



Bericht Team Grün

Studiengang Projektlabor Medizinische Software-Entwicklung

Name : Remziye Celik, Laura Huber, Yong-Chan Kwon, Jana Tomic, Thomas Liebgott

Studiengang Projektlabor Medizinische Software-Entwicklung

Semester Wintersemester 22/23

Standort Mannheim

Betreuer Prof. Dr. Mark Hastenteufel

Inhaltsverzeichnis

Inhaltsverzeichnis

A - Technische Dokumentation	2
1. Allgemeine Produktbeschreibung	2
1.1 Zweckbestimmung	2
1.2 High-Level Features der Software	2
1.3 Hersteller	3
1.4 Medizinischer Hintergrund	3
1.5 Ähnliche Produkte auf dem Markt	4
2. Software-Spezifikation	4
3. Software-Architektur	7
4. Software-Design	9
5. Software-Verifikation	15
5.1 Testspezifikationen und Testergebnisse	15
5.2 Verbleibende Bugs und Bewertung	16
5.2.a - Verschiedene Bugs vorhanden:	16
5.2 b - Mögliche Verbesserungen:	17
5.3 Traceability Matrix	17
6. Liste der SOUP	18
B - Entwicklungsprozess	18
7. Software-Entwicklungsplan	18
8. Meilensteine/Sprints	18
9. Verwendung der Versionsverwaltung	19
10. Team Meetings	19
11. Dokumentation und Erhebung von SW-Anforderungen	19
12. Testplan	20
13. Mitarbeiter und Rollen/Verantwortlichkeiten	21
14. Liste der eingesetzten Tools	22
Anhang	24

A - Technische Dokumentation

1. Allgemeine Produktbeschreibung

Eine schnelle Reaktion auf sich ändernde Parameter ist besonders bei der Patientenüberwachung essentiell und kann für den Gesundheitszustand entscheidend sein. Umso wichtiger sind übersichtliche und echtzeitfähige Vitaldaten-Überwachungssysteme, die das Klinikpersonal bei der Überwachung der Patienten unterstützen und zur Fehlerminimierung beitragen.

1.1 Zweckbestimmung

Bei dem in diesem Projekt entwickelten Produkt „ViewMed“ handelt es sich um ein System zur dauerhaften Überwachung von Vitalparametern und Verwaltung der Zimmerbelegung von Patienten. Das System ist für den Krankenhausgebrauch gedacht und soll auf Intensiv- sowie Krankenstationen eingesetzt werden. Hierbei ist die Software für geschultes, medizinisches Personal, wie Ärzte, Krankenpfleger und sonstiges Pflegepersonal konzipiert und soll die Überwachung der Patienten erleichtern.

„ViewMed“ ist für die Überwachung von Patienten ausgelegt, deren gesundheitlicher Zustand instabil ist und überwacht werden muss, um schnell auf kritische Veränderungen der Vitalparameter reagieren zu können. Mit dem System können Patientengruppen in verschiedenen Altersgruppen (Säuglinge, Kinder und Erwachsene bis ins hohe Alter) überwacht werden. Hierbei werden Parameter wie Herzfrequenz, Sauerstoffsättigung, Blutdruck, Körpertemperatur und Atemfrequenz angezeigt.

Das System erlaubt es, bis zu 16 Patienten auf einem Monitor anzuzeigen. Darüber hinaus verfügt das System über ein Alarmsystem, welches auf kritische Veränderungen der Vitaldaten hinweist und für jeden Patienten individuell eingestellt werden kann.

1.2 High-Level Features der Software

Die entwickelte Software setzt sich aus drei Subsystemen zusammen: dem Remote Monitor, einem Admin-System und einem Vitaldatensimulator.

Der Remote Monitor wird für die Darstellung der Vitalparameter eingesetzt. Hierbei kann der Nutzer zwischen 4 Ansichtseinstellungen wechseln. Es gibt eine Gesamtansicht, bei der bis zu 16 Patienten gleichzeitig angezeigt werden können. Möchte man den Fokus auf einen bestimmten Patienten legen, kann man in die Einzelansicht wechseln. Eine Ansicht mit 4, oder 9 Patienten ist auch möglich. Zusätzlich verfügt der Remote Monitor über die Funktion, für jeden Patienten individuelle Alarmgrenzen für bestimmte Parameter wie Herzfrequenz, Sauerstoffsättigung, Blutdruck, Körpertemperatur und Atemfrequenz einstellen zu können. So kann auf die individuellen Bedürfnisse des Patienten eingegangen werden. Neben den Vitaldaten wird auch ein Early-Warning-Score berechnet, der eine ungefähre Prognose über den Zustand des Patienten erlaubt.

Die Verwaltung von Patienten und Monitoren wird über das Admin-System geregelt. Hier können über eine Konsolenanwendung, Patienten sowie Monitore angelegt werden. Zusätzlich werden über das Admin-System Patienten den Monitoren zugewiesen und wieder gelöscht, wenn Patienten entlassen werden, oder Monitore ausfallen/ defekt werden.

Der Vitaldatensimulator simuliert demonstrativ die Patientenvitalwerte, um das System testen und verbessern zu können. Simuliert werden Parameter wie Herzfrequenz, Sauerstoffsättigung, Blutdruck, Körpertemperatur und Atemfrequenz.

1.3 Hersteller

Die „ViewMed“ Software wurde an der Hochschule Mannheim, im Rahmen der Vorlesung „Projektlabor Medizinische Softwareentwicklung“ entwickelt. Diese Vorlesung wird im Masterstudiengang angeboten. An der Entwicklung beteiligt waren Chan, Remziye, Thomas, Laura und Jana unter der Aufsicht und Beratung von Herrn Hastenteufel.

1.4 Medizinischer Hintergrund

Eine durchgehende und genaue Überwachung der Vitaldaten eines Patienten ist eine der wichtigsten Komponenten der Patientenversorgung in einem Krankenhaus. So wird die optimale Pflege und Behandlung der Patienten gewährleistet. Vor allem in der Intensivpflege ist es wichtig, schnell auf sich ändernde Parameter zu reagieren. Systeme, wie das hier entwickelte „ViewMed“, unterstützen das Krankenhauspersonal und erleichtern die Überwachung der Patientenvitaldaten.

Eine der Hauptkomponenten des Systems ist der Remote Monitor. Dieser ermöglicht dem Nutzer eine übersichtliche Darstellung von bis zu 16 Patienten gleichzeitig. Diese können explizit ausgewählt werden. Sollte es erforderlich sein, den Fokus auf einige wenige kritische Patienten zu legen, kann man zwischen einer Einzelansicht, einer Ansicht mit 4 Fenstern, oder einer Ansicht mit 9 Fenstern wechseln. Der Remote Monitor verfügt zusätzlich über die Funktion, für jeden der gemessenen Vitaldaten (Herzfrequenz, Sauerstoffsättigung, Blutdruck, Körpertemperatur und Atemfrequenz) individuelle Alarmgrenzen einstellen zu können. So können besonders kritische Parameter besser überwacht werden.

Das Administrationssystem ermöglicht eine einfache und übersichtliche Verwaltung der Patientenmonitore. Es bietet eine Übersicht über die Monitorbelegung und erlaubt eine schnelle Zuordnung von Patienten und Monitoren. Das trägt zur schnellen und übersichtlichen Organisation der Patientenzimmer bei.

Der Vitaldatensimulator ist primär für den Entwicklungsprozess relevant. Er erzeugt die Vitaldatenparameter und imitiert somit einen Patienten. Diese werden an den Remote Monitor gesendet.

1.5 Ähnliche Produkte auf dem Markt

Ähnliche Produkte auf dem Markt, die zum Anzeigen von Vitaldatenparametern in Krankenhäusern verwendet werden, sind zum Beispiel das „IntelliVue MX850“ von Phillips, oder der „E10 Patientenmonitor“ von Witleaf.

