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

In diesem Spiel kannst du fast unendlich viele Cookies backen! Mein Spiel nimmt das bekannte Modell des Webspiels Cookie Clicker und fügt viele nützliche Features wie automatisches Speichern hinzu.

### Die wichtigsten Funktionen hierzu sind:

|  |  |
| --- | --- |
| Funktion 1 | Auf das Cookie drücken und du hast nun ein Cookie mehr als zuvor! |
| Funktion 2 | Du kannst mit deinen Cookies Upgrades erwerben und somit mehr Cookies ganz von alleine produzieren! |
| Funktion 3 | Du kannst einfach anzeigen lassen, wie viele Cookies du pro Sekunde verdienst. |
| Funktion 4 | Das Spiel speichert dein Spielstand automatisch und du kannst Ihn sogar mit deinen Freunden Teilen. |
| Funktion 5 | Eine Rangliste mit den meisten Cookies befindet sich auf der Basislehrjahrwebseite. |

## Management Summary

Mein Projekt soll dem bekannten Spiel Cookie Clicker ähnlich sein. Durch drücken auf das Cookie, erhöht sich deine gesamte Anzahl Cookies. Mit Cookies können sich Upgrades gekauft werden, die dir automatisch Cookies sammeln, ohne dass du nur einen Finger rührst! Zusätzlich wird auch die Anzahl verdienter Cookies pro Sekunde angezeigt und das Spiel kann auch Spielstände importieren, beziehungsweise exportieren, falls dir die automatische Speicherfunktion nicht ausreicht.

## 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 |

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

# 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 bekannte Version läuft im Internet. Ich werde mein Spiel aber als Desktop-Applikation entwerfen. Eine Schwachstelle des aktuellen Cookie-Clicker ist, dass keine automatische Speicherung vorhanden ist. Meine Version wird die Option bieten, manuell Spielstände zu im- und exportieren, aber zusätzlich eine automatische Speicherung des aktuellen Standes, wenn das Spiel geschlossen wird. Beim Start wird dann automatisch der letzte Spielstand geladen.

# 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

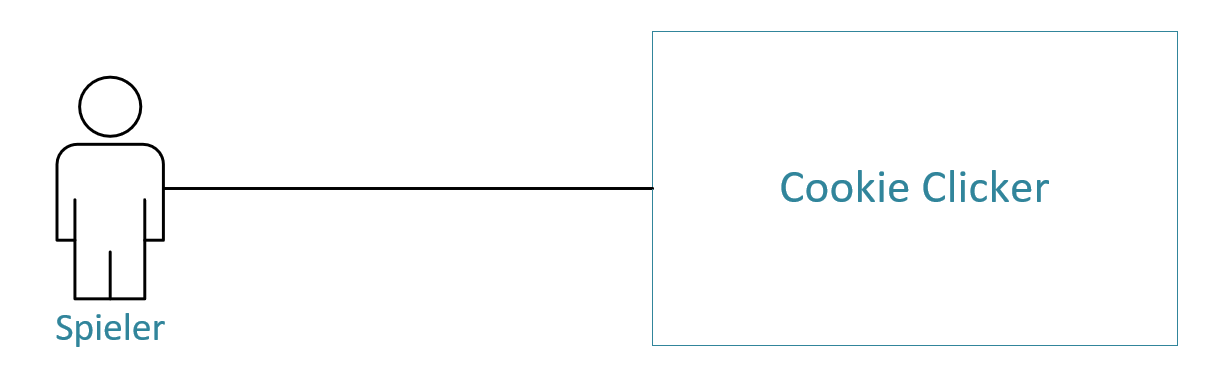
Durch die Veröffentlichung könnte Werbung platziert werden, und somit kann Geld gemacht werden. Das Spiel könnte auch für Geld verkauft werden, was aber wahrscheinlich nicht erfolgreich funktionieren wird, da die Konkurrenz Ihre Produkte gratis anbieten.

# 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 Spieler will ich ein Cookie drücken können. Die Anzahl Cookies werden dann von einem Counter gezählt. | 1 |
| F.REQ.002 [Z1] | Als Spieler will ich Upgrades kaufen können, die mir automatisch Cookies backen. | 1 |
| F.REQ.003 | Als Spieler will ich sehen wie viele Cookies alle meine Upgrades zusammen pro Sekunde backen. | 2 |
| F.REQ.004 | Als Spieler will ich beim Programmstart meinen alten Spielstand wieder geöffnet haben. | 2 |
| F.REQ.005 | Als Spieler will ich meinen Spielstand importieren und exportieren können. | 2 |
| F.REQ.006 | Als Spieler will ich mich in eine Rangliste eintragen, die die besten Spieler Weltweit nach dem besten Sortiert. | 3 |
| F.REQ.007 | Als Spieler möchte ich diese Rangliste auf der Basislehrjahr-Webseite anzeigen können. | 3 |
| F.REQ.008 | Als Spieler will ich mich mit meinem BLJ-Login anmelden können, unter dem mein Score, sowie meine Konfigurationsdatei gespeichert wird, damit ich mich an einem beliebigen Computer anmelden kann. | 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 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |