<Name der Software>

Anforderungsspezifikation

**FHNW**

**Hochschule für Technik**

**Mini-Projekt Requirements Engineering  
«Produktinnovation»**

**2015 / 2016**

**Über das Template v1.0-2015**

Das Template wurde für den Requirements Engineering-Unterricht an der FHNW entwickelt. Die Nutzung des Dokuments wird kontinuierlich überwacht und, falls notwendig, Anpassungen gemacht. Feedback und Empfehlungen nimmt [samuel.fricker@fhnw.ch](mailto:samuel.fricker@fhnw.ch) gerne entgegen.

Im Gegensatz zu den Studierendenprojekteen wird im Requirements Engineering Mini-Projekt eine Innovation mit einer Software angestrebt, welche potentiell durch die Studierenden selbst realisiert werden kann. Das vorliegende Dokument soll als Werkzeug genutzt werden, un eine möglichst einfache Software zu definieren, welches möglichst grosse Effekte generiert.

In jedem Kapitel haben Sie Instruktionen, was zu dokumentieren ist und wie evtl. die Unterkapitelstruktur angepasst werden soll. Ersetzen Sie den Template-Text durch Ihre eigenen Texte.

Denken Sie im Füllen der Kapitelinhalte an den Leser. Folgende Kapitelstruktur funktioniert oft gut:

* Kurze Einführung, was im Kapitel beschrieben ist und in welchem Bezug es zu vorhergehenden Kapiteln steht.
* Kurze Zusammenfassung der Kapitelinhalte.
* Details in Bildern, Tabellen und Texten.

Viel Glück und viel Spass am Entdecken, was mit Ihrer Software alles gemacht werden kann!

/Samuel Fricker

**Revisionen**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Datum** | **Kommentar** | **Autor** |
| Version 0.0 | 05.10.2015 | Vorlage | Samuel Fricker |
|  |  |  |  |

**Inhalt**

[1 Problemstellung 5](#_Toc431794748)

[1.1 Hintergrund 5](#_Toc431794749)

[1.2 Ziele einer erfolgreichen Lösung 5](#_Toc431794750)

[1.3 Bewertung von existierenden Alternativen 5](#_Toc431794751)

[2 Konzept: «Name der Software» 6](#_Toc431794752)

[2.1 Grundidee 6](#_Toc431794753)

[2.2 Gesichtspunkte 6](#_Toc431794754)

[2.3 Features 6](#_Toc431794755)

[2.4 Einschränkungen 7](#_Toc431794756)

[3 Anforderungen 8](#_Toc431794757)

[Nutzungsszenarien 8](#_Toc431794758)

[Detaillierte Anforderungen 8](#_Toc431794759)

[4 Anhänge 9](#_Toc431794760)

[4.1 Zukünftige Anforderungen 9](#_Toc431794761)

[4.2 Alles Andere 9](#_Toc431794762)

# Problemstellung

## Hintergrund

Die Problemstellung, welche möglichen Nutzer und Kunden motivieren wird, das in den kommenden Kapiteln spezifizierte Software zu nutzen.

Formalisieren Sie mit den folgenden Elementen des RUP-Visionstemplate:

* The problem of…
* Affects…
* The impact of which is…

Beschreiben Sie Ihre Evidenz, die zeigt, dass das Problem wirklich existiert und von den erwähnten Stakeholders getragen wird.

## Ziele einer erfolgreichen Lösung

Kriterien zum Entscheiden, ob eine Lösung des Problems erfolgreich ist. Dieses Unterkapitel entspricht dem Eintrag «a successful solution would be…» im RUP Visionstemplate.

* Name des Ziels (so dass Sie später darauf verweisen können),
* Beschreibung (was das Ziel ist, warum das Ziel wichtig ist und wer neben dem Kunden evtl. auch noch davon betroffen ist),
* Messung der Zielerreichung (quantitative Kriterien wann das Ziel als erreicht betrachtet wird).

## Bewertung von existierenden Alternativen

Was die Nutzer und Kunden alternativ bisher genutzt haen oder nutzen könnten, um das Problem zu lösen, und warum diese Alternativen nicht gut genug sind. Für jede Alternative:

Shortcut (<http://www.getshortcut.co/>)

Shortcut ist eine App, die Zertifizierte Contractor Friseuere für einen Fixpreis von 75 US-Dollars anbietet. Shortcut ist nur für männliche Kunden gedacht.

Wir würden die Friseuere nicht als

Inserate

Asdfadsföafadsfdf

CoiffureSuisse

Asfdsafasf

Coiffeur Vergleich

Asdfsadf

Walk-In Salons

adsf

* Name der Alternative (so dass Sie später darauf verweisen können),
* Beschreibung (was die Alternative ist, warum die Alternative nicht gut genug für das Erreichen der Ziele einer erfolgreichen Lösung ist)
* Bedeutung für Ihre Software (ob Sie Teile der Alternative übertragen oder nutzen können und was Ihre Software anders machen muss)

# Konzept: «Name der Software»

## Grundidee

Was die Grundidee Ihres Software ist, wer es nutzen wird und wie durch die Nutzung die Ziele einer erfolgreichen Lösung erreicht werden.

