Pflichtenheft

|  |  |
| --- | --- |
| Projektbezeichnung | Vier gewinnt |
| Projektleiter | Christian Chimani |
| Erstellt am | 10.03.2022 |
| Letzte Änderung am | 10.03.2022 |
| Status | in Bearbeitung |
| Aktuelle Version | 1.1 |

Änderungsverlauf

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nr. | Datum | Version | Geänderte Kapitel | Art der Änderung | Autor | Status |
| 1 | 10.03.2022 | 1.0 | Alle | Erstellung | Wittner Michael | - |
| 2 | 14.01.2022 | 1.1 |  | Korrekturen | 3AHITN | - |
| 3 |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |

fg         fertiggestellt

iB         in Bearbeitung

ab        abgebrochen

Pr        Prüfung

paus    pausiert

Inhalt

1       Zielbestimmung. 3

1.1      Musskriterien. 3

1.2      Wunschkriterien. 4

2       Allgemeines. 4

2.1      Ausgangssituation. 4

2.2      Team.. 4

3       Funktionale Anforderungen. 5

3.1      Konsolenanwendung. 5

3.2      Grafische Anwendung. 6

3.3      Erweiterung Einzelspielermodus. 7

4       Nichtfunktionale Anforderungen. 7

4.1      Programmiersprache / Programmierumgebung. 7

4.2      Diagramme. 8

4.3      Programmierrichtlinien / Dokumentation. 8

4.4      Softwareversionierung. 8

4.5      Arbeitsbericht 8

5       Rahmenbedingungen. 9

5.1      Zeitplan. 9

6       Liefer- und Abnahmebedingungen. 9

**1**      **Zielbestimmung**

Im vorliegenden Pflichtenheft werden alle funktionalen und nichtfunktionalen Anforderungen an das Spiel „Vier gewinnt“ beschrieben.

Es muss eine einfach zu bedienende Anwendung erstellt werden, welche eine Smart Watch bedient. Die Implementierung hat in den eingeteilten Gruppen zu erfolgen. Um eine einfache Projektorganisation zu gewährleisten, ist ein Softwareverwaltungssystem einzusetzen.

**1.1**   **Musskriterien**

·    Es muss die Uhrzeit ausgegeben werden

·    Zur vollen Stunde kommt eine Benachrichtigung

·    Aktuelles Datum

·    Aktuelles Wetter wird mit Symbolen angezeigt(Wolke/Sonne)

·    Kalenderwoche

·       Stoppuhr

 ·    Night/Day Mode

**1.2**   **Wunschkriterien**

·       Eine UNDO-Funktion muss ermöglicht werden, d.h. ein Zug muss rückgängig gemacht werden können

·       Eine Restart-Funktion muss ermöglicht werden

·       Ein Einzelspielermodus wird ermöglicht (KI-Modus)

**2**      **Allgemeines**

**2.1**   **Ausgangssituation**

Im letzten Projekt wurde ein Spiel nach dem MVC-Prinzip aufgebaut. Es wurde die Aufteilung in verschieden Klassen vorgegeben.

Innerhalb der Model-Klassen müssen alle für die Uhrlogik notwendigen Teile enthalten sein.

Die View-Klassen dienen zur Darstellung der Daten aus den Model Klassen, es darf keine direkte Beziehung zwischen Model und View-Klassen geben. Als Schnittstelle zwischen diesen dient die Controller Klasse. Diese Klasse ist für die Ausführung der Daten von der Uhr verantwortlich.

Nach Aktualisierung der Daten durch das Model müssen diese mit Hilfe der View-Klassen dargestellt werden.

Das vorliegende Projekt muss nach dem gleichen Prinzip aufgebaut werden.

