

## REQUIREMENTS MHC-PMS – COMMUNITY WORKER

Projekt **MHC-PMS**  
Autor/in **Team Black**  
Klassifizierung **intern**  
Ausgabedatum **20.03.13**  
Version **V01.00**

### Verteiler

- Team Black
- Projektkontrolle Jürgen Vogel und Urs Künzler

### Änderungskontrolle

Diese Seite zeigt den Änderungsstand dieses Dokumentes. Mit jeder Änderung erfolgt eine Neuausgabe.

Version	Überarbeitung	Ersteller	Datum
V01.00	Initialversion	Team Black Daniel Inversini	20.03.2013
V01.01	Erste Überarbeitung, Einfügen der Use Cases	Team Black Daniel Inversini	22.03.2013
V01.02	Erstellung alle Kapitel und Inhalte	Team Black	22.03.2013
V01.03	Zweite Überarbeitung	Team Black Marco Füllemann	22.03.2013

## Inhalt

1	Allgemein .....	3
1.1	Beschreibung MHC – PMS, Community Worker .....	3
1.2	Definitionen, Akronyme und Abkürzungen .....	3
1.2.1	Abkürzungen .....	3
1.2.2	Definitionen .....	3
2	Stakeholder .....	4
2.1	Zweck .....	4
2.2	Liste der Stakeholder .....	4
3	Auflistung der Benutzervoraussetzungen (User Requirements) .....	5
3.1	Funktionale Anforderungen .....	5
3.1.1	Planungsunterstützung (Apollo) (UC-1) .....	5
3.1.2	Verfügbarkeit von Klienten-Daten (UC-2) .....	5
3.1.3	Unterstützung bei der Benutzung des öffentlichen Verkehrs (UC-3) .....	5
3.1.4	Geo-Location Unterstützung (UC-4) .....	5
3.2	Nicht funktionale Anforderungen .....	5
3.2.1	Sicherheit .....	5
3.2.2	Kosten .....	5
3.2.3	Usability .....	5
3.2.4	Patienten-Info .....	6
3.2.5	Beständigkeit .....	6
4	Auflistung der Systemvoraussetzungen (System Requirements) .....	6
4.1	Funktionale Anforderungen .....	6
4.1.1	Planungsunterstützung (UC-1) .....	8
4.1.2	Verfügbarkeit von Klienten-Daten (UC-2) .....	8
4.1.3	Unterstütz bei der Benutzung des öffentlichen Verkehrs (UC-3) .....	9
4.1.4	Geo-Location Unterstützung (UC-4) .....	11
4.2	Nicht funktionale Anforderungen .....	12
4.2.1	Sicherheit .....	12
4.2.2	Kosten .....	12
4.2.3	Usability .....	13
4.2.4	Patienten-Info .....	13
4.2.5	Beständigkeit .....	13
5	Testing .....	14
6	System Architektur .....	15
7	Anhang .....	16
8	Abbildungsverzeichnis .....	16

# 1 Allgemein

Dieses Dokument enthält Requirements (Anforderungen) an das Mental Health Care – Patient Management System, kurz MHC-PMS, eine Software. In diesem Dokument werden Funktionale – und nicht Funktionale Anforderungen behandelt. Geschäftliche, finanzielle Anforderungen werden nicht behandelt.

## 1.1 Beschreibung MHC – PMS, Community Worker

Dieser Teil der Software MHC – PMS dient dazu, all die Prozesse und Bedürfnisse des Community Workers abzudecken. Die aktuellen Prozesse werden z.Z. manuell erledigt, mit einem grossen Mehraufwand und Papierverbrauch. Das Ziel hier ist, den Community Worker bei den alltäglichen Anforderungen zu unterstützen oder gar entlasten.

Die Bedürfnisse wurden in einem Schritt durch eine Feldstudie zusammengetragen. Siehe dazu Anhang [1] und [2], Interviews mit den Sozialarbeiterinnen.

Der Community Worker ist einer der Hauptakteure im gesamten MHC – PMS. Er ist verantwortlich für die Betreuung der Patienten – sei dies in verschiedenen Zentrum oder bei den Patienten zuhause, als Schnittstelle von Patient zu Arzt oder auch als Organisator für Sozialwohnungen. Details hierzu in Anhang [3].

## 1.2 Definitionen, Akronyme und Abkürzungen

### 1.2.1 Abkürzungen

Abkürzung	Bedeutung
MHC-PMS	Mental Health Care – Patient Management. Das Managementsystem für die gesamte Lösung, hier werden alle Akteure (Patienten, Ärzte, Sozialarbeiter, Besucher) und Datenbanken (Termine, Medikamente, etc) verwaltet. Weiter ist APOLLO ein Teil des MHC-PMS
GPS	Global Positioning System (GPS) ist ein globales Navigationssatellitensystem zur Positionsbestimmung und Zeitmessung.
DB	Datenbank
UC	Use Case

### 1.2.2 Definitionen

Definition	Bedeutung
APOLLO	Expertensystem für MHC – PMS, designed für den Benutzer „Community Worker“

## 2 Stakeholder

### 2.1 Zweck

Auf die Stakeholder-Nummern in der Spalte ‚STK-ID‘ (STK-1, STK-2, usw.) wird in den weiteren Kapiteln verwiesen. Die Idee ist es, die Anforderungen den Stakeholdern für die Beschreibung zuzuweisen.

### 2.2 Liste der Stakeholder

STK-ID	Stakeholder	Info
STK-1	Projektauftraggeber	BTI7081, Jürgen Vogel und Urs Künzler
STK-2	Fachverantwortlicher	Team Black

### 3 Auflistung der Benutzervoraussetzungen (User Requirements)

Die Benutzervoraussetzungen für die mobile Applikation (User „Health Visitor“) werden in funktionale und nicht- funktionale unterteilt.

#### 3.1 Funktionale Anforderungen

##### 3.1.1 Planungsunterstützung (Apollo) (UC-1)

Die Applikation soll automatisch eine optimale Tagesplanung erstellen. Dabei werden die aktuelle Position (geografisch), die eigenen Aufgaben (Tasks), sowie die Aufgaben der Kollegen berücksichtigt. Die Termine haben unterschiedliche Prioritäten, die massgebend für den Tagesablauf sind. Fixe Termine mit den Klienten dienen als Grundpfeiler des Tages (weil die nicht verschoben werden können oder dürfen), die restlichen Arbeiten werden möglichst optimal an diese angepasst.

Bei Aufgaben, die bereits über eine längere Zeit in der Liste sind und aus verschiedenen Gründen wie Auslastung, Arbeitsort, unerwartete Termin usw. verschoben werden mussten, erhöht sich die Priorität kontinuierlich.

Die spielerische Tagesablauf-Generierung überrascht den Benutzer täglich und lässt sein Arbeitsleben nicht in einen sich wiederholenden Trott verfallen.

##### 3.1.2 Verfügbarkeit von Klienten-Daten (UC-2)

Die Dossiers der Klienten können jederzeit und überall abgerufen werden, sofern eine Internetverbindung vorhanden ist. Eine lange Vorbereitungszeit vor den Klienten-Meetings, welche bis anhin noch auf Papier erfolgte, kann so vermieden werden. Selbst während der Verschiebung können also sämtliche Klienten-Infos, die für das bevorstehende Meeting essentiell sind, abgerufen und konsultiert werden. Genauso kann die gesamte Krankheitsgeschichte des Klienten angeschaut werden.

##### 3.1.3 Unterstützung bei der Benutzung des öffentlichen Verkehrs (UC-3)

Die Applikation soll anhand des Tagesplans (Kalender) und der aktuellen Position (geografisch) des Benutzers das Ticket zur Benutzung des öffentlichen Verkehrsmittels beziehen.

Das heisst, wenn der Kalender (Apollo) eine Verschiebung des Arbeitsplatzes vorsieht, sei es ein Bürowechsel oder ein Besuch eines Klienten, wird automatisch vorgängig ein Zugbillet bezogen, das für diese Zeit gültig ist. Somit entfällt eine mühsame Kostenführung der erworbenen Tickets.

##### 3.1.4 Geo-Location Unterstützung (UC-4)

Die Applikation soll anhand des aktuellen Tasks und eines Kartografie-Dienstes die schnellste Route zum nächsten Termin bereitstellen. Basierend auf bereits gewohnten Applikationen wie Google Maps wird der gesamte Tagesablauf visualisiert und genauestens beschrieben, wann der Benutzer wo sein muss. Sobald eine Kalender-Änderung vorgenommen wird, passt sich diese Route automatisch an.

#### 3.2 Nicht funktionale Anforderungen

##### 3.2.1 Sicherheit

- Sämtliche Daten, ob Kalender-Einträge oder Klienten-Daten, sind nur für berechtigte Benutzer verfügbar.
- Die Sicherheit, dass niemand ohne Autorisierung an die Daten kommt, ist gewährleistet.
- Das Smartphone wird von der App nicht bemerkbar beeinträchtigt.
- Daten können nicht verloren gehen.

##### 3.2.2 Kosten

- Die Kosten werden möglichst niedrig gehalten.
- Alle entstandenen Kosten können begründet werden.
- Die App wird im normalen Gebrauch keine zusätzlichen Kosten verursachen (wie z.B. anfallende Roaminggebühren)

##### 3.2.3 Usability

- Die App muss sehr benutzerfreundliche und ansprechend gestaltet werden.

### 3.2.4 Patienten-Info

- Sämtliche Informationen bezüglich der Klienten müssen leicht abrufbar und übersichtlich sein.

### 3.2.5 Beständigkeit

- Das System soll für den Benutzer gefühlt immer verfügbar sein.
- Die Arbeit darf nicht im negativen beeinträchtigt werden.

## 4 Auflistung der Systemvoraussetzungen (System Requirements)

Die Systemvoraussetzungen werden in funktionale und nicht funktionale unterteilt.

### 4.1 Funktionale Anforderungen

Status der funktionalen Anforderungen

UC-ID	Titel des Use Case	STK-ID (fett = Verantwortlich)	Priorität (M/K)	Status (OK/NOK)
1	Plan Planungsunterstützung	<b>2</b>	M	NOK
2	Verfügbarkeit von Klienten Daten	<b>2</b>	M	NOK
3	Unterstützung bei der Benützung des öffentlichen Verkehrs	<b>2</b>	M	NOK
4	Geolokation Unterstützung	<b>2</b>	M	NOK

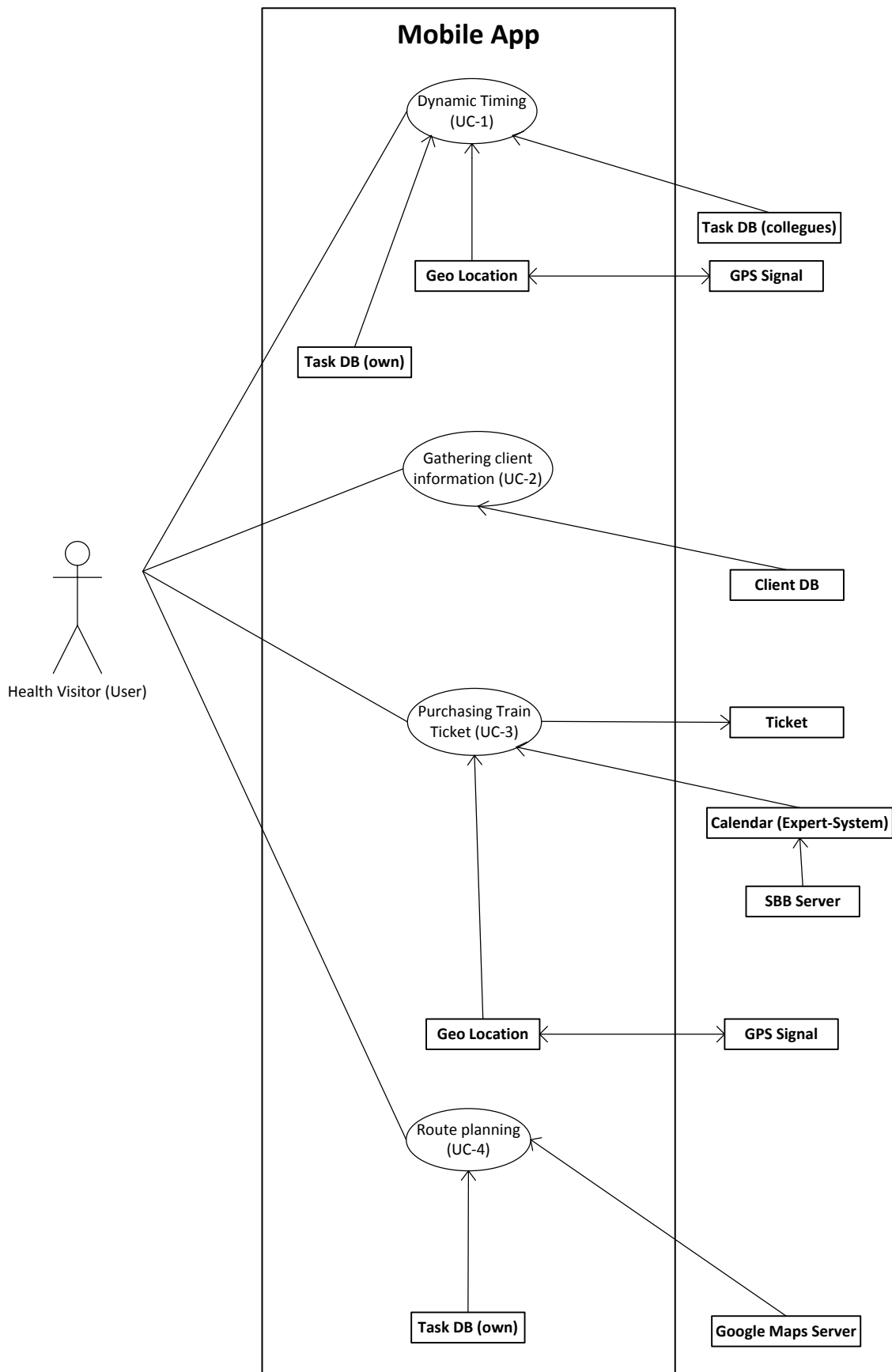
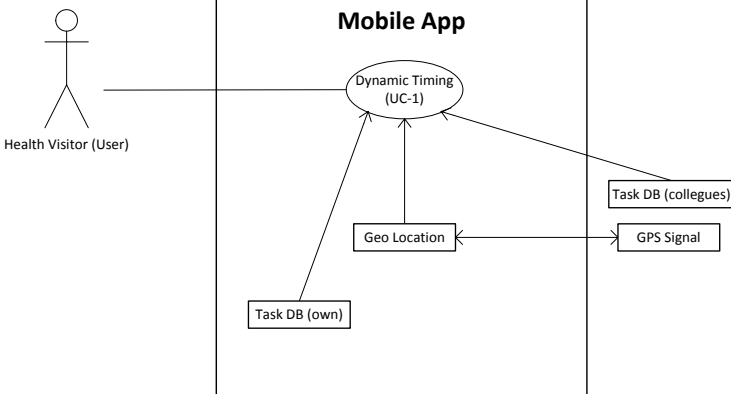
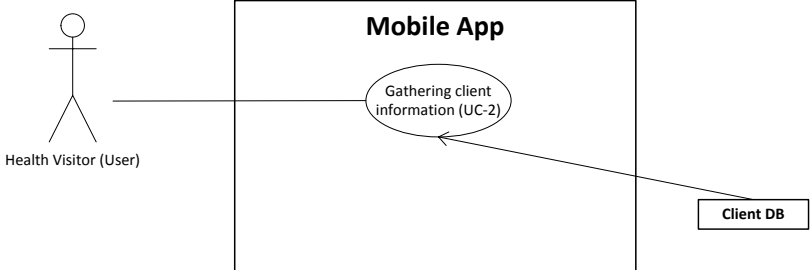


Abbildung 1: Use Cases

#### 4.1.1 Planungsunterstützung (UC-1)

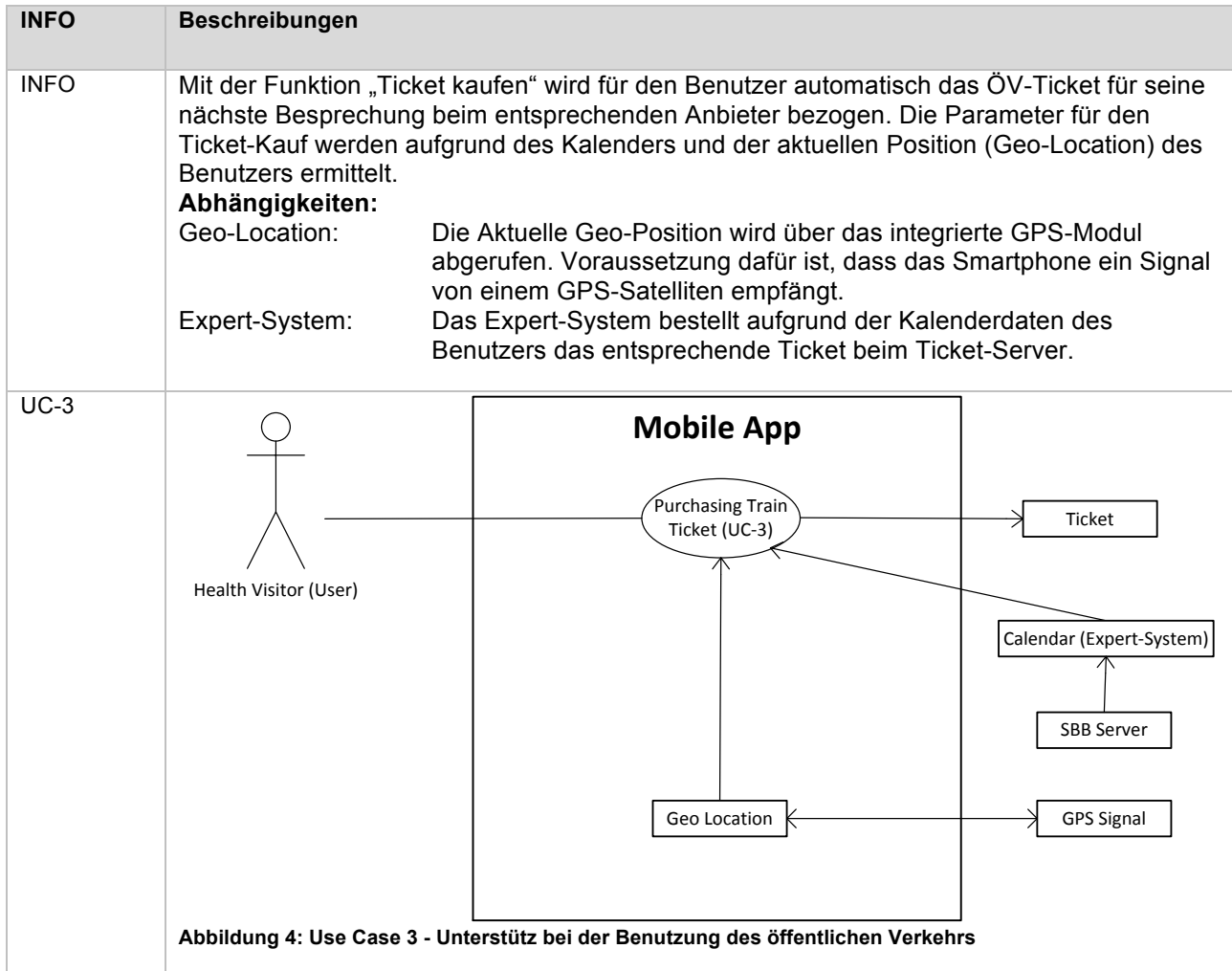
INFO	Beschreibungen
INFO	<p>Die Applikation verfügt über eine Funktion „Tagesplan erstellen“, diese erstellt einen optimalen Tagesplan anhand der Tasks des Benutzers und dessen Kollegen, sowie der aktuellen Position (Geo-Location) des Benutzers. Fixe Termine bilden das Grundgerüst des Plans und können nicht verschoben oder gelöscht werden. Anschliessend werden die weiteren Tasks nach Priorität und Ort hinzugefügt, dabei wird berücksichtigt, dass möglichst keinen Totzeit entsteht.</p> <p>Falls nötig kann der Benutzer seinen Tagesplan nach dem Erstellen modifizieren.</p> <p>Bei Terminen, die bereits mehrfach verschoben wurde erhöht sich die Priorität kontinuierlich.</p> <p><b>Abhängigkeiten:</b></p> <p>Eigene Tasks: Die Eigenen Tasks werden von der Task-Datenbank mittels SQL abgerufen</p> <p>Geo-Location: Die Aktuelle Geo-Position wird über das integrierte GPS-Modul abgerufen. Voraussetzung dafür ist, dass das Smartphone ein Signal von einem GPS-Satelliten empfängt.</p> <p>Task von Kollegen: Die Tasks der Kollegen werden von der Task-Datenbank des MHC-PMS-Servers mittels SQL abgerufen. Diese wird mit den Datenbanken aller Benutzer-Apps synchronisiert.</p>
UC-1	 <p><b>Abbildung 2: Use Case 1 - Planungsunterstützung</b></p>

#### 4.1.2 Verfügbarkeit von Klienten-Daten (UC-2)

INFO	Beschreibungen
INFO	<p>Die Dossiers der Klienten können jederzeit und überall abgerufen werden, sofern eine Internetverbindung vorhanden ist. Sie enthalten alle relevanten Daten über den Klienten.</p> <p><b>Abhängigkeiten:</b></p> <p>Klienten DB: Die Klienten-Daten werden von der Klienten-Datenbank des MHC-PMS-Servers mittels SQL abgerufen</p>
UC-2	 <p><b>Abbildung 3: Use Case 2 – Verfügbarkeit von Klienten-Daten</b></p>



#### 4.1.3 Unterstützung bei der Benutzung des öffentlichen Verkehrs (UC-3)



Nr. und Name:	3 – Ticket für ÖV automatisch lösen
Szenario:	
Kurzbeschreibung:	Der Health visitor hat einen externen Termin und wird mit dem ÖV anreisen. Das mobile Gerät erkennt via GPS, dass er sich an einer Bus-/Bahnhofstation befindet und löst automatisch das entsprechende Ticket (gem. Terminplanung)
Beteiligt Akteure:	Health visitor
Auslöser / Vorbedingung:	<ul style="list-style-type: none"> <li>Health visitor hat externen Termin</li> <li>Wird mit öffentlicher Verkehr anreisen</li> <li>Mobile hat GPS-Empfang</li> </ul>
Ergebnisse / Nachbedingung:	Benötigtes Ticket für die entsprechende ÖV-Strecke wurde korrekt gelöst

#### Ablauf:

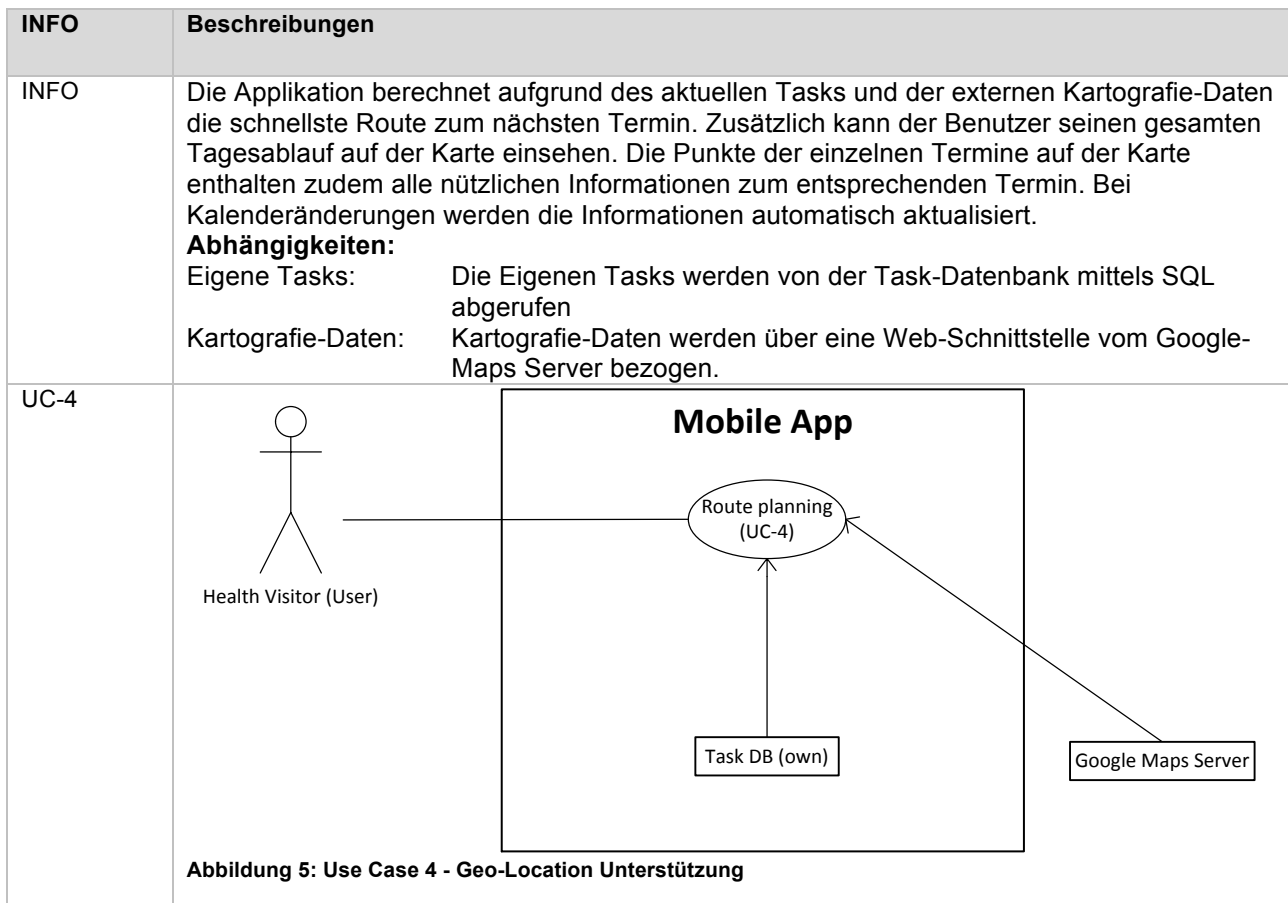
Nr.	Wer	Was
1.0	System	System fragt via GPS-Empfänger Standort des Users ab
1.1	System	Überprüfung, ob aktueller Standort mit Bus-/Bahnhofstation übereinstimmt
1.2	System	Ausnahme: Aktueller Standort befindet sich nicht an einer Station
1.3	System	Prüft, ob im Terminplaner einen Termin mit Anreise eingetragen ist

1.4	System	Ausnahme: Termin erfordert keine Anreise
1.4	System	Meldung an Benutzer, dass Bus-/Bahnhof erkannt wurde
1.5	System	Frage, ob Benutzer für Anreise zum aktuellen Termin ein Ticket lösen will
1.6	Benutzer	Bestätigen oder Verwerfen der Ticketanfrage
1.7	Benutzer	Ausnahme: Verwerfen der Anfrage
1.8	System	Kontaktiert Terminplaner und liest dort Uhrzeit und Ort des anstehenden Termins
1.9	System	Kontaktiert SBB-Server für Fahrplan-Abfrage (Zeit und Zielort gem. vorherige Abfrage Terminplaner)
2.0	SBB-Server	Ausnahme: keine Verbindung/Antwort
2.1	System	Wählt nächst-, bzw. bestmögliche Verbindung zum Zielort
2.2	System	Kontaktiert Benutzerprofil, um dort Informationen zum Ticketkauf zu erhalten (Halbtax, 1./2. Klasse, usw.)
2.3	System	Löst Ticketkauf aus
2.4	System	Ausnahme: Ticketkauf nicht erfolgreich
2.5	System	Meldung, dass Ticketkauf erfolgreich
2.6	System	Anzeige des gekauften Tickets

#### Ausnahmen, Varianten:

Nr.	Wer	Was
1.2	System	Aktueller Standort befindet sich nicht an einer Station
1.2.1	System	Keine spezielle Aktion, weiterhin Standort laufend abfragen
1.3	System	Termin erfordert keine Anreise
1.3.1	System	Keine spezielle Aktion,
1.7	Benutzer	Verwerfen der Anfrage
1.7.1	System	Keine speziellen Aktionen, solange Benutzer an der gleichen Bus-/Bahnhof keine weiteren Abfragen mehr zu Ticketkauf.
2.0	SBB-Server	Keine Verbindung/Antwort
2.0.1	System	Fehlermeldung an Benutzer ausgeben, dass SBB-Server für Ticketkauf nicht kontaktiert werden konnte, Abbruch des Vorgangs
2.4	System	Ticketkauf nicht erfolgreich
2.4.1	System	Fehlermeldung an Benutzer aufgeben, dass Ticketkauf nicht korrekt abgewickelt werden konnte, Abbruch des Vorgangs

#### 4.1.4 Geo-Location Unterstützung (UC-4)



Nr. und Name:	4. Routenplanung durch APOLLO
Szenario:	Health Visitor muss mehrere Kunden in der ganzen Stadt Bern besuchen.
Kurzbeschreibung:	<p>Health Visitor hat einen oder mehrere Termine an verschiedenen Lokalitäten. Dies können verschieden Spitäler, Community Zentrums oder auch Sozialwohnungen sein. Einfacherweise durch Geopositionsdaten identifizierbar.</p> <p>Es braucht eine optimale Route für die gewählten Ziele und das gewählte Verkehrsmittel.</p> <p>Weiter soll bei einer ÖV Benutzung automatisch beim Betreiber ein Ticket gelöst werden.</p>
Beteiligt Akteure:	Health Visitor
Auslöser / Vorbedingung:	Termine an verschiedenen Lokalitäten sind vorhanden.
Ergebnisse / Nachbedingung:	<p>Termine konnten wahrgenommen werden.</p> <p>Gesamtstrecke, resp. Zeit wurde optimal geplant</p>

#### Ablauf:

Nr.	Wer	Was
1.1	Health Visitor	Startet die Applikation
1.2	APOLLO	Analysiert die letzte Terminplanung, welche vom Benutzer abgesegnet wurde.

1.3	APOLLO	Analysiert die Prämissen (Positionsdaten und Verkehrsmittel) und stellt so eine Route fest. → Falls ÖV, suchen der nächsten Haltestellen bei den Start- und Zielorten und diese verwenden für die Planung. → Falls Individualverkehr, direkt mit einer Distanzmatrix beginnen.
1.4	APOLLO	Ausgabe der Route an den Health Visitor
1.5	Health Visitor	Akzeptieren, Ablehnen oder Modifizieren dieses Vorschlags
1.6	APOLLO	Benutzereingabe verarbeiten.
1.7	APOLLO	Route manifestieren
1.8	APOLLO	Anzeige Kartenmaterial für den Benutzer (als Kontrolle)

#### Ausnahmen, Varianten:

Nr.	Wer	Was
1.2.1	APOLLO	Keine Terminplanung vorhanden → keine Routenplanung nötig. Ende
1.2.2	APOLLO	Notfalltermin wurde durch das Management eingetragen. Automatische Anpassung des Terminplans des Benutzers durch APOLLO, ohne Benutzerrückfrage (Notfall)
1.3.1	APOLLO	Falls bereits bekannte Route (beispielsweise bei wöchentlichen wiederkehrenden Terminen) diese laden. Ende
1.5.1	Health Visitor	Akzeptieren der Route. APOLLO übernimmt die vorbereiteten Daten.
1.5.2	Health Visitor	Ablehnen der Route. APOLLO kehrt zum letzten Task zurück (Terminplanung, Audioaufnahme, etc). Ende
1.5.3	Health Visitor	Modifizieren der Route. APOLLO übernimmt die neuen Prämissen. Neuberechnung, 1.3.

## 4.2 Nicht funktionale Anforderungen

### 4.2.1 Sicherheit

- Jeder Sozialarbeiter muss sich einloggen und seine eigenen Daten abrufen können.
- User-spezifische Daten dürfen nicht von fremden Benutzern ersichtlich sein.
- Ohne Login dürfen keine Daten ersichtlich sein.
- Daten die geteilt werden, können nur von den ausgewählten Benutzern mit entsprechenden Berechtigungen angesehen werden.
- Patienten-Infos dürfen nur dem zugeteilten Sozialarbeiter zur Verfügung stehen.
- Patienten-Privatsphäre unterliegt den lokalen ethischen Grundsätzen.
- Die Backup-Server müssen physikalisch getrennt von den normalen Systemservern sein.
- Die Applikation darf das Smartphone nicht beschädigen.

### 4.2.2 Kosten

- Entwicklungskosten müssen vernünftig und begründet sein.
- Unterhaltungskosten müssen möglichst niedrig gehalten werden.
- Weiterentwicklungskosten müssen im Masse der Entwicklungskosten sein.
- Die Schulungskosten der User müssen möglichst tief oder gar nicht notwendig sein.

- Die Benutzung der App darf keine Roaminggebühren verursachen.

#### 4.2.3 Usability

- Sämtliche User müssen die App ohne grosse Schwierigkeiten bedienen können.
- Allgemein muss die App sehr benutzerfreundlich sein.
- Das Look-and-Feel der App muss modern, jung und freundlich wirken.

#### 4.2.4 Patienten-Info

- Aussergewöhnliche Vorkommnisse müssen in die DB gespeichert werden können.
- Sämtliche Informationen (ob Grund-Infos oder unerwartete Zusatz-Infos) müssen leicht abrufbar und übersichtlich sein.

#### 4.2.5 Beständigkeit

- Das System soll zu min. 98% der Zeit störungsfrei laufen.
- Die Daten müssen laufend backuped werden.
- Bei einem grösseren Ausfall der Systeme muss ein Notsystem innert kurzer Zeit verfügbar sein.

## 5 Testing

Das Testing basiert hauptsächlich auf den Use Cases. Ein Test kann die Existenz von Fehlern aufzeigen, nie aber das Nichtvorhandensein. Deshalb ist es wichtig, dass die Tests iterativ durchgeführt werden mit leicht angepasstem Setting.

