A SAFETY CRITICAL SYSTEMS

Project Report - **"Software Simulation of an Insulin And Glucagon Pump"**

Ahmed Asad(1321474)
Girdhar Shubham(1323003)
Shaikh Safir Mohammad(1322554)
Usmani Faiz(1323197)
Yelpale Kshitij(1322509)

Contents

1 INTRODUCTION

Insulin and glucagon are hormones produced by pancreas in the human body. When a person consumes a meal, the blood glucose level in the body surges. The insulin then comes into picture and transfers that glucose to muscles and other tissues. This keeps the blood glucose levels in check. Diabetes is a condition when the human body either does not produce insulin or doesn't respond to insulin properly. In either case, the blood glucose can reach very high levels (hyperglycemia), which could lead to serious health issues like heart, kidney and vision problems. On the counter side, blood glucose levels dropping to very low levels (hypoglycemia) can also have severe consequences like loss of consciousness, comma and even death. Glucagon comes to the rescue when such condition arrives in human body and helps in increasing the blood glucose level.

2 PROJECT SCOPE

This project is developed as part of the module Safety Critical Systems. Safety Critical Systems have a direct impact on life, human health and the environment. Medical instruments are certainly Safety Critical. The intent here is to study and replicate the human body's blood glucose behaviour. Then simulate an Insulin-Glucagon pump that works autonomously and injects the required hormone to make sure that the blood glucose levels remain in safe range. The pump should be able to select and calculate the quantities of the hormone that it has to be injected. The safety and security aspects like alert messages, email and user authentication are thought off and handled. The system is designed as a web application. User interface for the system is from the perspective of a patient. Although various other user perspectives could be implemented (like doctor, nurse etc), due to over-running cost estimates, currently the functionalities are limited to patient only.

2.1 MAJOR SOFTWARE FUNCTIONS

- Login/register functionality for user
- User authentication via email
- Monitoring and graphical/textual representation of blood glucose level
- Insulin/glucagon dosage calculation and injection
- Monitoring battery levels and insulin/glucagon reservoir
- Alerts at critical stages

3 REQUIREMENT ANALYSIS

The requirements for the simulation project can be majorly divided into following 2 categories:

3.1 FUNCTIONAL REQUIREMENTS

- Patient should be able to register/login to the web application
- Patient should be able to monitor current blood glucose levels
- The system should specify the safety range for the blood glucose level
- In case of blood glucose level crossing the threshold for safe limit, system should be able to decide which hormone needs to be injected and calculate the dosage amount
- The Insulin/glucagon reservoirs should be maintained and updated whenever a hormone is consumed for injection
- Battery of the system should be displayed to the user

3.2 SAFETY REQUIREMENTS

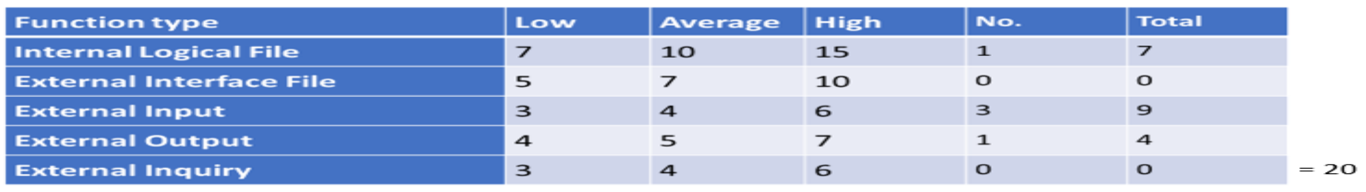
- User registration should be authenticated with email verification
- There should be appropriate alert messages whenever glucose level is critical
- If the Blood glucose level is critical for some threshold time, an email should be sent to the emergency contact
- There should be appropriate alert messages when the insulin/glucagon reservoir and battery go below a certain level.

