

Projekt Mastermind

Software Requirement Specification

Vorgelegt am 23-03-2017

|  |  |  |  |
| --- | --- | --- | --- |
| Datum | Version | Beschreibung | Autor |
| 16.03.17 | 1.0 | Erste Version | Schaude |
| 21.03.17 | 1.1 | Layout Anpassungen | Potsch |
| 28.03.17 | 1.2 | Use-Case Diagramm hinzugefügt | Potsch |
| 03.04.17 | 1.3 | Testplan hinzugefügt | Maier, Potsch |

Inhaltsverzeichnis

[Anforderungen 3](#_Toc478989520)

[Funktionale Anforderungen 3](#_Toc478989521)

[Nicht funktionale Anforderungen 3](#_Toc478989522)

[Anforderung an technische Realisierung 3](#_Toc478989523)

[Anforderung an den Projektablauf 3](#_Toc478989524)

[Use Cases 4](#_Toc478989525)

[Use Case Diagramm 4](#_Toc478989526)

[Beschreibungen der Use Cases 5](#_Toc478989527)

[Use Case MM01 - Schwierigkeitsgrad festlegen 5](#_Toc478989528)

[Use Case MM02 - Aussehen des Spielbretts anpassen 6](#_Toc478989529)

[Use Case MM03 – Spielmodus auswählen 7](#_Toc478989530)

[Use Case MM04 – Hilfefunktion in Anspruch nehmen 8](#_Toc478989531)

[Use Case MM05 - Farbcodes während des Spiels festlegen 9](#_Toc478989532)

[Use Case MM06 – Ausgabe S/W Pins 10](#_Toc478989533)

[Use Case MM07 – Spiel unterbrechen 11](#_Toc478989534)

[Use Case MM08 - Beenden des Spiels bei richtigem Code 12](#_Toc478989535)

[Use Case MM09 - Beenden des Spiels nach x Versuchen 13](#_Toc478989536)

[Use Case MM10 – Zug rückgängig machen 14](#_Toc478989537)

[Use Case MM11 – Scoreboard anzeigen 15](#_Toc478989538)

[Priorisierung 16](#_Toc478989539)

[Test Plan 17](#_Toc478989540)

[Unit-Tests 17](#_Toc478989541)

[Anwendungstests 18](#_Toc478989542)

[Performance-Tests 19](#_Toc478989543)

[Recovery-Tests 20](#_Toc478989544)

[Inbetriebnahme-Test 21](#_Toc478989545)

# Anforderungen

Es soll das Spiel MasterMind als Applikation für das Android Betriebssystem entwickelt werden.

## Funktionale Anforderungen

* Einstellen des Schwierigkeitsgrades (5-6-8 Farben, 3-4-5-6-8 Löcher) und Anzahl der Züge pro Spiel (Standard=10), sowie Festlegung ob Farben mehrfach auftauchen dürfen und ob Stellen leer sein dürfen
* Aussehen des Spielbretts und der Pins soll geändert werden können (inkl. Festlegung der Spielsteinfarben)
* Auswahl des Spielmodus („Duellmodus“ oder „VS. CPU“)
* Erläuterung der Bedienung und Spielregeln über Hilfefunktion
* Spieler legt mit den farbigen Spielsteinen die Farbcodes fest und bestätigt Eingabe
* Anzeige von s/w-Stiften als Antwort auf die Benutzereingabe
* Spiel soll unterbrochen und später fortgesetzt werden können
* Spiel soll beendet werden sobald die richtige Kombination erreicht wurde (Popup-Meldung) oder die maximale Anzahl der Züge erreicht ist.
* Rücknahme eines Spielzuges (undo-Taste)
* Eintrag in eine Scoreliste (nur wenn undo-Taste nicht genutzt wurde)

## Nicht funktionale Anforderungen

* Spiel soll einfach bedient werden können
* Spiel soll flüssig und stabil laufen
* Modularer Aufbau um einzelne Funktionen flexibel austauschen zu können

## Anforderung an technische Realisierung

Das Spiel soll für das Android Betriebssystem entwickelt werden und ab Android Version 4.1.x funktionieren. Die eingesetzte Entwicklungsumgebung ist Android Studio und als Programmiersprache wird Java verwendet.

## Anforderung an den Projektablauf

Regelmäßige Meetings um den aktuellen Projektstand und weiteres Vorgehen zu besprechen

# Use Cases

## Use Case Diagramm



## Beschreibungen der Use Cases

### Use Case MM01 - Schwierigkeitsgrad festlegen

#### Kurzbeschreibung (Ziel):

Schwierigkeitsgrades soll eingestellt werden können.

Dabei soll die Anzahl der Farben (5-6-8), die Anzahl der Steckplätze auf dem Spielfeld (3-4-5-6-8) und die Anzahl der Züge pro Spiel (Standard=10) festgelegt werden können.

Des Weiteren soll eingestellt werden können ob Farben mehrfach im Farbcode auftauchen dürfen und ob Steckplätze bei der Auswertung leer gelassen werden können.

#### Akteure:

Spieler

#### Auslösendes Ereignis:

Aufruf der Spiel-Einstellungen in der App

#### Vorbedingung:

* App muss gestartet sein
* Nicht möglich solange noch ein aktives Spiel vorhanden ist (bzw. Einstellung wird erst beim Start eines neuen Spiels übernommen)

#### Nachbedingung (bei Erfolg):

* Anzahl Steckplätze auf dem Spielfeld müssen angepasst werden.
* Anzahl Farben in der Farbauswahl müssen angepasst werden.
* Größe des Spielfelds muss angepasst werden.

#### Nachbedingung (bei Fehlerfall):

* Fehlermeldung, wenn noch ein aktiver Spielstand existiert
* Fehlermeldung bei unerfüllbarer Konfiguration

#### Standardablauf:

1. Auswahl der Einstellungen
2. Übernehmen der Einstellungen
3. Starten des Spiels

### Use Case MM02 - Aussehen des Spielbretts anpassen

#### Kurzbeschreibung (Ziel):

Das Aussehen des Spielbretts (Hintergrundfarbe), sowie das Aussehen der Pins (Form und Farbe) soll in den Einstellungen angepasst werden können.

#### Akteure:

Spieler

#### Auslösendes Ereignis:

Aufruf der Spiel-Einstellungen in der App

#### Vorbedingung:

* App muss gestartet sein
* Nicht möglich solange noch ein aktives Spiel vorhanden ist (bzw. Einstellung wird erst beim Start eines neuen Spiels übernommen)

#### Nachbedingung (bei Erfolg):

* Darstellung des Spielbretts und der Pins müssen angepasst werden

#### Nachbedingung (bei Fehlerfall):

* Fehlermeldung, wenn noch ein aktiver Spielstand existiert
* Fehlermeldung bei unerfüllbarer Konfiguration

#### Standardablauf:

1. Auswahl der Einstellungen
2. Übernehmen der Einstellungen
3. Starten des Spiels

### Use Case MM03 – Spielmodus auswählen

#### Kurzbeschreibung (Ziel):

Es soll zwischen zwei Spielmodi ausgewählt werden können.

**Duellmodus**: Der Farbkode wird durch einen Spieler festgelegt und ein anderer Spieler spielt das Spiel

**Vs. CPU**: Der Farbkode wird beim Starten des Spiels zufällig generiert. Der Spieler spielt gegen den Computer

#### Akteure:

Spieler

Gegenspieler

#### Auslösendes Ereignis:

Starten eines neuen Spiels

#### Vorbedingung:

* App muss gestartet sein
* Start eines neuen Spiels

#### Nachbedingung (bei Erfolg):

* Code muss generiert und das Spiel gestartet werden.
* Code muss von Spieler festgelegt werden und das Spiel gestartet werden