**2.2**   **Team**

In der nachfolgenden Tabelle sind die Daten zu den Teammitgliedern einzutragen. Seite 1 ist zu ändern, das Dokument muss in das Git-Repository geladen werden.

|  |  |  |  |
| --- | --- | --- | --- |
| Rolle(n) | Name | Git-Account | E-Mail |
| Projektleitung | Christian Chimani | cchimani | cchimani@htl-steyr.ac.at |
| Entwicklung | Constantin Riedler | criedler | criedler@htl-steyr.ac.at |
| Entwicklung | Elias Miklautsch | emiklaut | emiklaut@htl-steyr.ac.at |
|  |  |  |  |

**3**      **Funktionale Anforderungen**

* Es muss die Uhrzeit ausgegeben werden
* Zur vollen Stunde kommt eine Benachrichtigung
* Aktuelles Datum
* Aktuelles Wetter wird mit Symbolen angezeigt(Wolke/Sonne)
* Kalenderwoche
* Stoppuhr
* Night/Day Mode
* 12h/24h Zeitmodus umstellen

**3.1**   **Konsolenanwendung**

Die Uhrzeit und das Datum kann beim Starten mitgegeben, ansonsten werden diese Daten automatisch bezogen.  
Zu

**Wunsch-Funktionen**

·    Luftfeuchtigkeit

·       Temperatur

·       Geburtstag eintragen

**3.2**   **Grafische Anwendung**

Beim Starten der Applikation wird ein Fenster aufgerufen um seine Uhr einzustellen.

Wenn keine Daten eingegeben werden hat die Uhr folgende Starteinstellungen:

* aktuelles Datum
* aktuelle Uhrzeit
* aktuelle KW

 ·       Nachdem Datum, Uhrzeit und KW automatisch bezogen werden, wird die Anzeige gestartet und es können Eintellungen an der Uhr vorgenommen werden.

·       Das starten der Stoppuhr muss mit Hilfe eines Mausklicks auf den Stoppuhr-Button ermöglicht werden. Wird außerhalb des Spielfeldes geklickt bzw. in eine volle Spalte geklickt, so muss ein Hinweis in der Statuszeile des Programmes ausgegeben werden.

Der Hinweis muss durch den nächsten Mausklick wieder gelöscht werden.

·       Nach jedem gültigen Spielzug muss geprüft werden, ob das Spiel gewonnen wurde. Ist dies der Fall, muss dem Gewinner des Spieles gratuliert werden.

Außerdem muss geprüft werden, ob das Spiel unentschieden ausgegangen ist. Dies ist dann der Fall, wenn alle Spalten voll sind, und keiner der Spieler 4 Spielsteine in einer Reihe hat. Eine entsprechende Meldung muss ausgegeben werden

·       Nach dem Ende eines Spieles soll das Spiel durch einen „Restart“ Button neu gestartet werden können.

·       Die Anwendung muss ein Menü beinhalten, welches

o   es ermöglicht die Daten zu einem Spieler zu ändern

o   eine Hilfe ausgibt (wie ist das Programm zu verwenden, wie lauten die Spielregeln)

**Wunsch-Funktionen**

·    Luftfeuchtigkeit

·       Temperatur

·       Geburtstag eintragen

**3.3**   **Erweiterung Einzelspielermodus**

Die einfachste Form eines Einzelspielermodus ist es, dass der Computer-Spieler in eine zufällige Spalte wirft. Dieses System ist jedoch nicht wirklich „intelligent“. Eine bessere Möglichkeiten wäre dass der Computer-Spieler die unterschiedlichen Stellungen, die sich nach einem Einwurf ergeben bewertet. Er wirft in die Spalte mit dem besten Zug. Dabei gilt:

·       Eine 4er Reihe ist das Ziel – somit die beste Möglichkeit·       Ein 3er im nächsten Zug ist z.B. besser als ein Zug, in dem kein 3er entsteht.

·       Ein 2er im nächsten Zug ist besser als ein Zug, indem kein 2er entsteht.

