Cookie Clicker 2.0

Anforderungsanalyse

|  |  |
| --- | --- |
| **Autor(en):** | Leo Scherer |
| **Erstellt am:** | 5. Dezember 2018 |
| **Speicherort:** | Github / Lenoxy |

Inhalt

[1 Revisionen 3](#_Toc531702604)

[2 Einleitung 4](#_Toc531702605)

[2.1 Systemidee 4](#_Toc531702606)

[2.2 Management Summary 4](#_Toc531702607)

[2.3 Projektteam 4](#_Toc531702608)

[2.4 Glossar 4](#_Toc531702609)

[3 Ausgangslage (IST), Problembereiche 4](#_Toc531702610)

[3.1 Beschreibung der Ausgangslage 4](#_Toc531702611)

[3.2 Problembereiche und Schwachstellen 4](#_Toc531702612)

[4 Ziele (SOLL) 4](#_Toc531702613)

[4.1 Beschreibung der Ziele 4](#_Toc531702614)

[4.2 Produktperspektive, Nutzen 4](#_Toc531702615)

[4.3 Zielkonflikte 4](#_Toc531702616)

[4.4 Abgrenzung 4](#_Toc531702617)

[5 Anforderungsanalyse 4](#_Toc531702618)

[5.1 Identifizierung der Akteure 4](#_Toc531702619)

[5.2 Anforderungskatalog 4](#_Toc531702620)

[5.2.1 F.REQ: Funktionale Anforderungen 4](#_Toc531702621)

[5.2.2 NF.REQ: Nichtfunktionale Anforderungen 4](#_Toc531702622)

[6 Systemablaufmodelle (Aktivitäten) 4](#_Toc531702623)

[6.1 Aktivität "DVD ausleihen" 4](#_Toc531702624)

[6.2 Aktivität «XY» 4](#_Toc531702625)

[7 Risiko-Analyse 4](#_Toc531702626)

[7.1 Risikokatalog 4](#_Toc531702627)

[8 Anhang 4](#_Toc531702628)

[8.1 Termine 4](#_Toc531702629)

[8.2 Referenzen 4](#_Toc531702630)

# Revisionen

|  |  |  |
| --- | --- | --- |
| **Datum** | **Name** | **Kommentar** |
| 05.12.2018 | Leo Scherer | Dokument erstellt. |
|  |  |  |
|  |  |  |
|  |  |  |

# Einleitung

## Systemidee

Das System soll nach dem Modell des relativ bekannten Cookie Clicker funktionieren. Also ein Knopf, das einem Klick die Anzahl Cookies erhöht, die auch auf dem Programm erkennbar sind. Eine gewisse Anzahl Cookies kann dann jeweils für ein Upgrade eingetauscht werden, dass dem Spieler automatisch eine gewisse Anzahl Cookies zuschreibt.

### Die wichtigsten Funktionen hierzu sind:

|  |  |
| --- | --- |
| Funktion 1 | Bei Programmstart soll ein GUI geöffnet werden |
| Funktion 2 | Es soll auf einen Knopf gedrückt werden können, der die Anzahl Cookies erhöht. |
| Funktion 3 | Upgrades sollen gekauft werden können (Abzug der Cookies) |
| Funktion 4 | Die Upgrades sollen automatisch Cookies hinzufügen |
| Funktion 5 | Es soll anzeigt werden, wieviel Cookies pro Sekunde verdient werden. |
| Funktion 6 | Der Spielstand soll als Konfigurationsdatei abgespeichert und geladen werden sollen. |
| Zusatzfunktion (Falls noch Zeit bleibt) | Es soll eine Leaderboard-Funktion geben, die auf der BLJ-Webseite von Noah per SQL angezeigt wird. Dafür würde **eventuell** ein gemeinsames Account-System mit Noah, Philip und Nick in Frage kommen. |

## Management Summary

Mein Projekt soll dem bekannten Spiel Cookie Clicker ähnlich sein. Durch drücken auf das Cookie, wird der Cookie-Zähler um eins Erhöht. Mit Anzahl Cookies können sich Upgrades gekauft werden, die automatisch Cookies sammeln. Zusätzlich wird auch die Anzahl verdienter Cookies pro Sekunde angezeigt und das Spiel soll aus Konfigurationsdateien lesen können.

## Projektteam

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Kürzel** | **Funktion** | **Kontakt (Telefon, Mail...)** |
| Urs Nussbaumer | nuu | Projektleiter | 041 371 24 28 urs.nussbaumer@ict-bz.ch |
| Leo Scherer | scl | Entwickler | 079 630 60 62 lenoxy@gmx.ch |

## Glossar

Erläutern Sie alle im Projekt verwendeten Fachbegriffe und Abkürzungen in alphabetischer Reihenfolge.

|  |  |
| --- | --- |
| **Begriff** | **Beschreibung** |
|  |  |
|  |  |
|  |  |
|  |  |

# Ausgangslage (IST), Problembereiche

In diesem Kapitel wird die Ausgangslage beschrieben und es werden die Problembereiche identifiziert, die sich in der heutigen Situation zeigen.

## Beschreibung der Ausgangslage

Eine ähnliche Form des Spiels existiert bereits: <http://orteil.dashnet.org/cookieclicker/>

Ich werde meine Version dieses Spieles jedoch selber in Java schreiben.

## Problembereiche und Schwachstellen

Die Version läuft im Internet. Ich werde mein Spiel aber als Desktop-Applikation entwerfen. Noch eine Schwachstelle des aktuellen Cookie-Clicker ist, dass sie nur in einen Konfigurationsfile speichern können. Meine Version wird diese Option bieten, aber zusätzlich eine automatische Speicherung des aktuellen Standes, wenn das Spiel geschlossen wird. Beim Start soll dann nach Speicherständen überprüft werden.

# Ziele (SOLL)

In diesem Kapitel werden die übergeordneten Ziele beschrieben, die mit dem zu entwickelnden System erreicht werden sollen.

## Beschreibung der Ziele

Der Spieler soll durch ein lustiges und spannendes Spiel unterhalten werden.

## Produktperspektive, Nutzen

Durch die Veröffentlichung könnte Werbung platziert werden, und somit kann Geld gemacht werden. Das Spiel könnte auch für Geld verkauft werden, was aber wahrscheinlich nicht erfolgreich funktionieren wird, da die Konkurrenz Ihre Produkte gratis anbieten.

# Anforderungsanalyse

Die Anforderungen an das zu entwickelnde System definieren alle zu erfüllenden Eigenschaften oder die zu erbringende Leistung, sowie allfällige technische Vorgaben und weitere Forderungen des Kunden im Zusammenhang mit dem (den) zu erstellenden Produkt(en).

## Identifizierung der Akteure

Mein Spiel soll nur eine Benutzergruppe haben, nämlich der **Spieler**. Dieser soll das Spiel spielen können und sich in das Leaderboard eintragen können. (Das Leaderboard soll vom Administrator direkt in phpMyAdmin verwaltet werden können.)



## Anforderungskatalog

Der Anforderungskatalog ist eine priorisierte Liste, die alles enthält, was im zu entwickelnden Produkt enthalten sein soll. Es wird unterschieden zwischen funktionalen und nicht-funktionalen Anforderungen.

Jede Anforderung ist in Alltagssprache, in der Form einer "User-Story", formuliert, mit einer ID zur späteren Identifizierung versehen sowie mittels 3 verschiedener Kategorien priorisiert: 1 = hohe Prorität, 2 = mittlere Priorität, 3 = keine Priorität. Diese drei Prioritäten repräsentieren die Verpflichtungen "must", "should" und "nice to have".

Im Verlauf der Zeit können neue Anforderungen hinzukommen und/oder bestehende Anforderungen können wegfallen. Anforderungen, die wegfallen, sind im Dokument zu belassen und als ~~gestrichen~~ zu markieren.

Hinter einer Anforderung kann ihn eckigen Klammern in der Form [Z#] eine Zusatzinformation hinterlegt werden und mit [F#] bzw. [F#, F#, …] können eine oder mehrere offene Fragen referenziert werden.

### F.REQ: Funktionale Anforderungen

|  |  |  |
| --- | --- | --- |
| **ID** | **Anforderung** | **Priorität** |
| F.REQ.001 | Als User will ich einen Knopf drücken können. Diese werden dann gezählt. | 1 |
| F.REQ.002 | Als User will ich Upgrades kaufen können, die mir automatisch Cookies verschaffen. | 1 |
| F.REQ.003 | Als User will ich sehen wie viele Cookies pro Sekunde meine Upgrades verdienen. | 2 |
| F.REQ.004 | Als User will ich beim Programmstart meinen alten Spielstand wieder geöffnet haben. | 2 |
| F.REQ.005 | Als User will ich die Konfigurationsdatei auch manuell abspeichern können. | 2 |
| F.REQ.006 | Als User will ich mich ins Leaderboard eintragen können, dass auf der BLJ-Webseite angezeigt wird. | 3 |
| F.REQ.007 | Als User will ich mich mit meinem BLJ-Login anmelden können, unter dem mein Score, sowie meine Konfigurationsdatei gespeichert wird. | 3 |

#### Offene Fragen

[F1]

#### Zusatzinformationen

[Z1]

### NF.REQ: Nichtfunktionale Anforderungen

|  |  |  |
| --- | --- | --- |
| **ID** | **Anforderung** | **Priorität** |
| NF.REQ.001 | Ich will bei diesem Projekt die Clean Code Konventionen besonders gut einhalten, um die Erweiterbarkeit zu Garantieren. | 1 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Systemablaufmodelle (Aktivitäten)

Dieses Kapitel zeigt die wichtigsten und/oder komplexesten funktionalen Anforderungen in ihrem Ablauf. Dazu werden die einzelnen Aktivitätsschritte detailliert analysiert. Für die Darstellung der einzelnen Aktivitätsschritte werden Aktivitätsdiagramme nach UML verwendet.

# Risiko-Analyse

Ziel der Risiko-Analyse ist es, die Kontrolle zu haben über die wesentlichen Projektrisiken während des Projektverlaufs sowie die Identifizierung der grössten respektive der schwerwiegendsten Risiken, welche dann bei der Implementation als erstes angegangen/getestet werden (Greatest Risk First).

## Risikokatalog

Legende:

* **w** = Wahrscheinlichkeit des Auftretens: 1 = null, 2 = klein, 4 = eher klein 6=mittel 8=hoch, 10 = sehr hoch, >50%
* **b** = Bedeutung der Auswirkungen: 1 keine, 3 minim, 6 signifikant, 8 schwer, 10 fatal, Abbruch
* **p** = Prävention, Behebung, Vermeidung: 1 gratis, 3 leicht, 4 gut 90%, 6 mittel, 8 schlecht,10 unvermeidbar

Das grösste Risiko steht am Anfang des Katalogs, das geringste am Ende.

|  |  |  |  |
| --- | --- | --- | --- |
| **Nr: Risiko** | R.1: | **w \* b \* p** |  |
| Ursache |  | | |
| Folgen |  | | |
| Massnahmen |  | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Nr: Risiko** | R.2: | **w \* b \* p** |  |
| Ursache |  | | |
| Folgen |  | | |
| Massnahmen |  | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Nr: Risiko** | R.3: | **w \* b \* p** |  |
| Ursache |  | | |
| Folgen |  | | |
| Massnahmen |  | | |

# Anhang

## Termine

|  |  |  |
| --- | --- | --- |
| **Datum** | **Ziel, Beschreibung, Meilenstein** | **Verantwortlich** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## Referenzen

todo: falls es andere Dokumente gibt, die in Zusammenhang mit dem Projekt stehen, diese hier notieren.

|  |  |  |  |
| --- | --- | --- | --- |
| **Nr.** | **Titel/Beschreibung** | **Autor** | **Dokument oder URL** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |