Requirements Traceability Matrix

ID	Requirement	Related Use Case	Implemented by	Tested by	Description
1	The application interface is accessible and provides display of data and navigation to the pump's features	Start Device (UC1)	MainWindow.ui, MainWindow.cpp	Run the simulation in Qt, enter the passcode (1234), charge the device and enter the one of the home screens	Replicated the Tandem t:slim Insulin Pump user interface using Qt's built-in design interface. A page history is kept to allow the back buttons to dynamically determine the correct previous page.
2	The application displays date, time, battery and insulin levels which update according to the model	N/A	MainWindow.ui, MainWindow.cpp	Upon entering the home screen, observe the changing time, battery level and insulin gauge at the top of the screen	MainWindow contains a QTimer which upon entering the home screen will simulate 5 minutes every second, incrementing simulationTime which keeps track of how long the simulation has been running and updating the time, battery and insulin gauge.
3	Simulation has a CRUD interface for personal profiles	Setup Personal Profiles (UC2)	ProfileFormWidget.cpp, ProfilePageWidget.cpp, Profile.cpp, InsulinDeliveryProfile.cpp	From the home screen, navigate to Options > My Pump > Personal Profiles to create and activate a profile	The ProfilePageWidget contains all the user's profiles where they can choose the one they wish to set as active which will update curProfile in MainWindow. New profiles can be added using the add button which enters the ProfileFormWidget.
4	Profiles can set target blood glucose, correction factor, basal rate and carb ratio to customize automatic and manual insulin delivery	Setup Personal Profiles (UC2)	ProfileFormWidget.cpp, ProfilePageWidget.cpp, Profile.cpp, InsulinDeliveryProfile.cpp	From the personal profiles page, click the add button to open the profile creation form and tailor the values	The ProfileFormWidget enables intuitive customization of time segments, basal rate, correction factor, carb ratio, target blood glucose, maximum bolus and insulin duration. The profile created will be stored in a list in ProfilePageWidget.
5	Simulation has a menu where the user can deliver a bolus manually which can be set take effect immediately or extended over a	Manual Insulin Bolus Delivery (UC4)	MainWindow.cpp, BolusCalculator.cpp	From the home screen, the bolus button opens the bolus calculation menu. After confirming, the user can choose to have	MainWindow handles the buttons that interact with the BolusCalculator to set carb and blood glucose values used to calculate the amount of insulin to dose, which can be tweaked by the user. They can also

	period of time			portion of the bolus extended	choose what portion to delivery immediately and over time.
6	Pump data like IOB, blood glucose and carbs are automatically fetched and/or can be provided by the user to facilitate bolus calculations	Manual Insulin Bolus Delivery (UC4)	MainWindow.cpp, BolusCalculator.cpp	IOB and blood glucose can be seen from the home screen/CGM graph and shown in the bolus calculations popup	The current blood glucose and insulin on board is taken from the CGM by MainWindow and provided to the BolusCalculator to provide helpful values that can be used to quickly determine the ideal amount of insulin to dose.
7	A simulation time is used and kept track of to handle and log background tasks	N/A	MainWindow.cpp	The time displayed is closely tied to simulation time and is displayed and used for logging	The simulation time is converted into a QDateTime displayTime which is accessible to the logging functions to log CGM reading, insulin delivery and alerts.
8	Continuous Glucose Monitoring (CGM) data is provided to the program via simulated readings	Monitor Glucose Level (UC3)	CGM.cpp GlucoseReader_Mock.cpp	Upon entering the CGM home screen, a graph of the patients blood glucose can be seen fluctuating	A GlucoseReader (mock) object provides simulated blood glucose readings. Using the strategy pattern, different methods of simulation data can be switched out seamlessly.
9	Insulin is delivered automatically and dynamically adjusts based on CGM readings	Basal Insulin Delivery (UC5)	CGM.cpp GlucoseReader_Mock.cpp	The CGM graph can be observed to reduce when approaching the yellow line, or when over the profile's target blood glucose	The CGM passes effects to the GlucoseReader based on the profile's basal rate and correction factor which acts as basal insulin reducing blood glucose readings. This is stopped when the blood glucose dips below the profile's target.
10	Insulin on board (IOB) is tracked as the amount of insulin that is continuing to affect blood glucose	N/A	CGM.cpp GlucoseReader_Mock.cpp	When delivering an extended bolus, the IOB values on the home screens will display the amount and duration of insulin is persisting	The CGM can request the insulin on board from the GlucoseReader, which has this information in the form of a list of its active effects, whose insulin effect can be added up and duration returned.
11	CGM data, insulin delivery and alerts are automatically logged by the system	Generate Insulin Data/History (UC6)	MainWindow.cpp, Event.cpp EventHistory.cpp	The user can navigate to history and verify that the alert/CGM reading at a given time is logged	MainWindow will call functions to log CGM readings and the amount of insulin delivered (if applicable) in the event loop, which runs every real second or 5 simulated minutes.

12	Simulation has a menu where logs can be viewed in further detail based on the time they occurred	Generate Insulin Data/History (UC6)	MainWindow.cpp, Event.cpp EventHistory.cpp	From the home screen, Options > History allows the user to select and view the type of event they wish to see logs of	MainWindow keeps a list of Event objects. The time of each log is kept and displayed on the left of the history screen. For more detail, the time can be clicked to view the specific information associated.
13	Simulation has a PIN-based lock screen, charging, holding power button, and insulin refill mechanism	N/A (comes from 2. Getting Started in the specification)	MainWindow.ui, MainWindow.cpp	The PIN appears when the program is run. After entering the passcode, the power screen is displayed providing charging and insulin refill	The MainWindow uses a keypad layout to create a lock screen with a PIN of “1234”. A message is displayed when the incorrect PIN is entered.
14	Simulation creates alerts upon low battery, low insulin gauge and dangerously high/low blood glucose levels	Error Handling (UC 7-10, step 7. in the specification)	MainWindow.cpp	When the blood glucose crosses over the red/yellow line, a popup will alert the user that blood sugar levels are critically high/low	MainWindow keeps track of battery and insulin gauge and creates an AlertEvent if too low. It also receives blood glucose levels from the CGM and creates an alert if it is critically high or low.
15	Simulation suspends insulin delivery when insulin gauge is empty	Error Handling (UC 7-10, step 7. in the specification)	BolusCalculator.cpp CGM.cpp GlucoseReader_Mock	The CGM graph will not be affected by any insulin that was supposed to be delivered when the insulin gauge is at 0.	MainWindow keeps track of the insulin gauge and provides it to the CGM which will suspend insulin delivery when it reaches 0.
16	Simulation handles the pump running out of battery	Error Handling (UC 7-10, step 7. in the specification)	MainWindow.cpp	When the battery is 0, the program will kick the user to the power screen where they can recharge the pump	MainWindow keeps track of the battery and kicks the user back to the power screen when it reaches 0.