**4**      **Nichtfunktionale Anforderungen**

**4.1**   **Programmiersprache / Programmierumgebung**

Die Anwendung muss in der Programmiersprache Java umgesetzt werden. Für die grafische Anwendung ist JavaFX einzusetzen.

Als JavaFX Version muss die Version verwendet werden, welche am Schulserver unter

//acdc/software/Updates/Microsoft/Windows10/Applications/Java

zu finden ist.

Als Entwicklungsumgebung muss IntelliJ der Firma JetBrains (<https://www.jetbrains.com/de-de/idea/>) eingesetzt werden.

**4.2**   **Diagramme**

Für die Erstellung von Diagrammen muss [http://www.draw.io](http://www.draw.io/) oder [https://app.genmymodel.com](https://app.genmymodel.com/) verwendet werden. Es ist darauf zu achten die Diagramme so in der Versionsverwaltungssoftware zu speichern, dass diese weiter bearbeitet werden können.

Änderungen in Diagrammen müssen dokumentiert (einschließlich Begründung) werden.

**4.3**   **Programmierrichtlinien / Dokumentation**

Bei der Programmierung ist auf die Einhaltung der Programmierrichtlinien der HTL Steyr zu achten – siehe

<http://www.htl-steyr.ac.at/intern/wiki/doku.php?id=el-it:fsst:softwareentwicklung:programmierrichtlinien>

Die Inline-Dokumentation hat den Richtlinien zu entsprechen, welche im obigen Dokument beschrieben werden.

Die „Readme.md“ Datei muss alle für das Projekt notwendigen Informationen beinhalten.

**4.4**   **Softwareversionierung**

Für die Teamarbeit sowie für die Softwareversionierung ist GitHub zu verweden. Am Ende jedes Unterrichtsblocks müssen Änderungen in das Repository hochgeladen werden. Dabei ist darauf zu achten, dass im main-Branch immer ein lauffähiges Programm existieren muss.

Die Entwicklung der einzelnen Teilbereiche hat in eigenen Branches zu erfolgen. Jedes Teammitglied muss in seinem Branch arbeiten.

Alle Dateien welche für das Projekt notwendig sind müssen in GitHub enthalten sein.

**4.5**   **Arbeitsbericht**

Im Repository muss für jedes Teammitglied eine „.md“-Datei für einen Arbeitsbericht erstellt werden. Im Arbeitsbericht muss für jede Unterrichtsstunde von jedem Teammitglied dokumentiert werden, woran gearbeitet wurde, welche Probleme aufgetreten sind, und welche Teile fertig implementiert wurden.

**5**      **Rahmenbedingungen**

Vor der Implementierung ist ein Klassendiagramm zu erstellen welches alle notwendigen Klassen für die Konsolen- und für die grafische Anwendung beinhaltet.

**5.1**   **Zeitplan**

Tag 1 (2h):            Besprechung Pflichtenheft

Tag 2 (2h):            Entwurf Klassendiagramm, GitHub-Repository

Tag 3,4 (je 2h):     Umsetzung/Test Model Klassen

Tag 5 (2h):            Umsetzung/Test View Klassen Consolen Anwendung

Tag 6,7 (je 2h):     Umsetzung/Test View-Klassen GUI Anwendung

**6**      **Abgabebedingungen - Notengebung**

Die fertige (Teil-) Software ist am Abgabedatum in GitHub abzulegen.

Im „Main“-Branch ist

·       die fertige Konsolenanwendung mit dem Tag „Console“ zu benennen

·       die fertige GUI-Anwendung mit dem Tag „Gui“ zu benennen

Nicht lauffähige Projekte werden negativ beurteilt.

Während der Zeit des Projektes werden Aufzeichnungen zu folgenden Bereichen erstellt:

·       Einhaltung der zeitlichen Vorgaben

·       Arbeiten mit Versionskontrollsoftware

·       Kommunikation innerhalb des Teams

·       Dokumentation der Fortschritte