#### Nachbedingung (bei Fehlerfall):

* Fehlermeldung, wenn ein ungültiger Kode eingegeben wurde

#### Standardablauf:

1. Auswahl Spielmodus
2. Vergabe des Kodes
3. Starten des Spiels

### Use Case MM04 – Hilfefunktion in Anspruch nehmen

#### Kurzbeschreibung (Ziel):

Anzeigen der Bedienung und der Spielregeln

#### Akteure:

Spieler

#### Auslösendes Ereignis:

Aufruf der Hilfefunktion

#### Vorbedingung:

* App muss gestartet sein

#### Nachbedingung (bei Erfolg):

* Spielregeln und hinweise zur Bedienung werden angezeigt

#### Nachbedingung (bei Fehlerfall):

* -

#### Standardablauf:

1. Aufruf Hilfefunktion
2. Anzeigen Hilfetext
3. Rückkehr zum vorherigen Fenster

### Use Case MM05 - Farbcodes während des Spiels festlegen

#### Kurzbeschreibung (Ziel):

Im Spiel gibt der Spieler mit Hilfe der Spielsteine einen Farbcode ein. Diesen bestätigt er anschließend.

#### Akteure:

Spieler

#### Auslösendes Ereignis:

Spielereingabe

#### Vorbedingung:

* App muss gestartet sein
* Spiel muss gestartet sein

#### Nachbedingung (bei Erfolg):

* Eingegebener Farbcode muss gegen vorhandenen abgeglichen werden oder der Farbcode wird als Musterfarbkode verwendet.

#### Nachbedingung (bei Fehlerfall):

* Ausgabe Fehlermeldung

#### Standardablauf:

1. Eingabe Farbcode durch Spieler
2. Bestätigung der Eingabe durch Spieler
3. Prüfen des Farbkodes / festlegen des Farbkodes als Musterkode

### Use Case MM06 – Ausgabe S/W Pins

#### Kurzbeschreibung (Ziel):

Nach Eingabe eines Farbkodes wird dieser mit dem vorhandenen Musterkode abgeglichen. Farben an der richtigen Position mit der Richtigen Farbe werden dem Spieler mittels schwarzer Pins am Spielfeldrand angezeigt. Pins mit der richtigen Farbe an falscher Position werden als weiße Pins angezeigt. Falsche Farben führen zu keiner anzeige. Die Position der Pins am Spielfeldrand steht in keiner Relation zu den Positionen der Farben im Kode

#### Akteure:

CPU

#### Auslösendes Ereignis:

Bestätigung des eingegebenen Farbcodes durch Spieler

#### Vorbedingung:

* App muss gestartet sein
* Spiel muss gestartet sein
* Musterfarbkode vorhanden sein
* Farbcode muss von Spieler eingegeben und bestätigt worden sein

#### Nachbedingung (bei Erfolg):

* Ausgabe der schwarzen und weißen Pins

#### Nachbedingung (bei Fehlerfall):

* Ausgabe Fehlermeldung

#### Standardablauf:

1. Bestätigung des Farbkodes durch Spieler
2. Prüfen der Eingabe gegen den Mustercode
3. Ausgabe s/w Pins

### Use Case MM07 – Spiel unterbrechen

#### Kurzbeschreibung (Ziel):

Der Spieler unterbricht das sein Spiel, dieses will er zu einem späteren Zeitpunkt fortsetzen.

#### Akteure:

Spieler

#### Auslösendes Ereignis:

Abbrechen eines Spieles

#### Vorbedingung:

* Spiel muss gestartet sein
* Musterfarbcode vorhanden sein
* Spiel muss gesichert sein.

#### Nachbedingung (bei Erfolg):

* Fortsetzung bei gesichertem Spielstand

#### Nachbedingung (bei Fehlerfall):

* Ausgabe Fehlermeldung

#### Standardablauf:

1. App Start
2. Spielstart (fortsetzen)
3. Laden des Spielstandes
4. Start des Spieles (ab Spielstand)

### Use Case MM08 - Beenden des Spiels bei richtigem Code

#### Kurzbeschreibung (Ziel):

Das Spiel wird nach Eingabe eines richtigen Farbkodes beendet

#### Akteure:

CPU

#### Auslösendes Ereignis:

Eingabe eines Farbkodes

#### Vorbedingung:

* App gestartet
* Spiel muss gestartet sein
* Musterfarbkode vorhanden sein
* Eingabe eines Farbkodes

