Semesterarbeit Programmentwurf

„Elektronische Patientenakte“

Gruppe 04

im Rahmen der Vorlesung **Informatik 3**

Dozent: Prof. Dr. Ing. Alexander Grüning

Studiengang Elektrotechnik – Informations- und Nachrichtentechnik

Duale Hochschule Baden – Württemberg Ravensburg, Campus Friedrichshafen

Von:

Justin Bauer-Chen	203723	Airbus Defence and Space,	Immenstadt
Alexander Graf	1471521	Hensoldt,	Ulm
Jacob Jäger	8790687	Hensoldt,	Ulm
David Rösch	5281498	Primion Technology,	Stetten a.k.M.
Johannes Winter	1937294	Hensoldt,	Oberkochen

Abgabedatum:	Sonntag, 29. Dezember 2024
Bearbeitungszeitraum:	14.10.2024 – 29.12.2024

Kurs: TEN-23

KI-Hinweis

Bei der Umsetzung des Projekts wurde gezielt KI-basierte Unterstützung eingesetzt, um bestimmte Aufgaben zu erleichtern und die Effizienz zu steigern. Dazu gehörten unter anderem die Unterstützung bei der Code-Optimierung, das Generieren von Vorschlägen für Algorithmen sowie Hilfestellungen bei der Fehlersuche. Der Einsatz von KI diente dabei als ergänzendes Werkzeug, während die inhaltliche Planung und Umsetzung weiterhin im Team erfolgten.

Inhaltsverzeichnis

1. Einleitung

Da beim Erlernen einer Programmiersprache und der Objektorientierten Programmierung der Fokus zunächst auf dem grundlegenden Verständnis liegt, kommt der praktische Nutzen eines Übungscodes oft erst einmal zu kurz. In den ersten beiden Semestern wurde vor allem die Basis der C-Programmierung vermittelt, um ein solides Fundament zu schaffen. Hier standen grundlegende Programmierkonzepte und deren Anwendung im Vordergrund.

Im dritten Semester lag der Schwerpunkt darauf, das erlernte in sinnvolleren und praxisorientierten Anwendungen einzusetzen und dabei Erfahrung zu sammeln. Dabei wurde mit der Aufgabenstellung gezielt darauf hingearbeitet, theoretisches Wissen in die Praxis zu übertragen und so die Programmierfähigkeiten auf eine neue Ebene zu bringen. Dies ermöglichte es, den theoretischen Lernstoff aus den vorhergegangenen Semestern mit praxisorientierten Aufgaben zu verknüpfen, um die Programmierung und deren Konzepte besser zu verstehen. Das Projekt fand im Rahmen einer Gruppenarbeit statt, sodass zu den Inhaltlichen Aspekten der Aufgabe, auch das Arbeitsumfeld sehr nah an der späteren Realität im Berufsalltag war.

Ziel des Projektes war das Entwickeln einer Desktop-Anwendung zur Patienten-datenverwaltung. Hierbei wurden vom Dozenten Mindestanforderungen vorgegeben. Im Hinblick auf eine bessere Bewertung des Gesamtprojekts stellte man der Gruppe allerdings frei, die Anwendung um vorgeschlagene oder eigens erdachte Funktionen zu erweitern. Weitere Details zur Aufgabenstellung entnehmen sie bitte der selbigen. Im Verlauf der Entwicklung wurden von der Arbeitsgruppe folgende Erweiterungen entworfen und umgesetzt:

- *Umstellung der Datenverwaltung von einer csv-Datei in eine SQLite Datenbank*
- *Hinzufügen von neuen Datensätzen (Patienten)*
- *Mehrbenutzerfähigkeit über eine Anmeldemaske inklusive Schutz der Anmeldedaten*
- *Decodierung der ICD-Code*
- *Verbesserung der Benutzeroberfläche (Dark-Mode, Tooltips, Dropdown-Eingabe etc.)*
- *Programmstabilität und Abfangen von unerwarteten Nutzereingaben*

Die Umsetzung des Projektes erfolgte nach Vorgabe in der qt-Designer Entwicklungsumgebung und wurde in C++ programmiert. Für die Versionsverwaltung wurde Sourcetree und GIT verwendet.

