# **Team Members**

GitHub: <https://github.com/martin-wilkinson/CP3407-Group1>

Trello: https://trello.com/invite/cp3407group1/4ad8904092ea3bf883144bba8a746cb2

Project role(s), specific individual contributions to this assessment item:

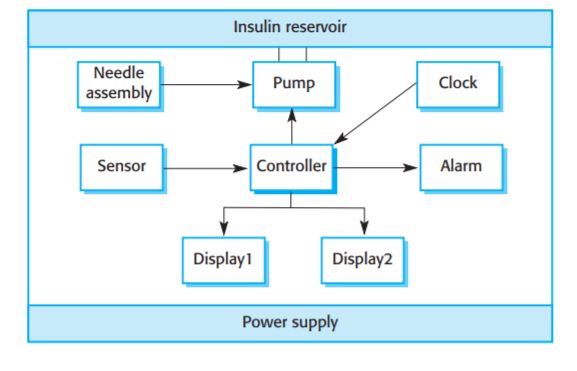
Student Name-1: Jack Chilby - Finalise Mockups, User stories and Scope.

Student Name-2: Martin Wilkinson - Project Description, UML diagram

Student Name-3: Jacob Jackson - Project Development and Database Design

# **Project Description**

Diabetes is a condition in which a person’s pancreas is unable to produce enough of the insulin hormone required to metabolise the sugar in their blood. Untreated, sufferers can experience a multitude of health issues including eye problems, kidney problems and death. Fortunately, diabetes can be managed by monitoring one’s BGL (blood glucose level) and administering insulin shots as required. As an alternative to the shots, a person with diabetes may instead opt to use an insulin pump. A typical insulin pump, is a device the size of a cell phone that can be configured by a medical professional with the user’s insulin requirements and administers the insulin in micro doses constantly throughout the day, closer mimicking the normal function of the pancreas. The pump itself consists of an insulin reservoir, a computer that can be programmed to control the pump, an LCD display to assist with programming the computer, a catheter which connects to the pump to deliver the insulin into the user’s body and a small motor that pumps the insulin. Currently, this system still requires the regular measuring of BGL and manual adjustment of the pump accordingly, it also doesn’t account automatically for food, with users still needing to calculate required insulin based on the calories they are about to eat and manually have the pump administer the extra insulin required. Our solution brings the insulin pump closer to replicating the natural function of the pancreas. As can be seen from *Figure 1* the hardware schematic for the pump we will be controlling includes a sensor, this sensor will provide accurate data on the user’s BGL allowing us to have the controller calculate automatically what amount of insulin the user requires.



**Project Planning/Scope**

|  |  |  |
| --- | --- | --- |
|  | **Description** | **Estimated Days** |
| **User story 1:**  Start Up Process. | The user starts the pump.  Pump runs through checks.(Battery, Sensor, Pump, Delivery, Needle Assembly, Insulin reservoir) | 5 Days |
| **User story 2:**  User wants to see the blood sugar level. | Blood sugar level is displayed on screen, updated every 10 minutes. | 2 Days |
| **User story 3:**  Pump should deliver the correct amount of insulin. | The pump calculated the amount to be delivered according to the current sugar reading, measured by the sensor.  Sugar Low = No Insulin  Sugar High = Insulin  If within the safe zone, insulin is only to be delivered if sugar level is rising.  Taking into account for Cumulative dose per day and Single injection limits. | 5 Days |
| **User story 4:**  User wants to be able to see the Dosage History. | The pump is to display the last delivery of insulin. | 2 Days |
| **User story 5:**  The user changes the Insulin reservoir. | The Insulin reservoir is malfunctioning or is empty and needs to be changed or re-seated. | 3 Days |
| **User story 6:**  The cumulative dose of insulin delivered, stored in Database and reset. | At the end of the 24hr period, (00:00:00) the cumulative dose is stored in the Database and reset. | 3 Days |
| **User story 7:**  The Dosages throughout the day are stored in the Database. | Each dosage is stored in a Database along with a Timestamp and Current sugar level. | 4 Days |
| **User story 8:**  The user switches to manual mode. | The user is able to switch the device to manual. | 2 Days |
| **User story 9:**  In manual mode the user enters Carbs and Insulin. | The user is prompted to enter the Carbs and Insulin levels so that the pump can deliver the correct dosage. | 4 Days |
| **User story 10:**  Device runs Battery check | Checks the voltage of the battery.  Less than 0.5v requires replacement. | 2 Days |
| **User story 11:**  Device runs pump sensor failure check. | Checks the sugar sensor is operational. | 2 Days |
| **User story 12:**  Device runs pump failure check. | Checks the pump is operational. | 2 Days |
| **User story 13:**  Device runs delivery check. | Checks if the delivery has been successful.  Delivery needle may be blocked or incorrectly inserted. | 2 Days |
| **User story 14:**  Device checks for Needle assembly. | Checks if the needle assembly has been installed correctly. | 2 Days |
| **User story 15:**  Device checks for insulin reservoir. | Checks if the reservoir has been installed correctly. | 2 Days |
| **User story 16:**  Device checks insulin level. | Checks the capacity of the insulin reservoir. | 2 Days |
| **User story 17:**  Device runs through all checks systematically. | Device runs through all checks on the pump and alerts user if any errors have occurred. | 1 Days |