#### Nachbedingung (bei Erfolg):

* Ausgabe Erfolgsmeldung
* Beendigung des Spiels

#### Nachbedingung (bei Fehlerfall):

* Ausgabe Fehlermeldung

#### Standardablauf:

1. Eingabe Farbcode
2. Prüfung Farbcode
3. Beendigung Spiels (Erfolg)
4. Ausgabe Erfolgsmeldung

### Use Case MM09 - Beenden des Spiels nach x Versuchen

#### Kurzbeschreibung (Ziel):

Das Spiel wird nach Eingabe von X (durch Nutzer festgelegten) Versuchen beendet

#### Akteure:

CPU

#### Auslösendes Ereignis:

Wiederholte Eingabe eines Farbcodes

#### Vorbedingung:

* App gestartet
* Spiel muss gestartet sein
* Musterfarbcode vorhanden sein
* Eingabe eines Farbkodes

#### Nachbedingung (bei Erfolg):

* Ausgabe Misserfolgsmeldung
* Beendigung des Spiels

#### Nachbedingung (bei Fehlerfall):

* Ausgabe Fehlermeldung

#### Standardablauf:

1. Eingabe Farbcode
2. Prüfung Farbcode
3. Prüfung Anzahl Versuche mit Vorgabe
4. Beendigung Spiels (Misserfolg)
5. Ausgabe Misserfolgsmeldung

### Use Case MM10 – Zug rückgängig machen

#### Kurzbeschreibung (Ziel):

Ein Spielzug (Eingabe Farbcode und bestätigen) kann über einen Button rückgängig gemacht werden.

#### Akteure:

Spieler

#### Auslösendes Ereignis:

Betätigung UNDO Button

#### Vorbedingung:

* App gestartet
* Spiel muss gestartet sein
* Musterfarbkode vorhanden sein
* Eingabe eines Farbkodes

#### Nachbedingung (bei Erfolg):

* Zurücksetzten des vorhergehenden Spielzugs

#### Nachbedingung (bei Fehlerfall):

* Ausgabe Fehlermeldung

#### Standardablauf:

1. Betätigung UNDO
2. Rücksetzten des Vorhergehenden Spielzugs
3. Rückkehr zu vorhergehendem Spielzug

### Use Case MM11 – Scoreboard anzeigen

#### Kurzbeschreibung (Ziel):

Nach erfolgreicher Beendigung eines Spiels wird der Spieler in eine Bestenliste eingetragen. Dies geschieht nur wenn er während des Spiels kein undo verwendet hat. In die Bestenliste werden Daten des Spiels sowie ein Spielername eingetragen. Der Spieler vergibt den Namen immer vor dem Eintrag.

#### Akteure:

CPU

Spieler

#### Auslösendes Ereignis:

Erfolgreiches beendigen eines Spiels

#### Vorbedingung:

* App gestartet
* Spiel muss gestartet sein
* Erfolgreiches Beendigen (ohne undo)

#### Nachbedingung (bei Erfolg):

* Eintrag Spielername
* Eintrag in Bestenliste

#### Nachbedingung (bei Fehlerfall):

* Ausgabe Fehlermeldung

#### Standardablauf:

1. Prüfung des Spiels (ob undo verwendet wurde)
2. Eingabe Spielername
3. Eintrag in Bestenliste

# Priorisierung

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Use Case Nr. | Titel | Priorisierung | | |
| hoch | mittel | niedrig |
| MM01 | Schwierigkeitsgrad festlegen |  | x |  |
| MM02 | Aussehen des Spielbretts festlegen |  |  | x |
| MM03 | Auswahl Spielmodus |  | x |  |
| MM04 | Hilfefunktion |  |  | x |
| MM05 | Festlegung der Farbcodes während des Spiels | x |  |  |
| MM06 | Ausgabe der S/W-Pins | x |  |  |
| MM07 | Spielunterbrechung |  |  | x |
| MM08 | Beenden des Spiels bei richtigem Code | x |  |  |
| MM09 | Beenden des Spiels nach x Versuchen | x |  |  |
| MM10 | Undo |  |  | x |
| MM11 | Scoreboard |  |  | x |

# Test Plan

## Unit-Tests

#### Ziel:

Das Sicherstellen von funktionierenden Funktionen. (Programmcode)

#### Zuständigkeit:

|  |  |
| --- | --- |
| Koordination und Dokumentation | Ralph Maier |
| Durchführung | Programierer der Funktion |
| Abnahme | Programmierer und Ralph Maier |

#### Zeitraum:

Eine Funktion wird nach Fertigstellung getestet.

#### Testmethode:

**Whitebox Testing** - Es werden die wahrscheinlichsten Abläufe der Funktion getestet.

#### Aufwandsschätzung:

Zwischen 10-40 Minuten (Abhängig von der Komplexität der Funktion)

#### Testumgebung:

Emulator Android Studio oder Android Endgerät.

#### Abschlusskriterium:

Erfüllen der definierten Funktionen.

## Anwendungstests

#### Ziel:

Das Sicherstellen der Funktionen der kompletten Applikation.

#### Zuständigkeit:

|  |  |
| --- | --- |
| Koordination und Dokumentation | Ralph Maier |
| Durchführung | Ralph Maier |
| Abnahme | Ralph Maier |

#### Zeitraum:

Die Funktion der Applikation wird mit jedem Prototyp getestet.

#### Testmethode:

**Blackbox Testing** - Funktionen der Use Cases werden überprüft. Sämtliche Funktionen und Eingabemöglichkeiten der App werden geprüft.

#### Aufwandsschätzung:

Bis zu 7 Stunden

#### Testumgebung:

Emulator Android Studio oder Android Endgerät.

#### Abschlusskriterium:

Erfüllen der Use Cases und fehlerfreies durchlaufen der alternativen Abläufe.

## Performance-Tests

#### Ziel:

Das Beheben von langen Laufzeiten und Performance Problemen.

#### Zuständigkeit:

|  |  |
| --- | --- |
| Koordination und Dokumentation | Ralph Maier |
| Durchführung | Ralph Maier |
| Abnahme | Ralph Maier |

#### Zeitraum:

Die Performance und Laufzeiten werden mit jedem Prototyp getestet.

#### Testmethode:

**Blackbox Testing** - Laufzeit der Use Cases werden überprüft.

#### Aufwandsschätzung:

Ca. 1 Stunde

#### Testumgebung:

Emulator Android Studio oder Android Endgerät.

#### Abschlusskriterium:

Erfüllen der Vorgabe von Laufzeiten. Kein Aufkommen von Performance Problemen.

## Recovery-Tests

#### Ziel:

Das Verhalten der Applikation nach einem Absturz oder beenden festzustellen.

#### Zuständigkeit:

|  |  |
| --- | --- |
| Koordination und Dokumentation | Ralph Maier |
| Durchführung | Ralph Maier |
| Abnahme | Ralph Maier |

#### Zeitraum:

Diese Funktion der Applikation wird mit jedem Prototyp getestet.

#### Testmethode:

**Blackbox Testing** - Funktionen der Use Cases werden nach Absturz geprüft überprüft.

#### Aufwandsschätzung:

Ca. 30 Minuten

#### Testumgebung:

Emulator Android Studio oder Android Endgerät.

#### Abschlusskriterium:

Erfüllen der Use Cases nach einem Absturz.

## Inbetriebnahme-Test

#### Ziel:

Sicherstellung der Funktionsfähigkeit der wichtigsten Schnittstellenanbindungen.

#### Zuständigkeit:

|  |  |
| --- | --- |
| Koordination und Dokumentation | Ralph Maier |
| Durchführung | Ralph Maier |
| Abnahme | Ralph Maier |

#### Zeitraum:

Diese Funktion der Applikation wird vor jedem Prototyp getestet.

#### Testmethode:

**Whitebox Testing** - Überprüfung der von allen Schnittstellen miteinander ausgetauschten Daten.

#### Aufwandsschätzung:

Ca. 4 Stunden

#### Testumgebung:

Emulator Android Studio oder Android Endgerät.

#### Abschlusskriterium:

Redundanzfreie und fehlerfreie Daten aus allen Schnittstellen.