Die Aufgaben wurden gemeinsam geplant und teils gemeinsam, teils eigenständig bearbeitet. Während ein Teil des Teams direkt zu Beginn mit der Erstellung der Grundfunktionen und der Versionsverwaltung startete, legten andere Teammitglieder den Umfang des Programms fest und erarbeiteten Konzepte, wie die Umsetzung erfolgen könnte. Diese wurden anschließend in der Gruppe diskutiert, bei Bedarf angepasst und umgesetzt. Die Verteilung der Aufgaben wurde so gestaltet, dass jedes Teammitglied seine Stärken einbringen konnte, um das bestmögliche Ergebnis zu erzielen. Ein kontinuierlicher Austausch und gegenseitige Unterstützung im Team sorgten für eine effiziente Arbeitsweise und trugen entscheidend zum Erfolg des Projektes bei.

In den folgenden Abschnitten gibt jedes Teammitglied Einblicke in Implementierung und Umsetzung der Aufgaben. Der Schwerpunkt der liegt hierbei auf den Methoden, der Funktionsweise und den Programmfunktionen, weniger auf der Rekonstruktion des gesamten Projektverlaufs.

2. Einzelberichte

2.1. Bericht Justin Bauer Chen

Backend für Benutzer Verwaltung

Die Klasse „user“ verwaltet alle Benutzer um ein an- und abmelden mit einem Benutzernamen und Passwort zu ermöglichen. Die Anmeldedaten werden in einer eigenen Tabelle „Users“ in der Datenbank (=DB) gespeichert. Diese hat folgende Attribute:

User_ID (integer):

Diese Interne ID dient als Primärschlüssel um jeden Datensatz eindeutig zu identifizieren und wird automatisch inkrementell ab 1 um 1 hochgezählt. Das Limit an Datensätzen und damit an Benutzern beträgt hier bei einer Speicherkapazität von 8 Byte pro integer bei $(2^8)^8$ -1 = 2^{64} -1 Benutzern oder 1,8E10 Milliarden. Das Limit liegt also weit über der Anzahl der Bevölkerung von Deutschland (ca. 84 Millionen in 2024). Es würde sehr lange dauern bis alle IDs ausgeschöpft sind.

Login_Name (text):

Speichert den Benutzernamen (=BN) der zum Anmelden benötigt wird und darf max. 32 Zeichen lang sein. Es könnte auch als Primärschlüssel dienen, da bei unserer Implementierung der BN eindeutig sein muss (keine 2 gleiche BNs). Das wäre aber unsauber da der BN vom Nutzer in Zukunft geändert werden könnte.

PW_Hashed (text):

Speichert das Passwort als Hash für die Anmeldung

Salt (text):

Wird intern benötigt beim Anmelden und zum Hashen des Passwortes.

Rechte (integer)

Die Rechte auf die Patientenakten werden als Zahl gespeichert.
Dabei entspricht die:

0 einen Fehler bzw. ungültig

1 Lese- und Schreibrechte (z.B. Arzt)

2 nur Lese- Rechte (z.B. Patient)

Hashen des Passworts

Um das Passwort sicher zu speichern, also gegen Hacker Angriffe zu schützen, wird es nicht als Klartext in der Datenbank hinterlegt. Es wurde ein SHA-256 Hash Algorithmus benutzt um das Passwort mit einem Salt in einen Hash umzuwandeln. Dieser wird in Hexadezimalzahlen in die Datenbank abspeichert. Anders als bei der Verschlüsselung kann dies nicht rückgängig gemacht werden.

Der Salt schützt gegen Rainbow Table Attacken, also eine Art pregenerierte Tabelle mit Hashen und ihren zugehörigen Passwörtern. Dies erschwert deutlich das Knacken des Passwortes, da für jedes Passwort eine Brut-Force Attacke von vorne starten müsste, da ein anderer Salt verwendet wird. Damit muss der Salt nicht zwingend geschützt sein. Zur Vereinfachung entspricht der Salt dem BN da dieser bereits eindeutig ist. Besser wäre aber ein zufälliger Wert der eindeutig ist bzw. nur einmal im Datensatz auftaucht.

Der Hash ist als hex mit 64 chars darstellbar aber speicher ineffizient (1 char ≙ mind. 1 Byte und 1 HEX ≙ 1 Nibble). Die 256 bezieht sich somit auf die Länge des Hashes (64 Hex Zahlen ≙ 32 Byte ≙ 256 Bit). Somit sollte das Passwort+Salt eine maximale Länge im Speicher von 32 Bytes haben, also 32 Zeichen. Es wurde einen einfache Salt funktion implementiert. Um ein möglichst langes Passwort zu haben wurde dafür 20 Zeichen für das Passwort und 12 Zeichen für den Salt reserviert (20+12=32).

Fehlerbehandlung

Fehler werden entweder intern als 0 (! nicht Null !) oder String zurückgegeben. Damit kann die Fehlermeldung direkt weitergeben und dem Benutzer angezeigt werden.

Funktionen

checkPW:

> verifiziert den Benutzer. Dabei wird das eingegebene Passwort mit dem Salt aus der DB gehasht und mit dem alten Hash aus der DB verglichen. Gleichzeitig wird die Berechtigung als Zahl zurückgegeben. Die Rückgabe dient gleichzeitig als Autorisierung: bei einem falschen Passwort wird eine 0 (Fehler) zurückgeben.

salt_generator sollte in Zukunft einen zufälligen Wert erzeugen. Bis jetzt wird nur der Benutzername wieder zurückgegeben.

encrypt_pw hasht das Passwort.

inserUserDB erstellt einen neuen Datensatz in der Tabelle Users bzw. fügt einen neuen Benutzer hinzu.

changePW ändert das Passwort.

2.2. Bericht Alexander Graf

Klasse io_data

Die Klasse „io_data“ dient der sicheren Konvertierung, Ein- und Ausgabe der Patientendaten. Im Konstruktor werden Membervariablen definiert, die patientenbezogene Informationen, wie ID, Name, Geburtsdatum, Geschlecht, Kontaktdaten und medizinische Details umfassen.

Zudem enthält diese eine Methode zur Altersberechnung anhand des Geburtsdatums des Patienten und Funktionen zur bidirektionalen Konvertierung des Datums zwischen QDate und QString.

Des Weiteren verwaltet Sie die Datenflüsse für der Dateneinabe und -ausgabe. Dies erfolgt mithilfe der Klasse „database“, die die Datenbankanbindung realisiert und Datensätze von der Datenbank liest, bzw. diese abspeichert. Die Klasse „database“, wird von Jacob Jäger genauer beschrieben.

Zudem sind die Funktionalitäten zum Einlesen von Patientendaten aus CSV-Dateien und Exportierung der gespeicherten Patientendaten in eine CSV-Datei enthalten.

returnAge Methode:

Diese Methode berechnet das Alter eines Patienten basierend auf dem Geburtsdatum und dem aktuellen Datum und gibt dieses als int zurück. Sie wandelt den QString der Membervariable „geburt“ über einen Standardstring in einen stream um und zerlegt diesen an den enthaltenen Punkten in drei Variablen „int tag“, „int monat“ und „int jahr“. Anschließend wird die Systemzeit geladen und durch Subtraktion des Jahres das Alter berechnet. Zudem wird über eine if-Abfrage das Alter korrigiert, falls das aktuelle Datum vor dem Geburtstag liegt um das korrekte tagesaktuelle Alter zurückzugeben.

convertQStringToQDate Methode:

Diese prüft den übergebenen String auf Gültigkeit (Format „dd.MM.yyyy“) und gibt bei Erfolg das Datum als QDate zurück. Ansonsten wird ein ungültiges QDate sowie eine qWarning zurückgegeben.

convertQDateToQString Methode:

Dies ist die inverse Funktion zur convertQStringToQDate Methode und wandelt das QDate in einen QString im Format „dd.MM.yyyy“ zurück.