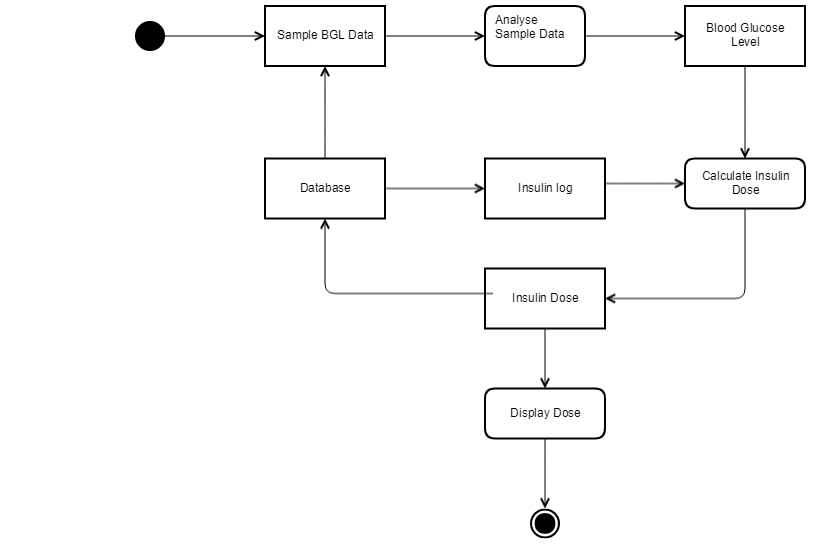
Days: 36 – 108 (min-max)

Total estimated: 47 Days

**Project Goals**

Our project is to develop the software that will control an insulin pump as per the hardware schematic in Figure 1. This software will then be run in a virtual simulation of the pump operation with a graphical user interface that displays the various responses of the pump software to different test data. To make our accessible to the lecturer we will be keeping all source code in a github repository said lecturer has access to. As such they merely have to clone the repository and run the source code in an appropriate IDE.

# **Project Design**



# database model v1.PNG

# **C:\Users\jimji\AppData\Local\Microsoft\Windows\INetCache\Content.Word\InsulinPumpMockup.pngC:\Users\jimji\AppData\Local\Microsoft\Windows\INetCache\Content.Word\FirstMockup.png**

# **Project Development and Infrastructure**

The programming language that will be used is Python, a general-purpose programming language. The majority of members within our group have experience with the language, and feel that the simplicity and flexible of the language would be easy to teach to the remaining people. Additionally, because it’s one of the more popular languages, there would be a ton of information and support out there to help in the creation of the application. There was some discussion about other programming language to use for development. Java was an option, and although there was praise for its adoption in many things, the problem was that nobody really liked working with the language. Other languages were brought up as well, but were almost instantly ignored because nobody knew how to work in it or it was simply disliked by everyone.

Since Python is a language that can be run on many different platforms, including Windows, Mac and Linux, everyone in the group can use any sort of PC to develop the application. In addition to that, Python has the capabilities to be run on a hosting server, which means that it could be accessed through any devices that have web capabilities. The IDE that each of us will be using for writing the Python code is PyCharm, by the developer JetBrains. This is because all of us have used JetBrain products and have found them to be useful tools in developing.

In terms of GUI, a proper decision about what to use hasn’t been decided yet. There has been a few different options presented, and each have their positives and negatives. One of them is TkInter, the default GUI package that has been built into Python. Another is a cross-platform framework named Kivy, which can be used for mobile applications and have multi-touch technologies. Finally, the other option is called PyGUI, which was a project designed to run a smooth, lightweight GUI implementation for Python.

There also needs to be an implementation of a database into the application. Therefore, the Python DB-API will be used, which has support for a whole range of different databases. The type of database that will be a relational database, which will be used to store all the values that the application will read in regards to blood level, etc. The chosen database to use will be SQLite, which is an open-source database which is embedded into the end-system, rather than being client-side. This means not having to worry online connections to a database, since it’s built into the application.

For collaboration purposes, the group chatting service Slack will be used, allowing for the creation of specific-purpose channels and integration of many different useful services. Additionally, the project management application Trello will be used during development, where roles and tasks can be assigned to different people. For configuration management, the version control command line Git will be used, due to its universal usage in terms of version control, and because of its integration within PyCharm. GitHub is going to be the hosting service for these git commits.

References

[https://sqlite.org](https://sqlite.org/)

<https://kivy.org/#home>

### [**http://www.cosc.canterbury.ac.nz/greg.ewing/python\_gui/**](http://www.cosc.canterbury.ac.nz/greg.ewing/python_gui/)

<https://www.medtronic-diabetes.com.au/pump-therapy/what-is-insulin-pump-therapy>:

<http://iansommerville.com/software-engineering-book/files/2014/10/InsulinPumpOverview.pdf>

<https://www.medtronic-diabetes.com.au/pump-therapy/other-pumps/minimed-paradigm>

<https://www.accu-chek.com.au/insulin-delivery-system/combo-system>

https://www.medtronic-diabetes.com.au/pump-therapy/other-pumps/minimed-veo