4 COST ESTIMATION

Project Estimation using COCOMO 2
We used Intermediate COCOMO to calculate the cost estimation for our project. Intermediate COCOMO takes into account personal and hardware factors into account which are known as "Cost Driver Attributes".

Historical Data used for estimates
Effort = 5 Person per month (6 hrs per week per person, 120 hrs per month), Schedule = 80-90 days (2.5 – 3 months)

Calculating Unadjusted Functional Points (UFP)



Function type	Low	Average	High	No.	Total
Internal Logical File	7	10	15	1	7
External Interface File	5	7	10	0	0
External Input	3	4	6	3	9
External Output	4	5	7	1	4
External Inquiry	3	4	6	0	0

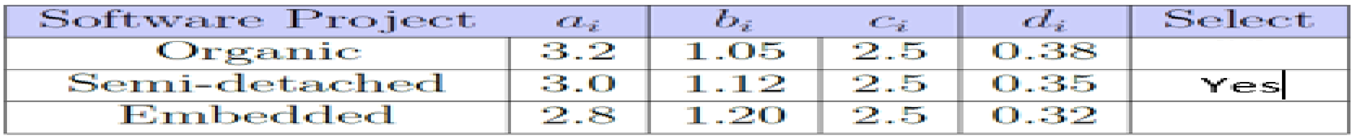
= 20

Calculating SLOC
JAVA(Spring Boot for Backend) and Javascript(Angular 6 for frontend) are chosen as our programming languages. Language Factor (LOC per function point is around 300 average for both.)
So Lines of Code = 20*300 = 6000. Converting SLOC into KLOC by dividing it by 1000 = 6000/1000 = 6

Calculating Effort Adjustment Factors using Cost Drivers (Factors with ratings)

- Product Attributes : Required Software Reliability : 1.40, Size of Application Database : 0.94, Complexity of the product : 0.70
- Hardware Attributes : Run-time Performance Constraints : 1, Memory Constraints : 1, Volatality of the Virtual Machine environment : 1, Required Turnabout time : 1
- Personnel Attributes : Analyst Capability : 1, Application Experience : 1, Software Enginner Capability : 1, Virtual Machine Experience : 1.1, Programming Language Experience : 1
- Project Attributes : Application of Software Engineering Methods : 0.82, Use of Software Tools : 0.83, Required Development Schedule : 1.08
- EAF = Multiplication of the choosen factor values = 0.74

Calculating Persons per month and Development time As our team is a mixture of experienced and inexperienced staff and the project contains some unfamiliar aspects that need to be developed, so we identified ourselves as semi- detached.



Software Project	a_i	b_i	c_i	d_i	Select
Organic	3.2	1.05	2.5	0.38	
Semi-detached	3.0	1.12	2.5	0.35	Yes
Embedded	2.8	1.20	2.5	0.32	

- Effort applied = a^i(KLOC) ^{bi} * EAF [man-months] = 3.0 * (6) $^{1.12}$ * 0.74 = 16.5 [man-months]
- Development Time = c^i (Effort Applied) ^{di} [months] = 2.5 * (15.4) $^{0.35}$ = 6.7 months
- People required = Effort Applied / Development Time [count] = 16.5/6.7 = 2.5 people

As we are a team of 5 people, Development Time = 16.5/5 = 3.3 months
We calculated this at the start of December and we had approximately 2.5 months to develop our application. So keeping that in mind we decided to develop only the core functionality of a patient's use case.

5 PROCESS MODEL

Agile iterative approach is most suited to deal with development of complex technical systems and change management but the development of a safety critical system is governed strictly by safety standards. So, the process model we have chosen for our project is Agile Scrum with some safety incorporations according to standard IEC-61508 [1]. The project was split into sprints of 2 weeks each. Sprint planning, requirement analysis, design, implementation, testing, sprint retrospective, backlog review and demo/presentation were phases in every sprint. There were bi-weekly team meetings to monitor the progress.

Mapping agile scrum to IEC-61508
The product backlog from scrum was mapped to software safety requirements additionally. The backlog review and sprint retrospective from scrum was mapped to system safety validation and verification.

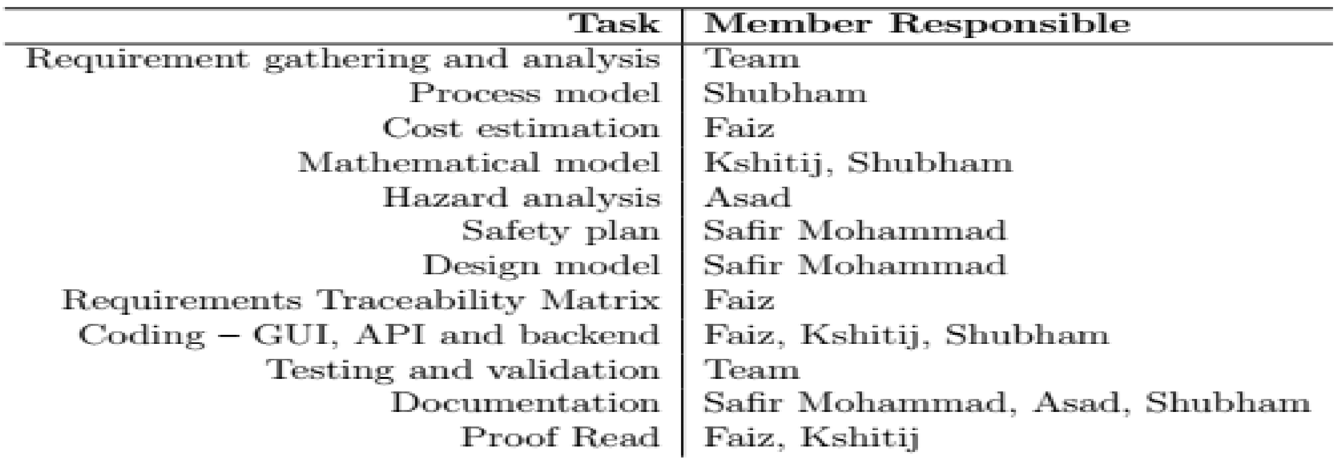
Difficulties in implementing AGILE for the development of SCS (and how we tackled these):

- Everybody has licence to change anything – this is handled by proper monitoring, review process, validation and verification.
- Working software over comprehensive documentation – this is handled by enabling comprehensive documentation on a sprint wise basis. Traceability is also taken care of.

5.1 TEAM ORGANISATION

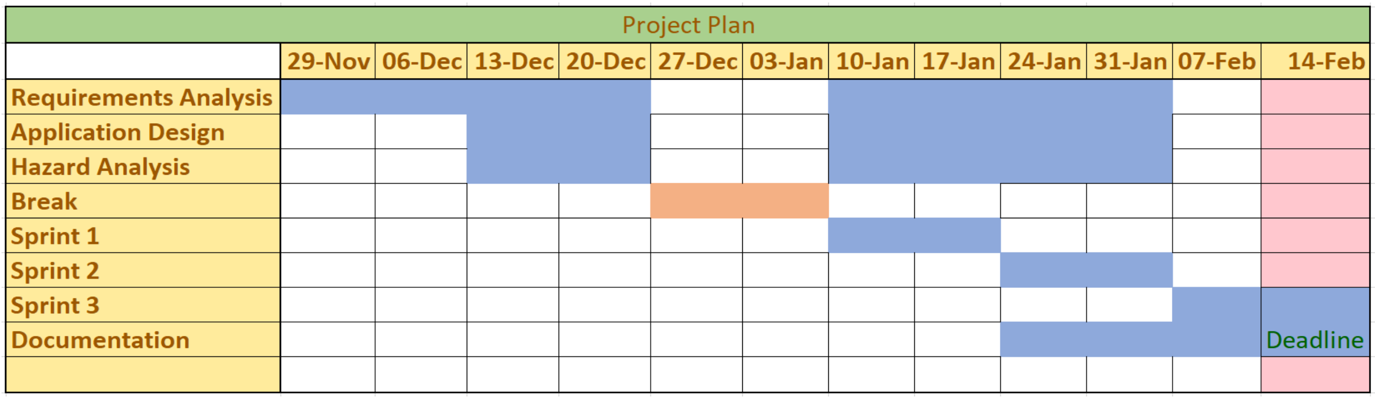
The entire team worked as one self-organising cross functional scrum team. Each team member worked as a designer, developer or tester depending on the phase of the sprint. As a result of this approach, the complete capacity of team was well utilised during the course of the project.

5.2 TASK DISTRIBUTION



Task	Member Responsible
Requirement gathering and analysis	Team
Process model	Shubham
Cost estimation	Faiz
Mathematical model	Kshitij, Shubham
Hazard analysis	Asad
Safety plan	Safir Mohammad
Design model	Safir Mohammad
Requirements Traceability Matrix	Faiz
Coding – GUI, API and backend	Faiz, Kshitij, Shubham
Testing and validation	Team
Documentation	Safir Mohammad, Asad, Shubham
Proof Read	Faiz, Kshitij

6 MILESTONES AND PROJECT TIMELINE



Project Plan

	29-Nov	06-Dec	13-Dec	20-Dec	27-Dec	03-Jan	10-Jan	17-Jan	24-Jan	31-Jan	07-Feb	14-Feb
Requirements Analysis	🟦	🟦	🟦	🟦			🟦	🟦	🟦	🟦		🟥
Application Design			🟦	🟦			🟦	🟦	🟦	🟦		🟥
Hazard Analysis			🟦	🟦			🟦	🟦	🟦	🟦		🟥
Break					🟧	🟧						🟥
Sprint 1							🟦	🟦				🟥
Sprint 2									🟦	🟦		🟥
Sprint 3											🟦	🟦
Documentation									🟦	🟦	🟦	🟦 Deadline
												🟥

7 MATHEMATICAL MODEL

Blood glucose modelling

For the mathematical model we came across a research paper [2] where we had 2 Ordinary Differential Equations (ODEs):

$$\frac{dC}{dt} = -k_1 C \quad (1) \qquad \textit{and} \qquad \frac{dG}{dt} = -k_1 C - k_2(G - G_0) \qquad (2)$$

Here C and G denote concentration of carbohydrates in the stomach and concentration of glucose in blood. The initial conditions $C(0) = A_0$ and $G(0) = G_0$. Constants k_1 and k_2 denote the metabolism rate and rate of release of insulin. These ODEs were then solved to reach an equation that would be used to simulate blood sugar levels.

- First Eq. 1 was solved to with the initial condition to find the value of C(t).
- This value was then substituted in Eq.2, which was then solved with its initial condition to get an equation for G(t).
- This equation was then used to estimate the value of blood glucose at various time intervals. The values of constants that give results closest to real valued data are:
  $G_0 = 90, \quad A_0 = 121.7, \quad k_1 = 0.0453, \quad k_2 = 0.0224$

BGL Range	Current BGL – Target BGL
120 – 130	5
130 – 150	10
150 – 180	15
180 – 220	20
220 +	25

So, the final equation for the glucose level that was derived is:

$$G(t) = A_0 \frac{k_1}{k_2 - k_1}(e^{-k_1 t} - e^{-k_2 t}) + G_0$$

After substituting the values of the constants from step 3, the equation becomes:

$$G(t) = 90 - 240.7428(e^{-0.0453t} - e^{-0.0224t})$$

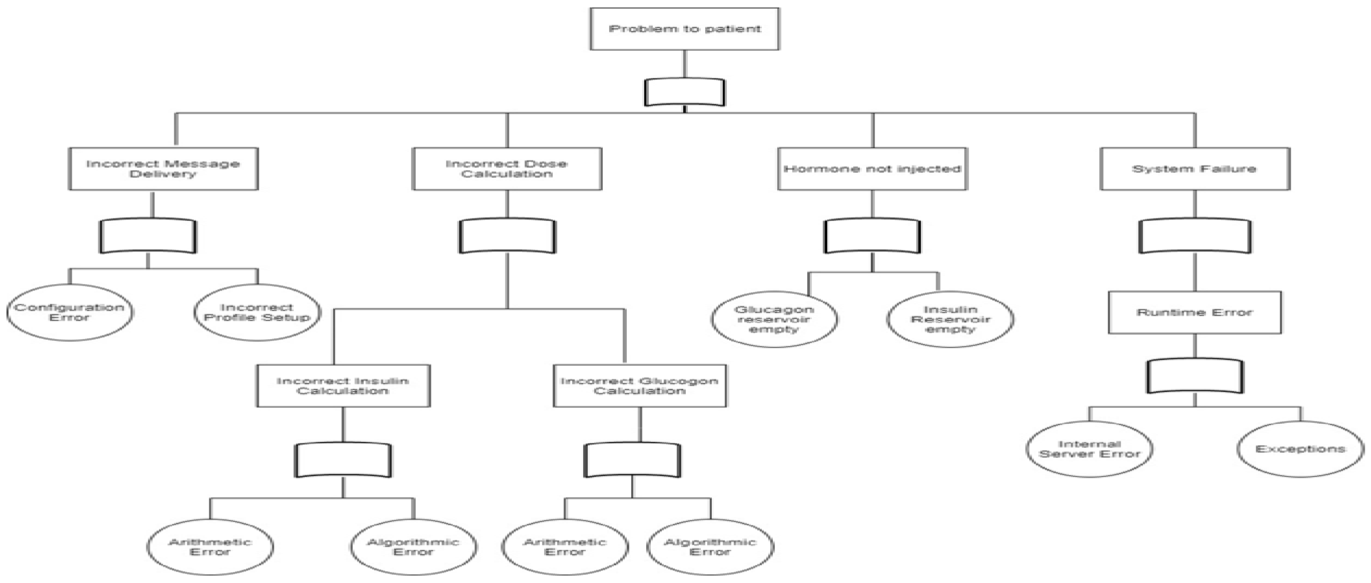
The safe range of blood glucose level is considered to be from 70 mg/dL – 120 mg/dL. If the levels rise above 120 gm/dL then insulin should be injected by the pump to bring the level down to safe range. Similarly, when glucose levels fall below 70 gm/dL then glucagon should be injected. The calculation for the dosage of the insulin is shown as below (Glucagon calculation is done in a similar way, so not explicitly included in report):
Insulin dosage calculation:
For the calculation of dosage of Insulin to be injected, we are considering Insulin Sensitivity Factor (ISF). This is a metric that tells us by how much the blood glucose level will come down after injection of 1 unit of insulin. The ISF depends on a lot of factors like age, weight etc. For the sake of simplicity, we are taking the ISF value as constant (50). So, the insulin dose is defined as:
Insulin dose = (Current BGL – Target BGL)/ISF
Current BGL – Target BGL is taken according to the following table:

8 HAZARD ANALYSIS

We have considered hazard analysis as one of the most important part of our project. In hazard analysis our objective is to identify hazards and causes due to which problem could occur.