CSVeinlesen Methode:

CSVeinlesen ist mittels Try-Catch-Blöcken realisiert, um die Programmstabilität sicherzustellen. Sie öffnet die ausgewählte CSV-Datei, liest dessen Patientendaten zeilenweise ein und validiert diese mittels Regex, bevor die Methode insertPatient der Klasse „database“ aufgerufen wird. Nach dem vollständigen Einlesen der Datei, wird diese wieder geschlossen, um Speicherzugriffsproblemen, Programmstabilitätseinbrüche und Bugs vorzubeugen.

CSVerstellen Methode:

Hier werden die die Patientendaten aus der Datenbank in eine CSV-Datei exportiert. Diese Methode erstellt (bzw. überschreibt) die CSV-Datei und fügt in dessen erste Zeile die Spaltenüberschrift der Patientendaten ein. Anschließend werden die Patientendatensätze

der Datenbank nacheinander durchgegangen und in die CSV-Datei geschrieben, bevor die Datei wieder geschlossen wird.

Um die Stabilität des Programms zu gewährleisten, ist diese Methode ebenfalls in einen Try-Catch-Block eingebettet. Dies stellt sicher, dass mögliche Fehler durch Dateioperationen auf Systemebene, die durch Systeminteraktionen oder das Verhalten anderer im Hintergrund laufender Anwendungen entstehen können, abgefangen und behandelt werden.

Widgetverbindung, Implementierung von Optimierungen und Anpassungen

Im mainwindow.cpp werden durch connect-Befehle die Signale mit den Slots (auszuführenden Methoden) verbunden, um beispielsweise beim Auswählen eines Patienten in der Tabelle die gesammelten Daten des Patientendatensatzes im Ausgabefenster (TextEdit - Hauptfenster rechts unten) zu aktualisieren. Die Slot Methode wird auch nach dem Bearbeiten eines Datensatzes aufgerufen, um die Ausgabe im Hauptfenster zu aktualisieren.

Die „on_..._clicked()“ Methoden werden ausgeführt, wenn die verbundenen Schaltflächen betätigt werden. So wird zum Beispiel beim Speichern Knopf (Disketten Symbol) über den QFileDialog der Dateipfad für die speichern Funktionalität eingelesen. Beim Aktivieren der Öffnen Schaltfläche wird der ausgewählte Dateipfad an die Methode „CSVerstellen“ der Klasse „io_data“ übergeben.

Eine besondere Herausforderung war die Implementierung der Klassenübergreifenden Zeiger, der doppeltes öffnen von „Datensatz bearbeiten“ realisieret, um mehrmals denselben Datensatz nicht gleichzeitig bearbeiten zu können. Hierbei sollten dennoch parallel neue Datensätze angelegt werden können. Hierfür wurde eine Liste von Zeiger angelegt, die eine Maximalanzahl an „Datensatz hinzufügen“ Fenster zulässt. Diese ist aktuell auf fünf „Datensatz hinzufügen“ Fenster implementiert.

Wird im „Datensatz anzeigen“ Fenster der Button „Datensatz bearbeiten“ ausgewählt, wird geprüft, ob der Zeiger auf „Datensatz bearbeiten“ ein Nullzeiger ist oder bereits ein „Datensatz anzeigen“ Fenster offen ist. Dieser Zeiger wird auch bei der Schaltfläche zum Bearbeiten eines Datensatzes abgefragt und gesetzt, bzw. beim Schließen des Fensters wieder auf einen Nullzeiger zurückgesetzt. Durch Aufruf des Destruktors wird dafür gesorgt, dass die Objekte beim Schließen der Fenster korrekt zerstört werden und die Zeiger zu Nullzeiger werden.

Für die „Nutzer anlegen“ Klasse wurde der Zeiger als Doppelzeiger angelegt und übergeben, da damit das „mainwindow“ in der „Nutzer anlegen“ Klasse nicht inkludiert werden muss. Somit läuft die Anwendung stabil, handhabt und unterdrückt Fehler.

Zudem wurden sämtliche Widgets auf „Grid Layout“ umgestellt, damit auch beim Vergrößern des Fensters der Fensterinhalt flexibel mitskaliert wird. Die Scrollbalken, wie z. B. im „Datensatz bearbeiten“ Fenster, werden dadurch ebenfalls skaliert und können sogar verschwinden.

Am Ende wurde die Anwendung noch mit QMessageBoxen abgerundet. Diese geben Informationen über die Anzahl an Zeilen beim CSV einlesen, sowie Warnungen bei unzulässigen Operationen des Benutzers zurück. So beispielsweise bei dem Versuch eines doppelten Aufrufs für Datensatz bearbeiten, anzeigen oder Nutzer anlegen.

2.3. Bericht Jacob Jäger

Datenbank Klasse

Die Datenbank Klasse dient zur Erstellung und Verwaltung von Daten innerhalb der Tabellen Patienten und ICD- Codes. Um die benötigten Bibliotheken verwenden zu können muss das Sql Modul in der CMakeLists- Datei mit den anderen verknüpft werden. Zudem wird dort die SQLite Datenbank von den Source Files in den build Ordner mithilfe des configure_file Befehls kopiert. Dies hat den Vorteil, dass man so ungewollte Änderungen an der Datenbank einfach rückgängig machen kann, indem man dem build Ordner löscht und dann das Projekt neu erstellt.

Im Konstruktor der Klasse wird die Datenbank geöffnet. Das Datenbankobjekt wird dabei ganz am Anfang in der main Methode erstellt. Wenn die Datenbank nicht erfolgreich geöffnet wurde, wird dabei noch kein Fehler angezeigt, erst wenn man tatsächlich eine Datenbankanfrage durchführt. Im Destruktor wird die Datenbank wieder geschlossen.

insertPatient Methode:

> In dieser Methode wird ein neuer Patient zu der Tabelle Patienten hinzugefügt. Dies geschieht mit prepared statements um SQL- Injections vorzubeugen. Wenn es bei der Datenbankanfrage einen Fehler gibt wird ein runtime error geworfen.

editPatient Methode:

> Hier können die Daten eines bestehenden Patienten verändert werden. Die Methode bekommt als Parameter ein Objekt der Klasse io_data mit den aktualisierten Daten des Patienten. Es werden wieder prepared statements verwendet und bei Fehlern ein runtime error geworfen.

getPatientbyColumn Methode:

> Diese Methode findet alle Patienten in einer bestimmten Spalte, deren Daten mit dem anderen Parameter übereinstimmen. Es wird ein Vektor mit allen gefundenen Patienten zurückgegeben. Wenn die gesuchte Spalte die Patienten-ID ist werden keine substrings durchsucht, sonst schon.

getICD_Code_Information Methode:

> Hier können die Daten zu einem dazugehörigen ICD- Code abgerufen werden. Die Methode verwendet wieder prepared statements und es wird ein Vektor mit den gefundenen Daten zurückgegeben.

Backend für Datensatz anzeigen, Datensatz bearbeiten und Datensatz erstellen

In der Klasse Datensatz anzeigen wird im Konstruktor das Datensatz anzeigen UI-File ausgefüllt. Hier werden neben den Informationen aus der Patienten Tabelle auch die ICD-Code Informationen und das Alter des Patienten angezeigt. Von hier aus kommt mach auch auf das Patient bearbeiten Fenster. Um die Berechtigung des Nutzers zu überprüfen, wird hier auch das mainwindow Attribut gesetzt.

Die Klasse Datensatz bearbeiten dient zur Bearbeitung eines Patienten, sowie der Erstellung. Zur Unterscheidung wird im Konstruktor entweder -1 für die Erstellung oder die

tatsächliche ID, um einen Patienten zu bearbeiten, übergeben. Das UI-File wird dann einfach nicht ausgefüllt oder mit den Informationen des Patienten mit der jeweiligen ID. Wenn dann der Speicher- Button geklickt wird, werden die Informationen aus dem UI- File in das erstellte io_data Objekt gespeichert. Dann wird die jeweilige Datenbank Anfrage durchgeführt. Alle Ausnahmen werden dabei in der mainwindow Klasse gefangen.

Backend für Patientensuche und Filterfunktion

Wenn man die Enter- Taste oder auf die Lupe drückt wird die getPatientbyColumn Methode der Datenbank Klasse aufgerufen und es wird nach der Spalte gesucht, die mit dem Filter zuvor ausgewählt wurde. Die Methode geht dann durch alle Patienten in dem zurückgegebenen Vektor und fügt sie zu der Tabelle hinzu. Wenn die ID ausgewählt wird, dann wird nicht nach Sub Strings gesucht, sonst schon. Die Filterfunktion bei keiner Auswahl die Spalte PatientID ausgewählt-

Berechtigungssystem

Es gibt 3 Berechtigungen: Lese-, Schreib- und Admin- Berechtigung. Der user wird beim Login dynamisch erstellt und beim Schließen des Main Window wieder gelöscht. Bevor man ein Ui-File öffnen kann wird die Berechtigung geprüft, die durch ein Attribut in der user Klasse dargestellt werden. Nur ein Admin kann neue user erstellen und mit Leseberechtigung kann man nur die Patienten als CSV- Datei exportieren und Patienten anzeigen.

Darkmode Umstellung in allen offenen Fenstern

Das Style- Sheet wird bei dem öffnen jeder Datei gesetzt. Wenn man den Modus ändert, während andere Fenster geöffnet sind, dann wird durch alle Fenster iteriert und alle werden automatisch geändert.

Generelle Organisation

Dazu gehören Überlegungen zur Struktur des Programms, Aufgabenverteilung, Zusammenführen von Zweigen und Behebung von Merge- Konflikten.

2.4. Bericht David Rösch

Die erste Aufgabe bestand darin, den Rahmen für das Programm abzustecken und Konzepte zur Umsetzung von bestimmten Funktionen zu entwickeln. Hierbei half ein Besuch beim Arzt im Oktober 2024, bei welchem sich auf Nachfrage die Gelegenheit bat, einen Blick in die Praxisinterne Patientendatenverwaltung zu werfen. So konnten sowohl designtechnische Inspirationen als auch Funktionen der Software erfasst werden.

Mithilfe dieser sehr Praxisnahen Recherche war es möglich, auf Blatt ein erstes Bild der Oberfläche zu zeichnen, an dem man sich entlang hangeln konnte. Auch diverse Erweiterungsfunktionen konnten so vorgeschlagen und Diskutiert werden.

Beispielsweise stand im Raum, einen Lesezugriff für Patienten bereitzustellen, der über die Abfrage von Datensatznummer und Geburtsdatum eine schreibgeschützte Einsicht in den eigenen Datensatz gewährt. Allerdings wurde dieses Konzept nicht umgesetzt.

Die Umsetzung weiterer Funktionen, wie dem Hinzufügen und bearbeiten von Patientendaten folgte dem Konzept der Multiwindow-Anwendung. Das heißt, jede Funktion wird mit einem sich neu öffnenden Fenster repräsentiert. Die Erstellung dieser Fenster stellte meine Hauptaufgabe dar. Das Fenster zum Bearbeiten von Datensätzen besteht aus mehreren Eingabefeldern denen jeweils der Entsprechende Teil eines Datensatzes zugeordnet wird.

Sowohl die Erstellung des Fensters als auch die Gestaltung des anzuzeigenden Inhalts kann vollständig grafisch im Designer erfolgen. Qt bietet die Möglichkeit, die GUI ohne Code mithilfe eines CAD-Editors zu entwerfen. Dabei können verschiedene Ein- und Ausgabewidgets frei im festgelegten Fensterbereich platziert und angepasst werden. Das Fenster wird also nicht programmiert, sondern gezeichnet. Jedes Widget wird im automatisch generierten Code mit einem Slot verknüpft. Dieser Slot enthält den Code, der ausgeführt wird, wenn eine Eingabe erfolgt. So kann bei bestimmten Eingaben eine spezifische Reaktion ausgelöst werden, die als Routine im Slot hinterlegt ist. Beispielsweise wird hinter einem „PushButton“ eine Funktion ausgeführt, sobald der Button geklickt wird. Es ist außerdem möglich, Tooltips anzuzeigen, die beim Hovern über ein Eingabefeld erscheinen und dem Nutzer Details zur Eingabe geben. Auch kann bereits ein Platzhaltertext im Eingabefeld angezeigt werden, der beispielsweise erklärt, wie die Eingabe formatiert werden muss. Diese und viele weitere Details im Oberflächendesign sorgen dafür, dass der Anwender sich später beim nutzen des Programms leichter zurecht findet.

Beim Bearbeiten soll später der markierte Datensatz in das Fenster hineingeladen werden und nach Bearbeitung und Speichervorgang mit den Werten aus dem Fenster überschrieben werden. Nachdem klar war, dass das Neuerfassen von Patientendaten definitiv umgesetzt

werden soll, kam die Idee auf, ein Fenster für zwei Funktionen zu designen. Das Konzept sieht vor, dass sowohl beim bearbeiten als auch beim erstellen eines Datensatzes das selbe Fenster geöffnet wird. Allerdings mit unterschiedlicher Hintergrund-Routine. Soll ein neuer Datensatz geöffnet werden, so wird zunächst ein leerer Datensatz generiert und dann in das Fenster geladen. Dieser kann dann wie ein normaler Datensatz bearbeitet und gespeichert werden. Die Methode hat den Vorteil das der Code lediglich um die Erstellung des leeren Datensatzes erweitert werden muss, um zwei grundlegende Funktionen zu realisieren.

Neben dem Bearbeiten und Erstellen von Datensätzen sollte es die Möglichkeit geben, einen Datensatz detailliert einsehen zu können. Das Fenster hierfür listet den im ausgewählten Datensatz in einem QTableWigdet auf und gibt außerdem Einblick in das Krankheitsbild (mittels Decodierung des ICD-10 Codes). Bei der Erstellung des Fensters wurde zunächst noch eine weitere Funktion oberflächlich implementiert aber im späteren Verlauf wieder herausgenommen da sie nicht mehr umgesetzt werden konnte. Konkret ging es um ein Texteingabefeld, welches dem Arzt dazu dient, Protokoll über die Behandlung zu führen. Allerdings hätte man diesen Text jedem Datensatz als zusätzliche Kategorie anhängen müssen und sämtliche Funktionen, die auf Daten zugreifen, dementsprechend anpassen müssen. Da diese Idee erst aufkam als das Projekt schon gut fortgeschritten war wurde auf eine Umsetzung verzichtet.

Das dritte und letzte zusätzliche Fenster der Multiwindow-Anwendung diente der Neuanlegung von Benutzerkonten. Hierbei kann über eine Dropdown-Eingabe der Status des Accounts (Admin oder Benutzer) festgelegt werden, sowie der Benutzername und ein Passwort angelegt werden. Das Passwort muss zum Schutz vor Eingabefehlern wiederholt werden. Bei den Eingabefeldern wurde eine Maximale Zeichenanzahl hinterlegt um sich an die Vorgaben der Nutzerdatenverwaltung anzupassen.

Die Erstellung der Dokumentation (Deckblatt, Formatierung, Einleitung) markiert den letzten Punkt meines Aufgabenbereiches.

2.5. Bericht Johannes Winter

Dokumentation Johannes Winter

Thema: GUI für das Mainwindow und Loginfenster der EPA

Die Benutzeroberflächen (GUI) für das Mainwindow und das Loginfenster der EPA dienen dazu, die im Code implementierten Klassen visuell darzustellen. Sie ermöglichen es dem Nutzer, die bereitgestellten Funktionen intuitiv zu bedienen, z.B. durch das Klicken auf Buttons oder die Interaktion mit anderen Steuerelementen.