Fassen Sie das Konzept zusammen und formalisieren Sie es mit den folgenden Elementen des RUP-Visionstemplates:

* For…
* Who… v
* The…
* That…
* Unlike…
* Our solution…

Beschreiben Sie Ihre Evidenz, die zeigt, dass Ihr Konzept funktioniert und die Ziele einer erfolgreichen Lösung erfüllt.

## Gesichtspunkte

Betrachten Sie folgende Gesichtspunkte:

* Die verschiedenen Nutzergruppen
* Der Kunde
* Das Rechenzentrum, der Appstore, oder andere Wege des Vertriebs und Betriebs
* Drittsysteme mit denen Daten ausgetauscht werden müssen
* Andere (Personen, Gruppen, oder Organisationen)

Definieren Sie für jeden Gesichtspunkt:

* Name des Gesichtspunkts
* Charakterisierung (Kriterien zum entscheiden, ob jemand dieser Rolle entspricht: z.B. ist es ein Oberstufenschüler der sich für Beruf X interessiert, ein Produzent von Y der mehr Kunden an seinen Verkaufsstand locken möchte, etc.)
* Hintergrund (z.B. welche Alternativen dieser Stakeholder typischerweise kennt)
* Ziele des Gesichtspunkts (typische von Ihrer Software unterstütze Bedürfnisse des Stakeholders, die ihn/sie motivieren, die Software zu verwenden oder unterstützen und warum diese für den Stakeholder wichtig sind)
* Ansprechsperson(en) die in das Requirements Engineering einbezogen sind

## Features

Die wichtigsten Merkmale (Features) der Software, welche für das Erreichen der Ziele notwendig sind. Ein guter Weg, Features zu erstellen, ist sich zu überlegen, welche Arten von wichtigen Daten von der Applikation verwaltet werden müssen. Sortieren Sie die Features gemäss ihrer Priorität für die Entwicklung (oben das zuerst zu implementierende Feature). Definieren Sie für jede solche Art ein Feature. Für jedes Feature:

* Name (so dass Sie später darauf verweisen können)
* Kurzbeschreibung (was die Idee des Features ist)
* Ziele (wie das Feature hilft, die Ziele einer erfolgriechen Lösung oder die Ziele von Gesichtspunkten zu erfüllen)

## Einschränkungen

Entscheide im Projektmanagement, Architektur, Implementierung oder Testing, welche Ihnen Gesichtspunkte vorweggenommen haben. Für jede Einschränkung:

* Kurzbeschreibung (die Einschränkung und warum sie für den Kunden wichtig ist)

# Anforderungen

Strukturieren Sie dieses evtl. lange Kapitel so, dass Zusammengehöriges in ein Unterkapitel gelegt wird. Falls möglich, erstellen Sie ein Unterkapitel pro Feature. Sortieren Sie die Unterkapitel gemäss ihrer Priorität für die Entwicklung (oben das zuerst zu implementierende Unterkapitel).

In diesem Kapitel wird erwartet, dass Sie sinnvolle Notationen selbständig auswählen. Machen Sie einen aktiven Transfer der Kursinhalte in dieses Dokument.

## Nutzungsszenarien

Beschreiben Sie, wie Ihre Software genutzt wird und erklären Sie, wie diese Nutzung die Ziele einer erfolgreichen Lösung und die relevanten Stakeholderziele erreicht. Sie können die Nutzung in verschiedene Szenarien oder Situationen aufteilen. Für jedes Szenario oder Situation:

* Name des Szenarios oer Situation (so dass Sie später darauf verweisen können)
* Beschreibung in einer Art Filmskript, das Sie für die Demos nutzen. Verwenden Sie die Use Case-Notation, falls Sie diese kennen.
* Kurze Erklärung, wie das Szenario hilft, die Ziele einer erfolgreichen Lösung zu erreichen.
* Kurze Erklärung, wie das Szenario hilft, die relevanten Stakeholderziele zu erreichen.

## Detaillierte Anforderungen

Erstellen Sie eine Liste von Anforderungen, für die der Projektmanager Arbeiten plannen kann. Anforderungen können sein:

* Daten und deren Verhalten
* Funktionen
* GUI (nur Name falls die Dokumentation des Usability Engineers auf die Anforderung Bezug nimmt)
* Qualitätseigenschaft (Performance, Sicherheit, Zuverlässigkeit, Benutzbarkeit, Änderbarkeit, Übertragbarkeit/Plattformunterstützung)

Für jede Anforderung:

* Einmaliger Identifikator der Anforderung (in den Usability-Dokumenten, im Testplan, etc. genutzt)
* Beschreibung der Anforderung (nutzen Sie eine User Story, falls Sie diese Notation kennen und sie hier relevant ist)
* Kommentare (Klärungen soweit notwendig)

# Anhänge

## Zukünftige Anforderungen

Hier dokumentieren Sie eine Liste der gewünschten Verbesserungen oder Veränderungen an der Software, welche nach Ihrem Projekt realisiert werden sollen.

## Alles Andere

Verweisen Sie hier auf Dokumente, Modelle, Photos, Videos, etc., welche zum Verstehen der Anforderungen oder als Evidenz zur Korrektheit der Anforderungen genutzt werden können.