geöffnet wird. Weiterhin sollen die offiziellen Schachregeln implementiert werden (z.B. abwechselnde Züge zwischen den Spielern, erlaubten Züge der Figuren auf dem Feld, Schlagen von Figuren auf dem Feld, Rochade, etc.)1 . Außerdem soll ein Timer für jeden Spieler implementiert werden, nach dessen Ablauf der entsprechende Spieler das Spiel verloren hat. Während ein Spieler nicht am Zug ist, ist der Timer pausiert. Weiterhin soll ein Spieler zur jeder Zeit die Partie aufgeben können. Beim Gewinn einer Schachpartie erhält der Gewinner 10 Punkte und der Verlierer verliert 10 Punkte. Im Falle eines Remis sollen keine Punkte vergeben werden.

(Schachclub) Ein Benutzer kann einen Schachclub mit einem eindeutigen Namen erstellen. Jeder Nutzer kann die Liste aller erstellten Schachclubs einsehen. Weiterhin ist jeder Benutzer in der Lage einen Schachclub beizutreten. Nutzer innerhalb eines Schachclubs können miteinander über einen Chat in Echtzeit kommunizieren (siehe Chat). Außerdem soll im Profil eines jeden Nutzers ersichtlich sein, in welchen Schachclubs er beigetreten ist.

(Schachpuzzle) Insgesamt sollen 10 vordefinierte Schachpuzzles existieren, welche von einem Benutzer in das System mittels einer CSV-Datei2 importiert werden können. Ein Schachpuzzle zeigt dabei ein laufendes Schachspiel. Das Puzzle besteht darin, dass der Benutzer den besten nächsten Zug ermitteln muss. Sobald der Benutzer 3 vordefinierten Schachpuzzle erfolgreich löst, erhält er eine Anerkennung auf seinem Profil mit der Aufschrift „Schachexperte“.

(Leaderboard) Jeder Benutzer kann ein Leaderboard einsehen, welches die besten Spieler anhand ihrer Punkte in absteigender Folge anzeigt.

Zyklus 3

(Spielhistorie) Jeder Spieler soll in seinem Profil die drei zuletzt gespielten Partien einsehen können. Hierbei sollen Name und Punkte der Teilnehmer sowie das Ergebnis angezeigt werden.

(Schachspiel wiederholen) Ein Benutzer soll in der Lage sein sich den Spielverlauf eines jeden Spieles der Spielhistorie nachträglich anzuschauen. Dabei soll es möglich sein, die Züge manuell umzuschalten (d.h. sich den nächsten bzw. vorherigen Zug anzuschauen).

(Export/Import von Partien) Der Verlauf einzelner Spiele aus der Spielhistorie soll als PGN3 exportiert werden können. Weiterhin soll ein Spieler in der Lage sein, in einer PGN-Datei gespeicherte Züge einer Partie zu importieren. Anschließend soll er die einzelnen Züge dieses Spiels auf dem Schachbrett anzeigen lassen können. Die Funktionalität entspricht der des Features Schachspiel wiederholen.

(Schachpartie gegen Computer spielen) Ein Spieler kann eine Schachpartie mit einem Computer-Gegner starten4 . Der Computer-Gegner soll dabei in mindestens drei Schwierigkeitsstufen einstellbar sein. Das Spielverlauf gleicht dem der Funktionalität Schachpartie spielen.

(Streaming von Live-Spielen) Spieler sollen in der Lage sein, live gespielte Spiele öffentlich zu streamen. Weiterhin sollen Spieler, welche sich ein live-gestreamtes Spiel anschauen wollen, eine Liste sehen, in welcher alle aktuell gestreamten Spiele angezeigt werden. Aus dieser Liste können die Spieler ein gestreamtes Spiel auswählen und dieses anschließend live anschauen.

(Assistentenbot) Spieler sollen in der Lage sein einen Assistentenbot um Rat hinsichtlich des nächsten zu spielenden Zuges einzuholen. Dieser nächste zu spielende Zug soll dabei der beste nächste zu spielende Zug sein. Für die Hilfe des Assistentenbot müssen die Spieler allerdings einen Punkt bezahlen.

# Zyklus I

## Spezifikationsplanung

Jedes Artefakt, das im Rahmen des SEP erstellt wird, muss mit dem Namen genau einer Verantwortlichen/eines Verantwortlichen versehen werden. Das bedeutet, dass jede User Story, jeder Papierprototyp, jedes Komponentendiagramm, jedes Klassendiagramm, jedes Kommunikationsdiagramm und sämtliche Tests mit dem Namen der/des Verantwortlichen versehen und hier in der untenstehenden Tabelle entsprechend eingetragen werden muss. Natürlich kann die Gruppe gemeinsam an einem Artefakt arbeiten, als Verantwortlicher sollte aber genau eine Person eingetragen werden.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Artefakt** | **Art des Artefakts** | **Verantwortlicher** | **Status** |
| **1.** | Registrierung der Nutzer |  |  |  |
| 1.1 |  |  |  |  |
| 1.2 |  |  |  |  |
| 1.2 |  |  |  |  |
| **2.** | Freundesliste |  |  |  |
| 2.1 | Hinzufügen und Verwalten von Freunden der Freundesliste | User Story |  | In Bearbeitung |
| 2.2 | Freundschaftsanfragen | User Story |  | In Bearbeitung |
| 2.3 | Benachrichtigung der Freundschaftsanfrage per E-Mail | User Story |  | In Bearbeitung |
| 2.4 | Sichtbarkeit der Freundesliste | User Story |  | In Bearbeitung |
| 2.5 | Profile der Freunde | User Story |  | In Bearbeitung |
| 2.6 | Profil Fenster Freund | Papierprototyp |  | In Bearbeitung |
| 2.7 | Fenster Freundesliste | Papierprototyp |  | In Bearbeitung |
| 2.8 | Fenster Verwaltung der Freundesliste | Papierprototyp |  | In Bearbeitung |
| 2.9 | Pop-Up Freundschaftsanfrage | Papierprototyp |  | In Bearbeitung |

## User-Stories

**Template:**

|  |  |
| --- | --- |
| **User Story-ID** | <Eindeutiger Identifizierer> |
| **User Story-Beschreibung** | <Text der User Story mittels Satzschablone:  Als <Rolle> möchte ich <Ziel> [, um/sodass <Nutzen>]  (s. Foliensatz „Anforderungen“)> |
| **Geschätzter Realisierungsaufwand** | <Einschätzung der Zeit, die benötigt wird, um die Userstory zu implementieren> |
| **Priorität** | <Wichtigkeit der User Story hinsichtlich der Aufgabenstellung z.B. hoch, mittel niedrig> |
| **Autor** | <Hier bitte nur einen Zuständigen eintragen z.B. Max Mustermann> |
| **Abhängigkeiten zu anderen User Stories** | <Auflistung verwandter User Stories> |

**Schlechtes Beispiel:**

|  |  |
| --- | --- |
| **User Story-ID** |  |
| **User Story-Beschreibung** | Ich möchte ich rechtzeitig informiert werden, wenn ein Patient einen Termin nicht wahrnimmt. |
| **Geschätzter Realisierungsaufwand** | 1337 |
| **Priorität** | - |
| **Autor** | Emmett Brown, Rick Sanchez, Amelia Pond |
| **Abhängigkeiten zu anderen User Stories** |  |

**Gutes Beispiel:**

|  |  |
| --- | --- |
| **User Story-ID** | 1.6 |
| **User Story-Beschreibung** | Als Arzt möchte ich mindesten fünf Minuten vor dem Termin informiert werden, wenn ein Patient einen Termin nicht wahrnimmt, sodass ich andere Patienten vorziehen kann. |
| **Geschätzter Realisierungsaufwand** | 2 Tage |
| **Priorität** | Hoch |
| **Autor** | Emmett Brown |
| **Abhängigkeiten zu anderen User Stories** | 1.3, 1.5 |

Papierprototypen

Das Erstellen eines Papierprototypen dient als Methode des Brainstormings, Designs, Herstellens, Testens und des Kommunizierens von Benutzer Interfaces.

## Strukturdiagramme (Komponenten- und Klassendiagramme)

Im SEP soll die statische Struktur des Systems mittels Komponenten- und Klassendiagramme modelliert werden. Ein Komponenten- und Klassendiagramme dienen der grafischen Darstellung von Komponenten/Klassen, Schnittstellen und deren Beziehungen. Die Diagrammtypen helfen dabei, Quellcode und Implementierungsarbeiten zu strukturieren, bevor diese starten und ermöglicht eine Aufteilung der Programmieraufgaben.

## Verhaltensdiagramme (Kommunikationsdiagramme)

Im SEP soll das dynamische Verhalten des Systems mittels Kommunikationsidagramme modelliert werden. Ein Kommunikationsdiagramme ermöglicht die grafische Darstellung des Nachrichtenaustausches zwischen Systemobjekten. Systemobjekte können Komponenten im Komponentendiagramm und Klassen im Klassendiagramm sein. Kommunikationsdiagramme zielen darauf ab, die Zusammenarbeit der Systemobjekte darzustellen

## Funktionalitätsplanung

Anhand der Funktionalitätsplanung werdet Ihr geprüft. Diese sollte dementsprechend **immer** aktuell gehalten werden und pro Funktionalität **nur einen** Verantwortlichen enthalten. Als Quellcodereferenz solltet Ihr immer euer Package, eure Klasse und die dazugehörige Methode angeben. ggf. könnt Ihr auch Zeilenangaben machen.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Funktionalität** | **Verantwortlicher** | **Abhängige**  **Funktionalitäten** | **Verknüpfte**  **User-Stories** | **Quellcode-**  **referenz** | **Status** |
| **1.** | **Registrierung von Admins** |  |  |  |  |  |
| 1.1 | Registrierungsview im Frontend | Max Muster | 1.2 |  |  | Fertig |
| 1.2 | Datenbank-modell des Admins | Max Muster | 1.3, 1.4 |  |  | WIP |
| 1.2.1 | …. |  |  |  |  |  |
| **2.** | **Anliegen von Liegen** |  |  |  |  |  |
| 2.1 | Ligaübersicht  im Frontend | Anja Muster | 2.4, 2,7 |  |  | Fertig |
| … |  |  |  |  |  |  |

## Systemtests

Systemtests sind Tests des Gesamtsystems gegen die Anforderungen nach erfolgreicher Integration. Eingaben und Sollverhalten werden dabei aus der Anforderungs-spezifikation abgeleitet.

Die Systemtests werden von Eurer Parallelgruppe spezifiziert und durchgeführt, daher ist dieser Bereich von den Mitgliedern der Parallelgruppe auszufüllen.

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** | 03.03.2019 | | |
| **Tester** | Martina Musterfrau | | |
| **SW-Version** | V 0.1.2 | | |
| **Vorbedin-gung(en)** | Nutzer „Max Mustermann“ ist am System mit Passwort „geheim“ registriert | | |
| **Schritt** | **Aktion (User)** | **Erwartete Reaktion (System)** | **√ / X** |
| 1 | Der Benutzer gibt den Benutzername „Max Mustermann“ auf der Tastatur ein. | Das System zeigt „Max Mustermann“ auf dem Display an. | **√** |
| 2 | Der Benutzer gibt das Passwort „geheim“ auf der Tastatur ein. | Das System zeigt das Passwort durch „\*“-Symbole zensiert an. | **√** |
| 3 | Der Benutzer klickt auf „Anmelden“. | Das System zeigt die Meldung „Anmeldung erfolgreich“ auf dem Display an. | **X** |
| **Nachbe-dingung(en)** | Nutzer ist am System angemeldet, Anmeldezeitpunkt ist im System gespeichert. | | **X** |
| **Testurteil** | **Test nicht bestanden.** | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** | 03.03.2019 | | |
| **Tester** | Martina Musterfrau | | |
| **SW-Version** | V 0.1.2 | | |
| **Vorbedin-gung(en)** | Nutzer „Max Mustermann“ ist am System mit Passwort „geheim“ registriert | | |
| **Schritt** | **Aktion (User)** | **Erwartete Reaktion (System)** | **√ / X** |
| 1 | Der Benutzer gibt den Benutzername „Max Mustermann“ auf der Tastatur ein. | Das System zeigt „Max Mustermann“ auf dem Display an. | **√** |
| 2 | Der Benutzer gibt das Passwort „geheim“ auf der Tastatur ein. | Das System zeigt das Passwort durch „\*“-Symbole zensiert an. | **√** |
| 3 | Der Benutzer klickt auf „Anmelden“. | Das System zeigt die Meldung „Anmeldung erfolgreich“ auf dem Display an. | **√** |
| **Nachbe-dingung(en)** | Nutzer ist am System angemeldet, Anmeldezeitpunkt ist im System gespeichert. | | **√** |
| **Testurteil** | **Test bestanden.** | | |