Loginfenster

Das Loginfenster der elektronischen Patientenakte besteht aus QLabels, QLineEdits und QPushbuttons. Das Stylesheet wurde im ui-File erstellt, da es Probleme beim Übernehmen vom Stylesheet im Code gab.

on_auge_btn_clicked():

Diese Funktion ist mit dem **auge_btn** verknüpft. Das bedeutet, dass sie ausgeführt wird, sobald der zugehörige Push-Button betätigt wird. Innerhalb der Funktion wird mithilfe einer **if-else-Abfrage** überprüft, ob der Echo-Modus des Passwort-**QLineEdits** auf „Password“ gesetzt ist. Ist dies der Fall, wird der Echo-Modus bei Betätigung des Buttons auf „Normal“ (sichtbar) geändert. Gleichzeitig wird das Icon des Buttons von einem offenen Auge auf ein geschlossenes Auge umgeschaltet. Bei erneutem Betätigen des Buttons erfolgt die umgekehrte Änderung: Der Echo-Modus wird wieder auf „Password“ gesetzt, und das Icon wechselt zurück zu einem offenen Auge.

Der Benutzer hat die Möglichkeit, seinen **Benutzernamen** und das **Passwort** in die entsprechenden **QLineEdits** einzutragen. Sobald entweder die **Enter-Taste** gedrückt oder der Push-Button mit der Aufschrift „Login“ betätigt wird, werden die eingegebenen Strings überprüft. Bei korrekten Eingaben wird der Benutzer ins **Mainwindow** weitergeleitet. Andernfalls wird eine Warnung in Form einer **QMessageBox** angezeigt.

Mainwindow

Das mainwindow.ui-File bildet die Startseite der EPA und dient als zentrale Benutzeroberfläche. Es besteht aus mehreren funktionalen Komponenten, die eine intuitive Navigation und Interaktion ermöglichen.

Aufbau und Funktionen

1. Tabelle mit Such- und Filterfunktion
    - Im suche_txt_line können Nutzer nach spezifischen Patienteninformationen suchen, wobei man in der rechts angeordneten Combobox die Suchkriterien vorgibt (Patienten-ID, Vorname oder Nachname).
    - Die Suchergebnisse werden nach Betätigung der Enter-Taste oder des suche_btn in der Tabelle angezeigt.
2. Detailansicht eines Patienten
    - Das kleinere Fenster neben der Tabelle dient der detaillierten Ansicht eines einzelnen Patienten, sobald dieser in der Tabelle ausgewählt wird.

3. Pushbuttons mit spezifischen Funktionen
    - Das Mainwindow enthält mehrere Pushbuttons, die folgende Aktionen unterstützen:
        - Speichern oder Laden von CSV-Dateien
        - Bearbeiten von Patientendaten
        - Hinzufügen neuer Patienten
    - Jeder Button ist mit passenden Icons und Tooltips versehen, die beim Hovern mit der Maus angezeigt werden und die Bedienung erleichtern.

Design und Stylesheet

1. Darkmode und Whitemode
    - Ein Radiobutton ermöglicht das Umschalten zwischen Darkmode und Whitemode.
    - Die Funktion on_darkmode_btn_toggled(bool checked) erkennt den Zustand des Radiobuttons und wechselt dynamisch das Stylesheet der Benutzeroberfläche.
    - Beide Styles (Lightmode und Darkmode) sind als Raw-Strings hinterlegt. Standardmäßig ist der Lightmode aktiv.
2. Logout-Pushbutton
    - Der Logout-Button verwendet ein eigenes Stylesheet, um optisch hervorgehoben zu werden.
    - Beim Betätigen des Buttons wird das mainwindow.ui geschlossen und das Loginfenster wieder geöffnet.

Diese GUI ermöglicht eine übersichtliche und effiziente Bedienung der EPA, wobei sowohl Funktionalität als auch Benutzerfreundlichkeit im Fokus stehen.

Filtericon: https://static-00.iconduck.com/assets.00/filter-icon-512x512-v9trade2.png

Augenicon(Login): https://www.pngwing.com/de/free-png-ajnuo