2. Software-Spezifikation

Remote - Monitor:

Nr.	User-Stories	Akzeptanzkriterium
R1	Als Pflegepersonal möchte ich die Patientendaten (Name, Vorname) direkt erkennen können, um den Patienten zu identifizieren.	Vorname und Nachname werden in Einzel- und Gesamtansicht gut sichtbar angezeigt.
R2	Als Pflegepersonal möchte ich aktuelle Vitaldaten des Patienten auf einem Monitor sehen, um die Vitaldaten besser überwachen zu können.	In der Einzel- und Gesamtansicht werden die aktuellen Vitaldaten in Echtzeit angezeigt.
R3	Als Pflegepersonal möchte ich die unterschiedlichen Vitaldaten (wie Herzfrequenz, Sauerstoffsättigung, Blutdruck, Körpertemperatur und Atemfrequenz) eindeutig voneinander unterscheiden können	In der Einzel- und Gesamtansicht werden die unterschiedlichen Vitaldaten in unterschiedlichen Farben angezeigt. So können die Vitaldaten eindeutig voneinander unterschieden werden.
R4	Als Pflegepersonal möchte ich nur den aktuell angeschlossenen Monitor eines Patienten sehen, um die richtigen Vitaldaten des Patienten zu überwachen	In der Einzel- und Gesamtansicht ist die Monitor-ID des aktuell zugeordneten Monitors angezeigt
R5	Als Personal möchte ich bis zu 16 Patienten gleichzeitig, mit zugehörigen Patientenmonitoren, sehen können, um mehrere Patienten gleichzeitig überwachen zu können	Patienten sind in der Datenbank angelegt und in der Belegungstabelle Monitoren zugewiesen. Alle diese 16 Patienten können gleichzeitig angezeigt werden
R6	Als Pflegepersonal möchte ich jedem Patienten individuelle Grenzwerte für jeden Vitalparameter angeben können. So kann eine	Für jeden Patienten kann ein Einstellungsmenü geöffnet werden, in dem individuelle

	bessere individuelle Versorgung sichergestellt werden.	Alarmgrenzen für jeden Vitalparameter einzeln eingestellt werden können
R7	Als Pflegepersonal möchte ich, neben den Vitalparametern, für jeden Patienten den individuellen Early-Warning-Score angezeigt bekommen. So kann eine ungefähre Prognose über den Gesundheitszustand des Patienten erstellt werden.	Neben den Vitaldaten wird, für jeden Patienten, der berechnete Early-Warning-Score in einem Fenster angezeigt
R8	Als Pflegepersonal möchte ich, im Falle einer kritischen Veränderung der Vitalparameter einen hörbaren Alarmton wahrnehmen, um schnell reagieren zu können	Sobald der kritische Wert eines Vitalparameters* über- oder unterschritten ist, ertönt ein kurzer Alarmton.
R9	Als Pflegepersonal möchte ich, im Falle einer kritischen Veränderung der Vitalparameter, eindeutig sehen, um welchen Parameter es sich handelt, damit ich schnellstmöglich und spezifisch auf die Veränderung reagieren kann	Der Vitalparameter, der einen kritischen Wert über- oder unterschreitet wird, erscheint in roter Schrift und es erscheint ein roter Rahmen um den Wert
R10	Als Pflegepersonal möchte ich durch ein visuelles Signal auf dem Remote Monitor informiert werden, wenn die Netzwerkverbindung unterbrochen ist, um die Aktualität der Daten gewährleisten zu können	In der Einzel- und Gesamtansicht erscheint ein Alarmfenster, das auf die unterbrochene Netzwerkverbindung hinweist. Dieses erscheint, sobald keine Vitalparameter empfangen werden

Administration

Nr.	User-Stories	Akzeptanzkriterium
A1	Als Administrator möchte ich einer Patientendatenbank neue Patienten, mit dazugehörigen Patientendaten, anlegen können (Name, Vorname, Geburtsdatum, Adresse)	Über die Konsole können Patienten neu angelegt werden und der Patientendatenbank hinzugefügt werden
A2	Als Krankenhauspersonal möchte ich einem bestimmten Patienten einen Monitor mit Monitor-ID zuordnen, um seine Vitaldaten zu visualisieren.	Über eine Konsole können Patienten mit einem Monitor, mit Hilfe einer Tabelle, verknüpft werden
A3	Als Administrator möchte ich angelegte Patienten aus der Datenbank löschen können	Über die Konsole können Patienten gelöscht werden

Nr.	User-Stories	Akzeptanzkriterium
A1	Als Administrator möchte ich einer Patientendatenbank neue Patienten, mit dazugehörigen Patientendaten, anlegen können (Name, Vorname, Geburtsdatum, Adresse)	Über die Konsole können Patienten neu angelegt werden und der Patientendatenbank hinzugefügt werden
A4	Als Administrator möchte ich hinterlegte Patientenmonitore aus der Datenbank löschen können	Über die Konsole können Patientenmonitore gelöscht werden und die zugehörige Belegung aus der Datenbank
A5	Als Administrator möchte ich einem belegten Monitor einem neuen Patienten zuordnen können, um die Vitaldaten des neuen Patienten zu visualisieren	Über die Konsole kann, in der Patientendatenbank, die Verknüpfung zwischen Patienten und Monitor getrennt werden. Der Monitor kann mit einem anderen Patienten verknüpft werden. Die aktualisierten Vitaldaten und Monitor-ID werden korrekt angezeigt.

Vitaldaten-Simulator:

Nr.	User-Stories	Akzeptanzkriterium
V1	Als Administrator möchte ich einen neuen Patientenmonitor, für die Simulation von Vitaldaten, durch eine eindeutige Seriennummer angeben können	In einem dafür vorgesehenen Feld kann die Monitor-ID angegeben werden. Die Vitaldaten werden an die angegebene Monitor-ID gesendet. Bei Änderung der ID, wird sofort an die neue ID gesendet.
V2	Als Tester möchte ich, dass der Simulator Vitaldaten wie Herzfrequenz, Sauerstoffsättigung, Blutdruck, Körpertemperatur und Atemfrequenz in einem realistischen Wertebereich simuliert. Die Vitaldaten sollen sich regelmäßig ändern und so natürliche Schwankungen simulieren	Beim Starten des Vitaldatensimulators werden Vitaldaten wie Herzfrequenz, Sauerstoffsättigung, Blutdruck, Körpertemperatur und Atemfrequenz angezeigt. Die Werte ändern sich im 1-Sekunden-Takt, um

		natürliche Schwankungen zu simulieren. Die Werte schwanken in realistischen Grenzen.
V3	Als Tester sollen am Simulator kritische Grenzwerte für jeden der angegebenen Vitalparameter simuliert werden können, um die Funktionalitäten des Remotemonitors zu testen.	Über Slider können im Vitaldatensimulator für jeden Vitalparameter kritische Werte eingestellt werden
V4	Als Tester möchte ich eine abgetrennte Netzwerkverbindung, durch Ausschaltung des Internets simulieren können	Durch Ausschalten des Internets wird Sendevorgang beendet

3. Software-Architektur

Abbildung N°1 - Vereinfachte Projektarchitektur

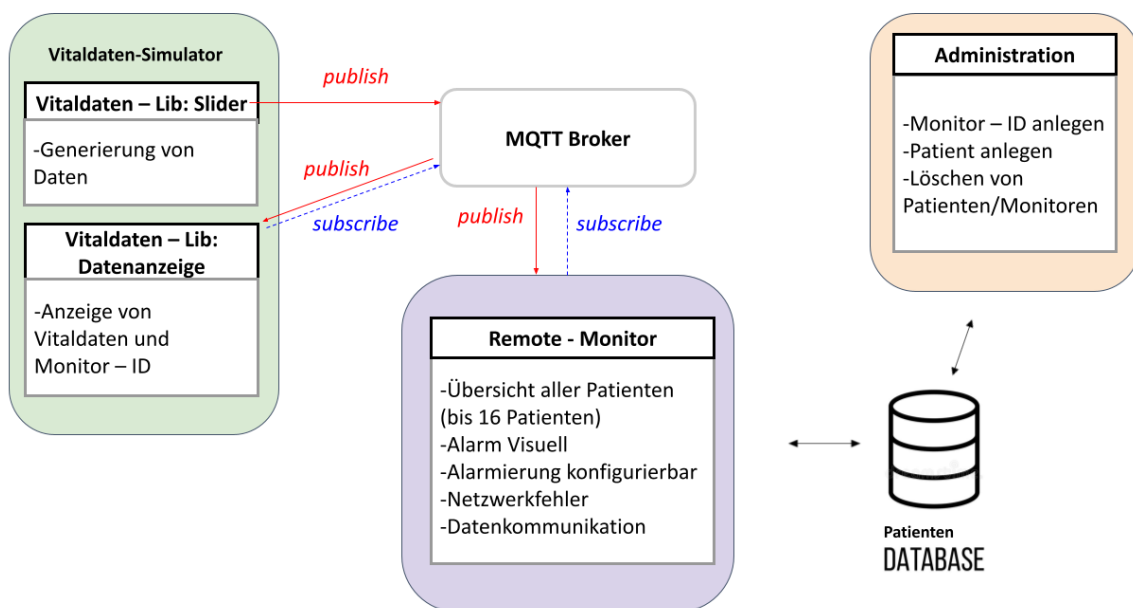


Abbildung N°2 - Detaillierte Projektarchitektur

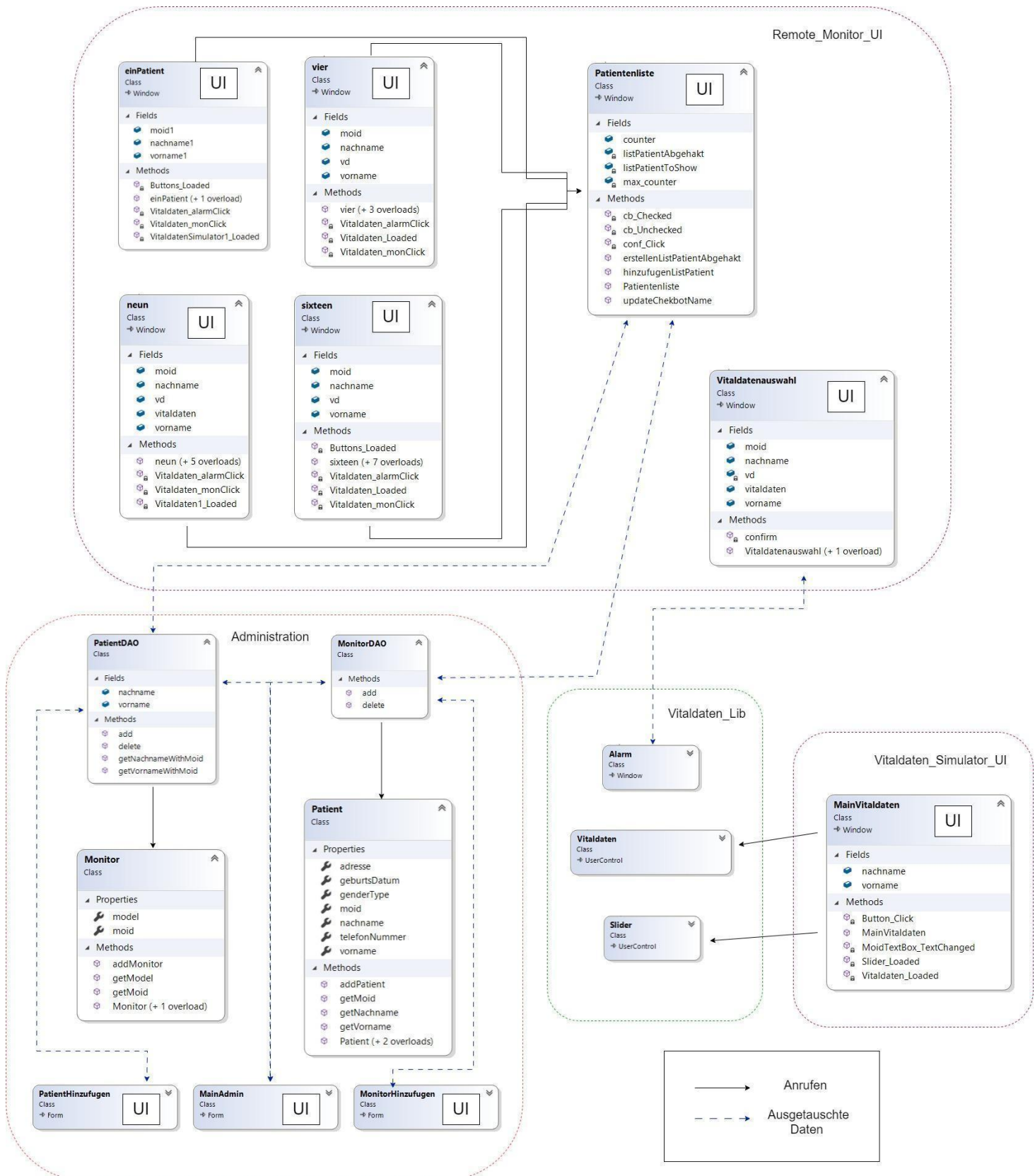
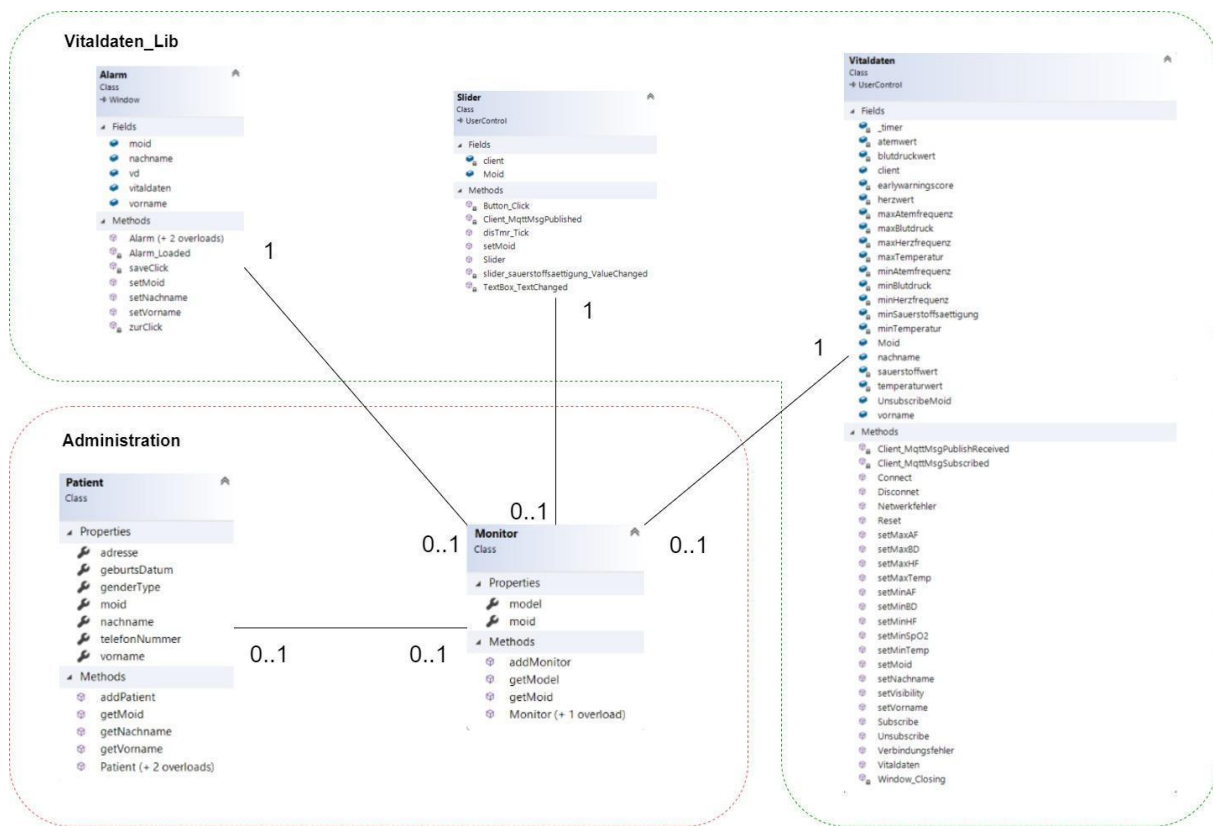


Abbildung N°3 - UML Diagramm

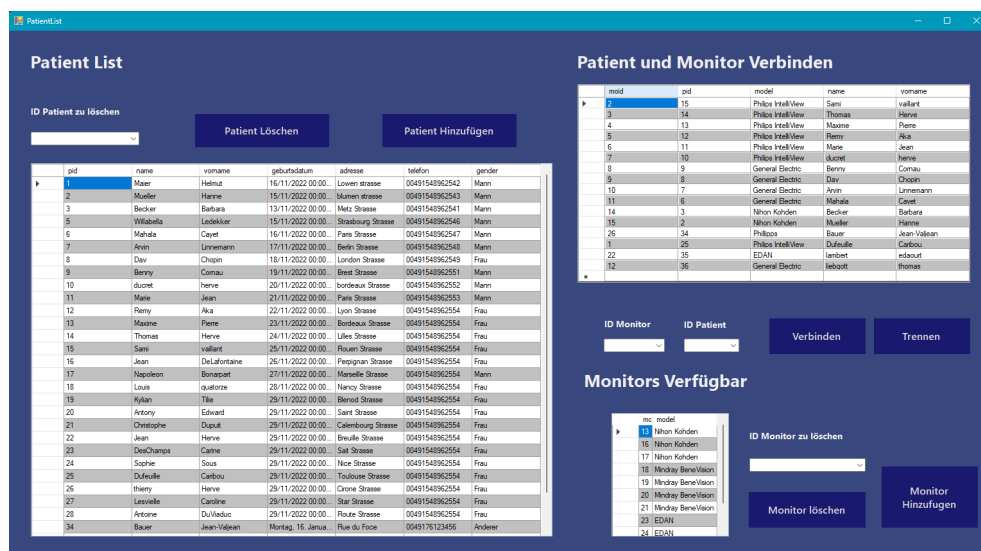


4. Software-Design

1. Administration:

Die Oberfläche des Admin-Systems ist in 3 Bereiche unterteilt. Der linke Bereich dient der Verwaltung der Patienten. Hier können neue Patienten in die Krankenhaus-Akten aufgenommen und über ihre ID-Nummer aus dem Krankenhaus entlassen werden. Weiterhin bildet die rechte untere Ecke die Datenbank für die verfügbaren Monitore im Krankenhaus. Diese kann ggf. um weitere Monitore ergänzt werden, während defekte Modelle nach demselben Prinzip der Patienten ersetzt werden können. Schließlich werden diese in der rechten oberen Ecke mit den Patienten über die ID-Nummern verknüpft.

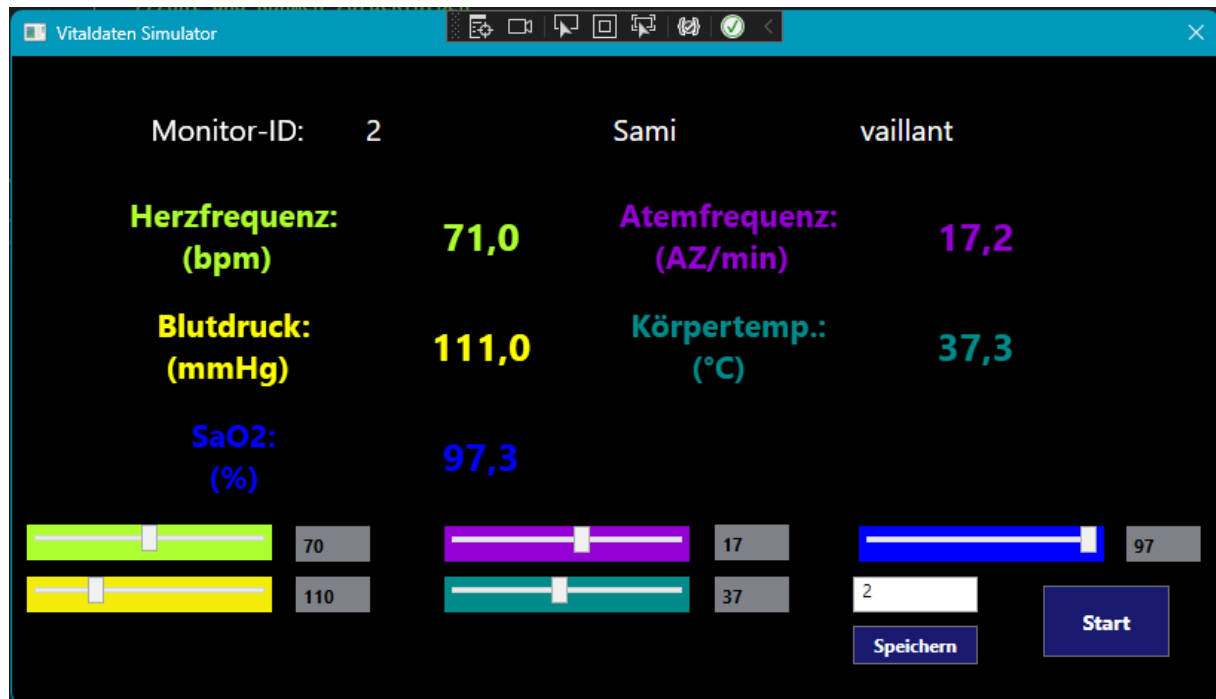
Abbildung N°4 - Administration UI



2. Vitaldatensimulator:

Der Vitaldatensimulator setzt sich aus zwei WPF-Usercontrol-Komponenten zusammen. Die erste obere Komponente ist die Vitaldaten-Schablone, die ebenfalls im Remote Monitor eingesetzt wird, und die zweite untere Komponente bilden die Slider, welche die Vitaldaten eines Patienten simulieren und anschließend über MQTT versenden. Die Vitaldaten-Schablone empfängt die versendeten Vitaldaten und aktualisiert die Anzeige entsprechend.

Abbildung N°5 - Vitaldaten Simulator UI



3. Remote-Monitor:

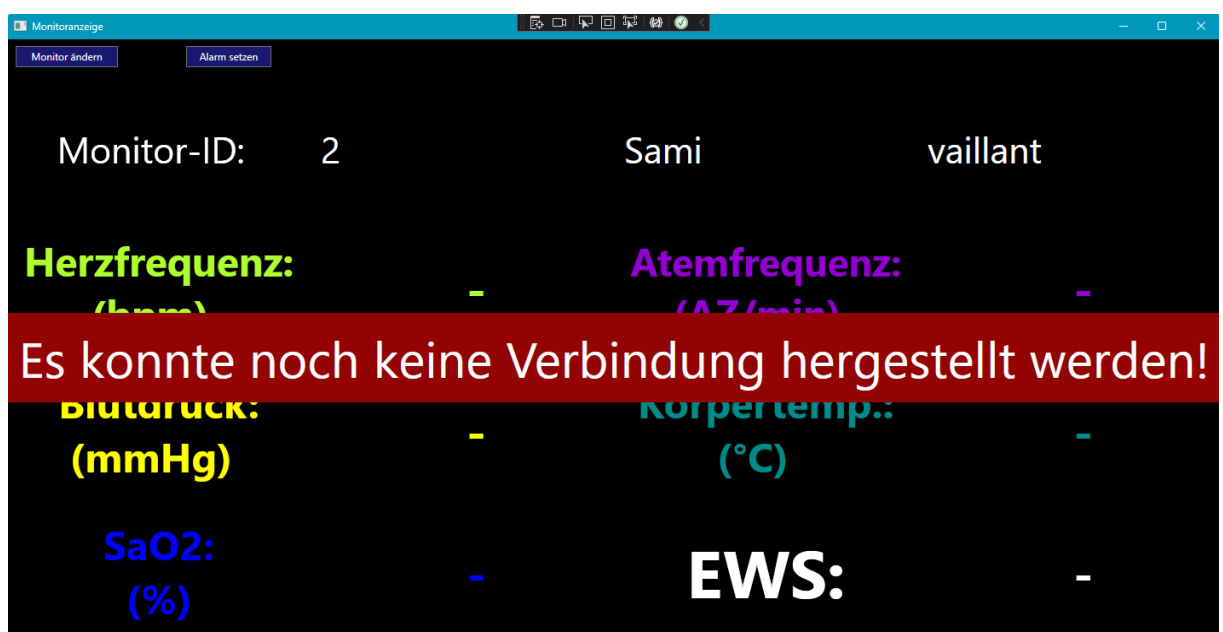
1. Der Startbildschirm zeigt eine Patienten-Liste zur Auswahl der Anzahl an Monitore zum Empfang der Daten an. Je nach Anzahl wird automatisch die Anzeige von einem, vier, neun oder sechzehn Monitoren gleichzeitig angepasst.

Abbildung N°6 - RemoteMonitor Willkommen UI



2. Die Anzeige des Remote Monitors bei der Auswahl eines einzelnen Monitors.

Abbildung N°7 - RemoteMonitor mit ein Patient-UI



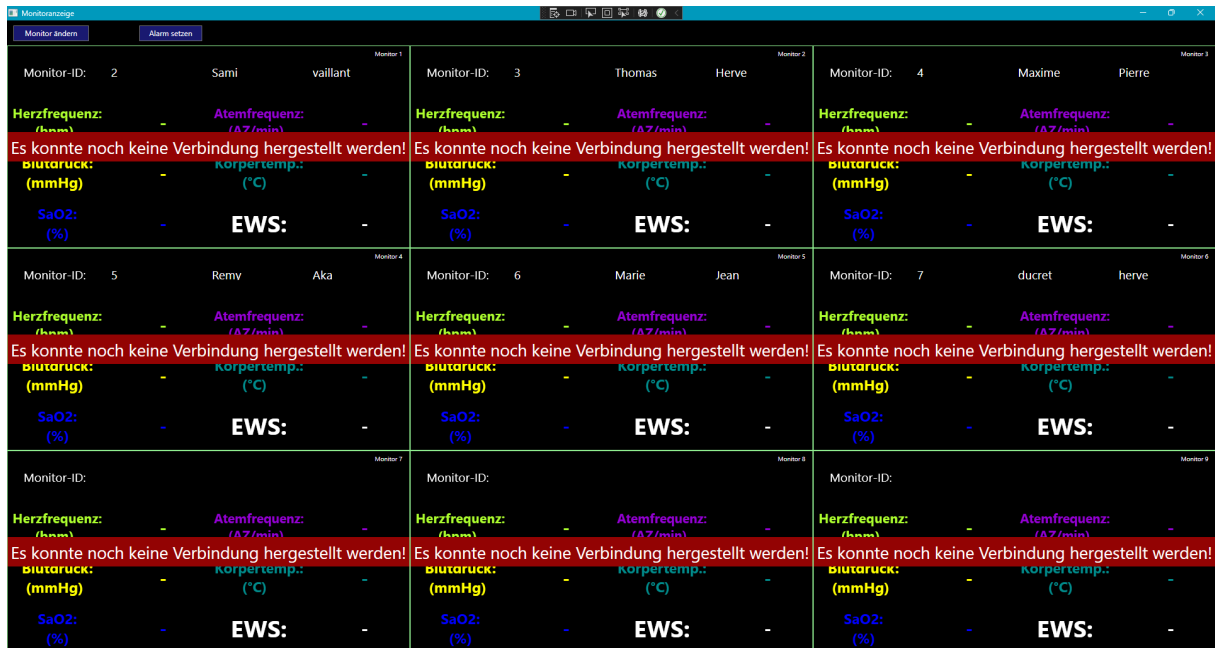
- Die Anzeige des Remote Monitors bei der Auswahl von einem bis zu vier Monitoren. Die Monitore sind jeweils nummeriert und mit einer Monitornummer in der rechten oberen Ecke versehen. Jeder Monitor zeigt ein anderes Szenario, das bei einer Verbindung mit einem Patienten-Monitor auftreten kann. Monitor 1 (links-oben) zeigt das Szenario, bevor eine Verbindung mit einem Patienten-Monitor hergestellt werden konnte. Die Meldung tritt 5 Sekunden nach dem Start des Remote Monitors auf. Monitor 2 (rechts oben) zeigt das Szenario bei einer bestehenden Verbindung mit einem Patienten-Monitor. Die Vitaldaten werden im Sekundentakt empfangen und angezeigt und dabei der EWS berechnet. Monitor 3 (links unten) zeigt den Fall, nachdem eine bestehende Verbindung getrennt wurde. Die Fehlermeldung wird ebenfalls 5 Sekunden, nachdem der letzte Wert empfangen wurde, auf dem betroffenen Monitor angezeigt. Monitor 4 (rechts unten) zeigt einen leeren Bildschirm, falls kein Patient hierfür ausgewählt wurde.

Abbildung N°8 - RemoteMonitor mit vier Patient UI

<div>Monitor-ID: 2 Sami vaillant</div> <div> <div>Herzfrequenz: (bpm) -</div> <div>Atemfrequenz: (A Z/min) -</div> <div>Blutdruck: (mmHg) -</div> <div>Körpertemp.: (°C) -</div> <div>SaO2: (%) -</div> <div>EWS: -</div> </div> <div>Es konnte noch keine Verbindung hergestellt werden!</div>	<div>Monitor-ID: 3 Thomas Herve</div> <div> <div>Herzfrequenz: (bpm) 68,0</div> <div>Atemfrequenz: (A Z/min) 17,5</div> <div>Blutdruck: (mmHg) 108,0</div> <div>Körpertemp.: (°C) 37,5</div> <div>SaO2: (%) 98,1</div> <div>EWS: 1</div> </div>
<div>Monitor-ID: 5 Remv Aka</div> <div> <div>Herzfrequenz: (bpm) 71,0</div> <div>Atemfrequenz: (A Z/min) 18,3</div> <div>Blutdruck: (mmHg) 111,0</div> <div>Körpertemp.: (°C) 37,3</div> <div>SaO2: (%) 98,2</div> <div>EWS: 0</div> </div> <div>Fehler! Netzwerkverbindung wurde getrennt</div>	<div>Monitor-ID:</div> <div> <div>Herzfrequenz: (bpm) -</div> <div>Atemfrequenz: (A Z/min) -</div> <div>Blutdruck: (mmHg) -</div> <div>Körpertemp.: (°C) -</div> <div>SaO2: (%) -</div> <div>EWS: -</div> </div> <div>Es konnte noch keine Verbindung hergestellt werden!</div>

4. Die Anzeige des Remote Monitors bei der Auswahl von fünf bis zu neun Monitoren.

Abbildung N°9 - RemoteMonitor mit neun Patient UI



5. Die Anzeige des Remote Monitors bei der Auswahl von zehn bis zu sechzehn Monitoren.

Abbildung N°10 - RemoteMonitor mit sechzehn Patient UI



6. Bei der Anzeige des Remote Monitors werden bei einer bestehenden Verbindung Werte empfangen. Für die Überwachung wurde die Möglichkeit hinzugefügt, Alarmgrenzen patientenindividuell zu setzen. Diese können mit der folgenden Oberfläche mit Default Werten gesetzt werden. Die Auswahl der Grenzen entspricht

dem Normalbereich der jeweiligen Vitalwerte. Diese können individuell eingestellt werden.

Abbildung N°11 - RemoteMonitor Alarm UI

	Minimalgrenze	Maximalgrenze
Blutdruck Min	90	Blutdruck Max 120
HF Min	60	HF Max 80
Temp Min	36	Temp Max 40
Atemfreq Min	11	Atemfreq Max 24
SpO2 Min	95	

Zurück Speichern

Wurden die Alarmgrenzen gesetzt, so werden die empfangenen Vitaldaten, die außerhalb der gesetzten Alarmgrenzen liegen, in rot gefärbt und mit einem roten Rahmen versehen (Im folgenden Beispiel ein Patient mit Fieber und Bluthochdruck). Hiernach können entsprechende Maßnahmen ergriffen werden.

Abbildung N°12 - RemoteMonitor ein Patient mit Alarm UI

Herzfrequenz: (bpm)	70,0	Atemfrequenz: (AZ/min)	17,9
Blutdruck: (mmHg)	172,0	Körpertemp.: (°C)	40,8
SaO2: (%)	98,8	EWS:	2

Monitor ändern Alarm setzen

5. Software-Verifikation

Die Funktionalität der Software wird durch die Durchführung von Systemtests sichergestellt. Hierfür wird eine Reihe von Testcases erstellt (siehe Anhang). Die Durchführung und Dokumentation der Tests wird mit Zephyr Scale realisiert.

Um die Funktionalität der Software zu testen, werden fünf Szenarien formuliert, die eine reale Situation simulieren. Die einzelnen Testcases orientieren sich an den Szenarien und sind so formuliert, dass sie alle Vorgänge in den Szenarien abdecken.

Insgesamt gibt es fünf Szenarien.

5.1 Testspezifikationen und Testergebnisse

Tests eines Computerprogramms dienen dazu, die Zuverlässigkeit einer Software zu überprüfen. Mit ihnen kann man überprüfen, ob das Programm wie erwartet funktioniert und keine Fehler enthält. Mit Tests kann auch nachgewiesen werden, dass das Programm die gewünschten Anforderungen erfüllt. Dies ist wichtig, wenn Software wie in unserem Fall ein Monitorverwaltungssystem für ein Krankenhaus erstellt wird.

Wir haben verschiedene Tests beschrieben und durchgeführt, um die Effektivität unseres Programms zu überprüfen. Wir haben die verschiedenen Tests in Jira über die Erweiterung Zephyr Scale verfasst.

Abbildung N°13 - Test Ergebnis

Testergebnisse (Fortschritt)

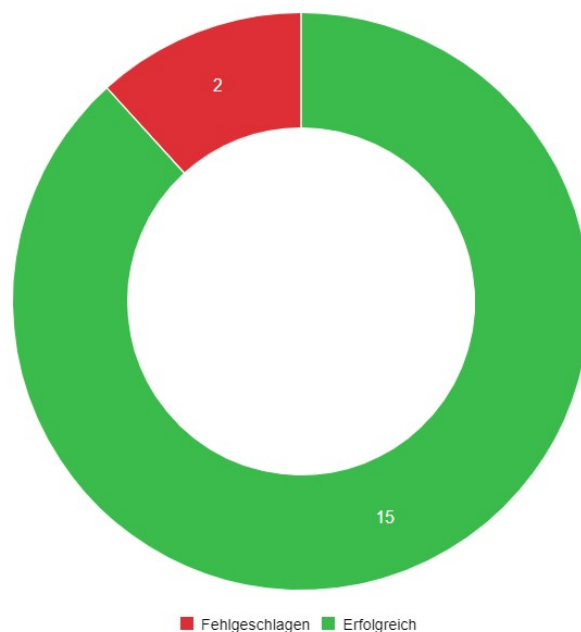


Abbildung N°14 - Detailliertes Testergebnis

Test Cycle Description						Test Execution Results	
Execution.Key	Execution.Assigned To	Test Case.Priority	Test Cycle.Key	Test Cycle.Name	Test Cycle.Owner	Test Case.Key	Execution.Result
PPG-E20	Laura Huber	High	PPG-R21	Netzwerkausfall	Remziye	PPG-T12	Pass
PPG-E21	thomas liebgott	High	PPG-R20	Monitoraustausch	Remziye	PPG-T11	Pass
PPG-E22	Yong-Chan Kwon	High	PPG-R19	Patient entlassen und hinzufügen	Remziye	PPG-T10	Pass
PPG-E26	Jana Tomic	High	PPG-R16	Vitaldatensimulator - Anzeige auf Remote Monitor	Remziye	PPG-T9	Pass
PPG-E28	Remziye	High	PPG-R26	Alarmierung	Remziye	PPG-T1	Pass
PPG-E29	Thomas liebgott	Normal	PPG-R27	Einen nicht existierenden Patienten löschen	Thomas liebgott	PPG-T14	Pass
PPG-E30	Thomas liebgott	Normal	PPG-R28	Einen Patienten hinzufügen	Thomas liebgott	PPG-T15	Pass
PPG-E31	Thomas liebgott	Normal	PPG-R29	Einen Patienten löschen	Thomas liebgott	PPG-T13	Pass
PPG-E32	Thomas liebgott	Normal	PPG-R30	Einen Patienten und einen Monitor zu verbinden und zu trennen	Thomas liebgott	PPG-T16	Pass
PPG-E33	Thomas liebgott	Normal	PPG-R31	Hinzufügen oder Entfernen eines Monitors	Thomas liebgott	PPG-T17	Pass
PPG-E34	Thomas liebgott	Normal	PPG-R32	Alarmierung mit mit einem nicht anwesenden Monitor	Thomas liebgott	PPG-T18	Fail
PPG-E35	Thomas liebgott	Normal	PPG-R33	Alle Patienten in Remote anzeigen	Thomas liebgott	PPG-T19	Pass
PPG-E36	Thomas liebgott	Normal	PPG-R34	Änderung des Patienten im Betrieb	Thomas liebgott	PPG-T20	Pass
PPG-E37	Thomas liebgott	Normal	PPG-R35	Falsche Daten versenden	Thomas liebgott	PPG-T21	Pass
PPG-E38	Thomas liebgott	Normal	PPG-R37	Daten von zwei verschiedenen "Vitaldaten Simulator" an denselben Monitor senden	Thomas liebgott	PPG-T22	Fail
PPG-E39	Remziye	Normal	PPG-R38	EWS Anzeige auf dem Remote Monitor	Remziye	PPG-T23	Pass
PPG-E40	Remziye	Normal	PPG-R39	ein Fehlerfenster geöffnet wird, das angibt, dass dieser Monitor bereits Daten sendet	Remziye	PPG-T24	Pass

Im nächsten Abschnitt werden die verschiedenen Ergebnisse erklärt, die erzielt wurden.

5.2 Verbleibende Bugs und Bewertung

5.2.a - Verschiedene Bugs vorhanden:

Im Laufe unserer Testspezifikationen wurden verschiedene Arten von Bugs entdeckt. Zunächst einmal gibt es Fehler, die zum Abbruch des Programms führen. Diese Fehler werden in der Regel dadurch verursacht, dass eine Ausnahme nicht implementiert wird. Wenn man z. B. ein Feld in einem Textfeld leer lässt, kann dies zum Abbruch des Programms führen. Durch das Implementieren einer Ausnahme ist es jedoch möglich, ein Fehlerfenster anzuzeigen, anstatt das Programm zu unterbrechen. Obwohl diese Fehler die ordnungsgemäße Funktion des Programms nicht gefährden, erfordern sie die besondere Aufmerksamkeit des Benutzers. Es handelt sich hierbei um einen Fehler unsererseits, der hätte behoben werden können, wenn wir diese Feinheit vor dem Einfrieren des Codes getestet hätten.

Dann gibt es noch Bugs, die das Programm nicht unterbrechen, aber seine Funktionsweise beeinträchtigen. Es ist z.B. möglich, dass von zwei verschiedenen Vitaldatensimulatoren unterschiedliche Daten an denselben Monitor gesendet werden. Die Lösung für dieses Problem wäre, die ID des ausgewählten Monitors zu überprüfen, um festzustellen, ob dieser bereits Daten empfängt. Wie bei dem zuvor erwähnten Fehler verhindert dies nicht die ordnungsgemäße Funktion des Programms, erfordert aber eine besondere Aufmerksamkeit des Benutzers hinsichtlich der Daten, die er eingibt.

Ein weiteres Problem, auf das wir gestoßen sind, bezieht sich schließlich auf die Anzeige der Daten von 16 Patienten auf leistungsschwachen Computern. Einige Personen in unserer Gruppe mit weniger leistungsfähigen Computern hatten Probleme mit der Anzeige der Daten von 16 Patienten.

5.2 b - Mögliche Verbesserungen:

Wie im vorherigen Abschnitt erwähnt, können verschiedene Fehler im weiteren Verlauf des Projekts behoben werden.

Wir wollten auch verschiedene mögliche Verbesserungen an unserem Programm aufzeigen, die das Nutzererlebnis verbessern könnten:

- Die Lesbarkeit der SpO2-Farbe könnte verbessert werden, indem man sie so verändert, dass sie einen stärkeren Kontrast zum Hintergrund aufweist, oder indem man eine andere Farbe verwendet, die leichter zu lesen ist.
- Es wäre interessant, eine Funktion einzurichten, mit der die Schriftgröße an die Anzahl der angezeigten Patienten angepasst werden kann. Dies würde den Nutzern das Lesen der Informationen erleichtern, insbesondere auf großen Bildschirmen, wo die Informationen schwer lesbar werden können, wenn 16 Patienten gleichzeitig angezeigt werden.
- Es wäre interessant, die Auswahl der Patienten und Monitoren im Verwaltungsteil intuitiver zu gestalten. Eine Idee könnte sein, neben den Namen der Patienten und Monitore in den Tabellen Kontrollkästchen einzufügen, die eine schnelle und effiziente Auswahl ermöglichen. Dies würde die Verwaltung von Patienten und Monitoren für den Benutzer einfacher und intuitiver machen.
- Sichern Sie zuvor angekreuzte Patienten bei der Bearbeitung von Wunscharten.
- Einige verwirrende Bezeichnungen wie "Monitor ID" und "Monitor Nummer" können geändert werden.

5.3 Traceability Matrix

In dieser Tabelle ist visualisiert, welche User Stories (siehe Software-Spezifikation) von welchen Testcases abgedeckt werden. In diesem Projekt hat man sich bei der Erstellung der Testcases an den Demo-Szenarien 1 bis 5 orientiert. Die einzelnen Testcases entsprechen den jeweiligen Demo-Szenarien. Eine ausführlichere Traceability-Matrix ist im Anhang "Traceability Matrix.pdf" zu finden.

Testcases	durch Testcase überprüfte User-Story
Vitaldatensimulator	R1, R2, R3, R4, R5, R7, A1, A2, V1, V2, V3
Alarmfunktion	R6, R9
Patient entlassen und hinzufügen	A1, A3, A4, A5
Monitortausch	A1, A3, A4, A5
Netzwerkausfall	R10, V4

6. Liste der SOUP

Name	Hersteller	Version	Zweck	Link
.NET Framework	Microsoft	4.7.2	Softwareanwendungen	https://support.microsoft.com/de-de/topic/microsoft-net-framework-4-8-f%C3%BCr-windows-10-version-1607-windows-10-version-1703-und-windows-server-2016-8ff8f85c-65f8-8fae-b85a-c556efce33fd
mqtt	BM, Cirrus Link Solutions	4.3.0	Netzwerkprotokoll	https://mqtt.org
PostgreSQL	PostgreSQL Global Development Group	14.1	Verwaltung der Datenbanken	https://www.postgresql.org/

B - Entwicklungsprozess

7. Software-Entwicklungsplan

Das Produkt „ViewMed“ wurde nach dem Vorgehensmodell „Scrum“ entwickelt. Auf diese Weise konnte die agile Softwareentwicklung umgesetzt werden. Die gesamte Software-Entwicklungsphase setzte sich aus drei Sprints zusammen. Die Aufgaben für jeden Sprint wurden am Anfang der jeweiligen Sprints festgelegt. Jede Woche fanden zwei Teammeetings statt, in denen Probleme, Aufgaben und Fortschritte besprochen wurden. Am Ende jedes Sprints fand eine Besprechung mit dem Kunden (Herrn Hastenteufel) statt, in denen Fortschritte und Ergebnisse präsentiert wurden.

8. Meilensteine/Sprints

Die Software-Entwicklungsphase des Produkts „ViewMed“ setzt sich aus 3 Sprints zusammen. Die ersten beiden Sprints umfassten eine Dauer von jeweils 3 Wochen. Der letzte Sprint hatte eine offizielle Dauer von 3 Wochen, allerdings wurden zusätzlich 2 Wochen der Vorlesungsfreien-Zeit zum zusätzlichen Entwickeln genutzt.

Für die Verwaltung der Sprints wird das Programm „Jira“ verwendet. Für jeden Sprint wird eine Roadmap erstellt, in der die zeitliche Planung festgelegt wird. Zusätzlich wird für jeden Sprint ein Backlog mit den jeweiligen User-Stories erstellt, die für den jeweiligen Sprint realisiert werden sollen. Die einzelnen User-Stories und Aufgaben, die im Sprint realisiert werden sollen, werden den

Entwicklern zugeteilt. Im Board werden die erstellten User-Stories/Aufgaben, je nach Status, in die passende Spalte gesetzt. Die sind „Aufgaben“, „In Arbeit“ und „Fertig“.

Am Ende jedes Sprints findet ein Review-Meeting mit Prof. Dr. Hastenteufel statt, in dem Ergebnisse, Probleme und das weitere Vorgehen besprochen werden. Nach dem Meeting gilt der Sprint als abgeschlossen.

9. Verwendung der Versionsverwaltung

Für die Versionsverwaltung wird Git eingesetzt. Für jede größere Aufgabe wird eine Branch angelegt. So stellt man sicher, dass im Fall von fehlerhaften Code auf ältere, funktionierende Versionen des Programms zugegriffen werden kann. Jedes Teammitglied bearbeitet die zugeordnete Aufgabe lokal auf der eigenen Aufgaben-Branch. Nach Fertigstellung des Codes wird dieser zunächst auf die lokale main gepusht, um die Funktionalität zu gewährleisten. Anschließend wird der funktionstüchtige Code auf die eigentliche main gepusht. Jedes Teammitglied kann auf die jeweiligen Branches zugreifen und ggf. bei Problemen die anderen Teammitglieder unterstützen.

10. Team Meetings

Jede Woche finden zwei Teammeetings statt. Freitags trifft sich das Team vor Ort in der Hochschule. Mittwochs findet ein Hybrid-Meeting statt, in dem ein Teil des Teams vor Ort in der Hochschule ist und ein Teil des Teams sich online hinzuschaltet. Diese zwei Meetings sind feste Termine, die eingehalten werden. Zusätzlich zu den festen Terminen finden Online-Meetings bei spontanen Problemen oder Fragen statt. Die Online-Meetings werden über die Plattform „Discord“ gehalten.

In den Meetings werden die jeweiligen Fortschritte präsentiert, oder anfallende Fragen sowie Probleme gelöst. Jedes Teammitglied ist mit seinem Aufgabenbereich vertraut und erklärt diesen den anderen Teammitgliedern im Meeting. Bei Problemen wird Hilfe angeboten, oder die Aufgabenverteilung so aufgeteilt, dass intensiv an Problemen gearbeitet werden kann.

11. Dokumentation und Erhebung von SW-Anforderungen

Am Anfang des Projekts wurden die SW-Anforderungen als Stakeholder-Anforderungen angegeben. Aus diesen wurden User Stories (siehe Tabelle in Unterkapitel 2 „Software-Spezifikation“) definiert. Zu Beginn jedes Sprints wurden die passenden User-Stories in Jira dem jeweiligen Sprint zugeteilt. Im Laufe der Entwicklung wurden die User-Stories des jeweiligen Sprints, je nach Bedarf, angepasst, umformuliert oder ergänzt.

12. Testplan


Um die Funktionalität der Software sicherzustellen, wurden Testcases formuliert und durchgeführt. Die Durchführung und Dokumentation der Tests wird mit Zephyr Scale in Jira durchgeführt. Die Testcases sind so formuliert, dass sie bestimmte Szenarien abdecken. Es gibt fünf Szenarien, die eine realistische Situation simulieren und alle erwarteten Funktionalitäten der Software abfragen. Für jedes dieser Szenarien sind Testcases formuliert.

Name	Zuständigkeit Tests
Remziye	Formulierung/Erstellung der Testcases, Testdurchführung "Alarmfunktion"
Jana	Formulierung/Erstellung der Testcases, Testdurchführung "Vitaldatensimulator"
Thomas	Testdurchführung "Monitortausch",
Laura	Testdurchführung "Netzwerkausfall",
Chan	Testdurchführung "Patient entlassen",

Die fünf Szenarien in Zephyr:

Abbildung N°15 - Screenshot unserer Testoberfläche auf zephyr




PME-Projekt-Grün / Testfälle / PPG-T1 (2.0)

Alarmierung
Zurück
Speichern
Neue Version
2.0 ...

Details
Schritte
Ausführung
Rückverfolgbarkeit
Anhänge
Kommentare
Änderungshistorie

Typ: Schritte

Schritte

	SCHRITT	TESTDATEN	ERWARTETES ERGEBNIS
1	Starte den Vitaldaten - Simulator		Simulator mit Anzeige von Vitalparameter. Es sind keine Vitaldaten vorhanden.
2	Gebe eine Monitornummer unten rechts in der Spalte ein und klicke auf Speichern.	Klicken Sie hier, um Testdaten einzugeben	Oben in der Liste sind die Patientendaten(Vor- und Nachname) mit der dazugehörigen Monitor - ID zu sehen.
3	Klicke dann auf Starten.	Klicken Sie hier, um Testdaten einzugeben	Vitaldaten werden angezeigt.
4	Starte den Remote Monitor	Klicken Sie hier, um Testdaten einzugeben	Der Remote - Monitor wird gestartet.

Remziye und Jana haben die Testcases und die durchzuführenden Schritte in Zephyr formuliert und dokumentiert. Die restlichen Teammitglieder haben diese Testcases durchgeführt unter Anweisung der formulierten Schritte für jedes Szenario. Ursprünglich sollten die Testdesigner (Jana und Remi) nur die Tests formulieren und erstellen und die restlichen Teammitglieder das Testen übernehmen. So stellt man sicher, dass es eine Trennung zwischen Testdesign und Testdurchführung gibt. Aus Zeitgründen wurden zwei Testcases auch von den Testdesignern durchgeführt.

Die Tests werden in der Woche vor der finalen Demonstration des Projekts durchgeführt. So wird sichergestellt, welche Komponenten funktionieren und welche Komponenten noch eventuelle Bugs aufweisen.

13. Mitarbeiter und Rollen/Verantwortlichkeiten

Die Entwicklung des Produktes findet nach dem „Scrum“-Modell statt. Nach diesem Modell werden im Scrum-Team folgende Rollen verteilt:

- Product Owner
- Scrum Master
- Entwickler

Als Product Owner wird Laura Huber festgelegt. Die Aufgabe des Scrum Masters wird Jana Tomic zugeordnet. Chan, Thomas und Remi sind Entwickler. Der Product Owner und der Scrum Master sind ebenfalls Entwickler in diesem Projekt. Jedes Teammitglied hat seinen Aufgabenbereich und ist in diesem der Experte. Bei Problemen, oder großen Aufgaben erhalten die jeweiligen Entwickler Unterstützung von den anderen Mitgliedern des Teams

Die Tabelle zeigt die Aufgabenverteilung der einzelnen Teammitglieder:

Name	Rolle	Systemkomponente
Chan	Entwickler	Remote Monitor
Jana	Scrum Master, Entwickler	Vitaldatensimulator
Laura	Product Owner, Entwickler	Vitaldatensimulator
Remziye	Entwickler	Remote Monitor
Thomas	Entwickler	Administration, Remote Monitor, Vitaldaten Simulator

Anmerkung: Im Laufe des Projekts wurden die zu Anfang bestimmten, starren Aufgabenverteilungen immer mehr verworfen. Die Aufgabenverteilung wurde dynamisch, um möglichst schnell und effizient auf Planänderungen reagieren zu können. So können Krankheitsausfälle oder Zeitmangel durch andere Fächer gut kompensiert werden. Zusammenfassend wurde darauf geachtet, dass die Aufgabenverteilung ausgewogen auf alle Gruppenmitglieder verteilt ist. Haben einige Teammitglieder mehr programmiert, so haben dafür andere Mitglieder am Bericht gearbeitet, oder Tests konzipiert und formuliert. Bei größeren Problemen und Hürden, die beim Programmieren aufgetreten sind, wurden Teammeetings einberufen und gemeinsam an einer Lösung gesucht sowie Meetings in der Hochschule veranstaltet.

14. Liste der eingesetzten Tools

In der folgenden Tabelle sind alle Programme aufgeführt, die während der Entwicklung des Projekts eingesetzt wurden

Name	Hersteller	Version	Zweck	Link
Jira	Atlassian	9.3.0	Organisation, Verwaltung	https://www.atlassian.com/de/software/jira
Discord	Discord Inc.	1.0.9008	Kommunikation, Online-Meetings	https://discord.com/
Visual Studio	Microsoft Corporation	16.11.20	Programmierung der Software	https://visualstudio.microsoft.com/de
Git	Junio Hamano,	2.39.0	Versionsverwaltung	https://git-scm.com/

	Shawn Pearce, Linus Torvalds			
Cisco AnyConnect Secure Mobility Client	Cisco	4.10.06079	VPN Verbindung mit Hochschule	https://www.cisco.com/c/de_at/product/s/security/anyconnect-secure-mobility-client/index.html
Microsoft Word	Microsoft Corporation	22.11	Technische Dokumentation	https://www.micro-soft.com/de-de/microsoft-365/p/word
Google Drive	Google	64.0.4.0	Ablage von Dokumenten	https://accounts.google.com/AccountChooser/signinchooser?service=writely&flowName=GlifWebSignIn&flowEntry=AccountChooser
Zephyr Scale	Atlassian	9.3.0	Testdurchführung und Dokumentation	https://marketplace.atlassian.com/apps/1213259/zephyr-scale-test-management-for-jira?tab=overview&hosting=cloud

Anhang

1. Traceability Matrix.pdf
2. Testdokumentation_Anhang.xlsx
3. Testerklärung und Ergebnis