Inhaltsverzeichnis

[1 Management Summary 3](#_Toc34901330)

[2 Zweck 4](#_Toc34901331)

[3 Bewertung 5](#_Toc34901332)

[4 Funktion 6](#_Toc34901333)

[5 Mockups 7](#_Toc34901334)

[6 Use Case 8](#_Toc34901335)

[7 Aktivitätendiagramme 9](#_Toc34901336)

[8 Systemtests 10](#_Toc34901337)

[9 Testprotokoll 11](#_Toc34901338)

[10 Installationsanleitung 12](#_Toc34901339)

[11 Reflexion 13](#_Toc34901340)

# Management Summary

Im ÜK M318 wurde das Projekt «Swiss Transport App» erstellt und entwickelt. Bei diesem Projekt sollte eine Applikation entwickelt werden welche HTTP Web APIs verwendet, um Daten zu holen. Diese Daten sollten mit Hilfe von C# und Visual Studio in ein GUI eingebunden werden.

# Zweck

Durch dieses Dokument wird das Programm «Swiss Transport App» beschrieben sowie die Entstehung erklärt. Dieses Dokument beinhaltet die Planung sowie die Umsetzung des Projektes und der Umgang mit der Web API. Dieses Projekt wurde von Simon Krieger im Rahmen des Kurses M318 entwickelt und fertiggestellt.

Der Zweck des Kurses war es, die Kenntnisse im Umgang mit C# und WPF/WinForms zu verbessern sowie die Kenntnisse im Objektorientierten Programmieren zu verbessern.

# Bewertung

Die Applikation musste mindestens die Aufgaben A001 – A003 erfüllen.

Das Projekt wird folgendermassen bewertet:

## Qualität

### Funktionalität

Hier wird geprüft, ob die Software richtig funktioniert.

### Zuverlässigkeit

Hier schaut man ob die Software zuverlässig funktioniert, was in Fehler-Situationen passiert, ob verständliche Fehlermeldungen angezeigt werden, ob sich Daten wiederherstellen lassen usw.

### Benutzbarkeit

Kann man die Software einfach bedienen? Ist die Benutzeroberfläche übersichtlich und selbsterklärend gestaltet? Versteht man die Funktionen?

### Effizienz

Definiert die Performance (das Zeitverhalten) der Applikation, den Speicherverbrauch usw.

### Wartbarkeit

Kann man die Software einfach/problemlos erweitern? Wie einfach ist es, Fehler einzugrenzen und zu lokalisieren? Wie stabil verhält sich die Software bei Änderungen (Stichwort: Side Effects)? Ist der Source Code gut und verständlich kommentiert? Wurden die Programmierrichtlinien durchgängig eingehalten?

### Portierbarkeit

Kann man die Software einfach installieren? Kann sie mit anderen Programmen koexistieren? Welche Auswirkungen hat die Installation der Software auf andere Programme? usw.

## Qualitätsziele

### Funktionalität

Die Funktionalität der Software soll den Anforderungen entsprechen, welche laut Dokumentation umgesetzt wurden.

Die Funktionen sollen korrekt umgesetzt sein.

Die richtige Funktionsweise soll durch Testfälle sichergestellt sein.

### Zuverlässigkeit

Die Software soll zuverlässig funktionieren und tun, was man von ihr erwartet.

Die Software soll in Fehlerfällen den Benutzer korrekt (mit verständlichen Fehlermeldungen) informieren, ohne abzustürzen.

Nach Auftreten eines Fehlers sollte die Software, wenn möglich, normal weiterlaufen.

### Benutzbarkeit

Die Bedienung der Software soll selbsterklärend sein.

Benutzereingaben sollen validiert werden.

Die grafische Benutzeroberfläche (GUI) soll übersichtlich gestaltet sein.

Das GUI soll sich an UI Standards halten.

### Wartbarkeit

Der Source Code soll sich an die Programmierrichtlinien (Coding Guidelines) halten.

Variablen, Klassen, Methoden und andere Elemente sollen sprechende Namen haben.

Der Source Code soll verständlich kommentiert sein, insbesondere sollen alle öffentlichen Methoden inklusive ihrer Parameter kommentiert sein.

Es soll einfach und problemlos möglich sein, Änderungen an der Software zu machen. Dazu gehört: kein repetitiver Code (copy paste), kein Spaghetticode, Beachtung des Kohäsionszprinzips und eine möglichst lose Kopplung der einzelnen Module und Klassen.

### Portierbarkeit

Die Software soll einfach zu installieren sein.

Die Software soll auch einfach wieder entfernt werden können.

Die Software soll nach der Installation auch auf einem Rechner ohne Entwicklungsumgebung lauffähig sein.

Zusammen mit der Software soll eine Installationsanleitung ausgeliefert werden.

## Dokumentation

* Autor, Datum.
* Eine Einleitung (Management Summary).
* Zweck des Dokuments.
* Was (d.h. welche Funktionen) wurde umgesetzt?
* Falls bestimmte Funktionen nur teilweise umgesetzt wurden: Welche? Welcher Teil der Funktionalität fehlt noch? Bekannte Fehler/Bugs?
* Mockups für alle Formulare/Dialoge (GUI) , die benötigt werden um die Anforderungen A001, A002 und A003 abzudecken.
* Use Cases, Use Case Beschreibungen und Aktivitätendiagramme mindestens für die mit Priorität 1 klassifizierten Anforderungen.
* Testfälle (Systemtests), verständlich und eindeutig nachvollziehbar geschrieben, so dass ein Tester diese ohne weiteren Erklärungen durchführen kann.
* Testprotokoll mit Durchführungsdatum der Tests, Name des Testers und Protokollierung der tatsächlichen Testresultate.
* Installationsanleitung: Wie wird die Software installiert? Wie wird die Software deinstalliert?
* Andere spannende Informationen für die Bewertung.

# Funktion

# Mockups

# Use Case

# Aktivitätendiagramme

# Systemtests

# Testprotokoll

# Installationsanleitung

# Reflexion