# Zyklus II

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Artefakt** | **Art des Artefakts** | **Verantwortlicher** | **Status** |
|  |  |  |  |  |
| … |  |  |  |  |
|  |  |  |  |  |
| … |  |  |  |  |
| … |  |  |  |  |

## Spezifikationsplanung

## User-Stories

Template:

|  |  |
| --- | --- |
| **User Story-ID** |  |
| **User Story-Beschreibung** |  |
| **Geschätzter Realisierungsaufwand** |  |
| **Priorität** |  |
| **Autor** |  |
| **Abhängigkeiten zu anderen User Stories** |  |

Papierprototypen

## Strukturdiagramme (Komponenten- und Klassendiagramme)

## Verhaltensdiagramme (Kommunikationsdiagramme)

## Funktionalitätsplanung

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Funktionalität** | **Verantwortlicher** | **Abhängige**  **Funktionalitäten** | **Verknüpfte**  **User-Stories** | **Quellcode-**  **referenz** | **Status** |
| **1.** |  |  |  |  |  |  |
| 1.1 |  |  |  |  |  |  |
| 1.2 |  |  |  |  |  |  |
| 1.2.1 | …. |  |  |  |  |  |
| **2.** |  |  |  |  |  |  |
| 2.1 |  |  |  |  |  |  |
| … |  |  |  |  |  |  |

## Modultests

Modultests sind Komponententests. Diese werden in der Softwareentwicklung angewendet, um die funktionalen Einzelteile (Units) von Computerprogrammen zu testen.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Getestete Funktionalität** | **Quellcode Referenz** | **Status** |
|  |  |  |  |

## Systemtests

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** |  | | |
| **Tester** |  | | |
| **SW-Version** |  | | |
| **Vorbedin-gung(en)** |  | | |
| **Schritt** | **Aktion (User)** | **Erwartete Reaktion (System)** | **√ / X** |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| **Nachbe-dingung(en)** |  | |  |
| **Testurteil** |  | | |

# Zyklus III

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Artefakt** | **Art des Artefakts** | **Verantwortlicher** | **Status** |
|  |  |  |  |  |
| … |  |  |  |  |
|  |  |  |  |  |
| … |  |  |  |  |
| … |  |  |  |  |

## Spezifikationsplanung

## User-Stories

Template:

|  |  |
| --- | --- |
| **User Story-ID** |  |
| **User Story-Beschreibung** |  |
| **Geschätzter Realisierungsaufwand** |  |
| **Priorität** |  |
| **Autor** |  |
| **Abhängigkeiten zu anderen User Stories** |  |
| **Zugehörige Szenarien** |  |

Papierprototypen

## Strukturdiagramme (Komponenten- und Klassendiagramme)

## Verhaltensdiagramme (Kommunikationsdiagramme)

## Funktionalitätsplanung

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Funktionalität** | **Verantwortlicher** | **Abhängige**  **Funktionalitäten** | **Verknüpfte**  **User-Stories** | **Quellcode-**  **referenz** | **Status** |
| **1.** |  |  |  |  |  |  |
| 1.1 |  |  |  |  |  |  |
| 1.2 |  |  |  |  |  |  |
| 1.2.1 | …. |  |  |  |  |  |
| **2.** |  |  |  |  |  |  |
| 2.1 |  |  |  |  |  |  |
| … |  |  |  |  |  |  |

## Modultests

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Getestete Funktionalität** | **Quellcode Referenz** | **Status** |
|  |  |  |  |

## Systemtests

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** |  | | |
| **Tester** |  | | |
| **SW-Version** |  | | |
| **Vorbedin-gung(en)** |  | | |
| **Schritt** | **Aktion (User)** | **Erwartete Reaktion (System)** | **√ / X** |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| **Nachbe-dingung(en)** |  | |  |
| **Testurteil** |  | | |

# Nutzerhandbuch

## Technische Anforderungen

Technische Mindestanforderungen, welche das System benötigt, um wie gewünscht bedienbar zu sein.

## Installationsanleitung

Genaue Erläuterung, wie das entwickelte System vollkommen funktionsfähig auf einem Rechner in Betrieb genommen werden kann.

## Bedienungsanleitung

Genaue Erläuterung, wie das entwickelte System zu bedienen ist.