Die ersten Testprozesse werden bereits während der Entwicklung durchgespielt. Sollten alle Tests erst nach einem umfassenden Build erfolgen, könnte das enorme Auswirkungen mit sich ziehen. Module ändern, die fest einprogrammiert sind, erweist sich als schwierig. Deshalb ist es erstrebenswert, laufend kleinere Module einer Testphase zu unterziehen und eine ausführliche Auswertung vorzunehmen.

### Phasen

1. Test planen
2. Test vorbereiten
3. Testpersonen instruieren
4. Test durchführen
5. Test auswerten
6. Test abschliessen

Die gesamte Testphase kann mehrmals iteriert werden. Die Punkte 4 und 5 können zusätzlich iterativ sein.

### Funktionales Testing

Diese Art von Testing kann laufend intern durchgeführt werden. Die Ergebnisse liefern im Normalfall klare, boolsche Antworten. Ja hat funktioniert, nein hat nicht funktioniert. Der Umfang dieser Tests kann von einer einzigen Abfrage, die innert Sekunden programmiert ist, bis hin zu einem gesamten Arbeitsablauf sein, der mehrere Module beinhaltet.

### Nicht funktionales Testing

Diese Tests werden in derselben Ausführung gleichzeitig an mehreren Benutzern getestet. Die Ergebnisse fallen nicht offensichtlich in zwei Kategorien. Deshalb müssen sie detaillierter analysiert werden, was einen grösseren Aufwand bedeutet. Die Auswertung umfasst eine umfassende Befragung der Testpersonen, die möglichst nahe an die realen Users rankommen. Aufgrund dieser Erkenntnisse können die Non-Functional-Requirements Schritt für Schritt an die ideale App angepasst werden.

## 6 System Architektur

Das Apollo EXPERTENSYSTEM, welches die gesamten Funktionen für den Health Visitor bereitstellt, beruht auf einer Client-Server Architektur. Serverseitig baut es auf einer MYSQL-Datenbank auf, auf welche die Businesslogik über eine JDBC Schnittstelle zugreift. Die Businesslogik ist zum einen Teil ein separater Layer, geschrieben in Java, welche auf dem Applikationsserver läuft. Ein zweiter Teil ist direkt auf der Datenbank mittels Funktionen und Prozeduren einprogrammiert.

Der Webservice greift auf den Java-Layer zu, welcher ihm die geforderten Daten aufbereitet und zur Verfügung stellt.

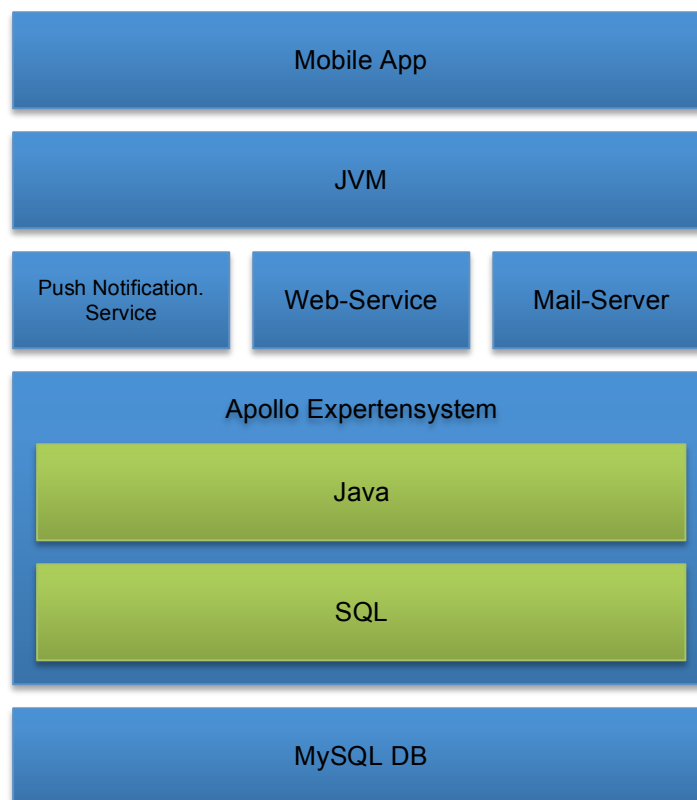


Abbildung 6: System Architektur

## 7 Anhang

Nr.	Titel
[1]	CS1 Task 3_Interview_SocialWorker.docx
[2]	interview_community_worker.docx
[3]	CS1_Task1.pdf

## 8 Abbildungsverzeichnis

Abbildung 1: Use Cases .....	7
Abbildung 2: Use Case 1 - Planungsunterstützung .....	8
Abbildung 3: Use Case 2 – Verfügbarkeit von Klienten-Daten .....	8
Abbildung 4: Use Case 3 - Unterstütz bei der Benutzung des öffentlichen Verkehrs .....	9
Abbildung 5: Use Case 4 - Geo-Location Unterstützung .....	11
Abbildung 6: System Architektur .....	15