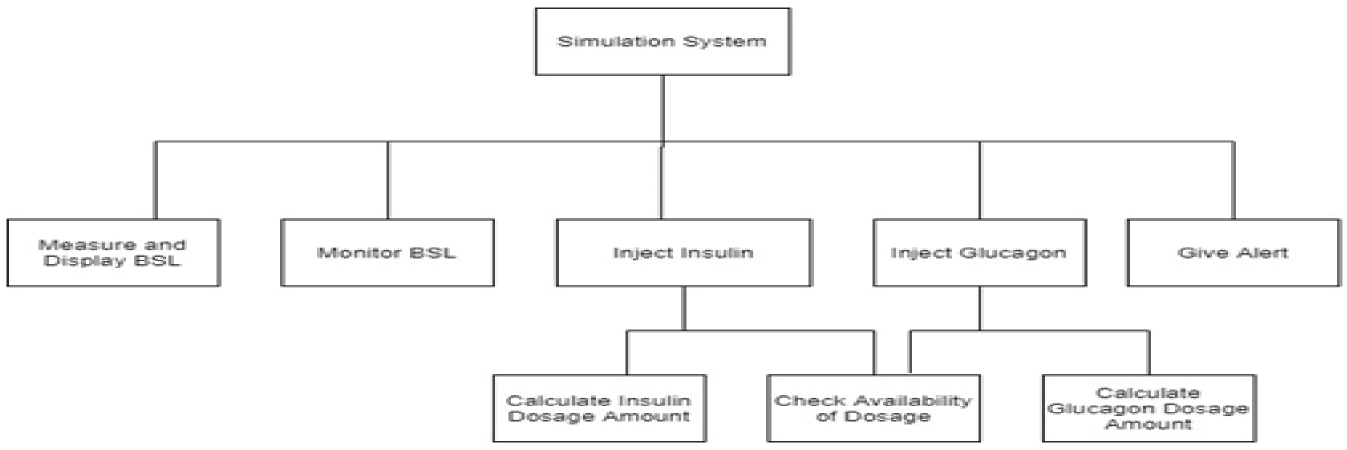
8.1 FAULT TREE ANALYSIS

We have created a fault tree analysis diagram considering our project. We have thought of three major branches Incorrect Message Delivery, incorrect dose calculation and system failure then we have drill down this to what problem could have occur in this.



8.2 STAMP APPROACH

- **Recognizing System Hazards**
  - It could also happen that insulin/glucagon is selected but problem occurs in its distribution within the time and amount
  - Reservoirs of Insulin or Glucagon can get empty.
  - Battery can be drained out.
  - In calculation of Insulin and Glucagon Arithmetic or Algorithmic errors can occur.
  - In a single instance, more than required amount of hormone can get injected.
- **Recognizing System Level Safety related to Requirements and Constraints**
  - In this project the constraint would be that it would be only applicable to Insulin/Glucagon dose calculation.
  - Another constraint would be as we are building a software simulation so no sensor would be involved in it.
  - In requirements there would be authentication as the system should be secured and we have implemented that through JSON Web Token.
  - In terms of safety system should display appropriate error and warning messages when required.
  - In our project user is alerted in the event of low pump battery, low insulin and glucagon level in reservoirs.
- **Basic System Control Structure**


Simulation System
Measure and Display BSL
Monitor BSL
Inject Insulin
Inject Glucagon
Give Alert
Calculate Insulin Dosage Amount
Check Availability of Dosage
Calculate Glucagon Dosage Amount

- **Recognizing inadequate control actions that could lead to a hazardous system state**
  - If runtime errors in our software system could reach hazardous state.
  - Another case would be if reservoirs of Glucagon or Insulin gets empty it could lead to hazardous state.
  - If Arithmetic or Algorithmic errors occurs during calculation of Insulin or Glucagon system could reach to hazardous state.
  - If Configurations are not done properly the system could move towards unsafe state
- **Determining how Constraints could be violated and how we can eliminate or reduce the probability of Hazards in our project**
  - There is a constraint that if BSL goes High (Hyperglycemia) or BSL goes Low (Hypoglycemia) an alert message will be sent to patient's contact in case of blood sugar level stays for longer period of time over the threshold.
  - Another constraint in our project is when reservoir is getting empty and in reservoir if only 10% of insulin or glucagon is left then email will be sent immediately so in this way, we are handling this to go in a hazardous state.
  - In our project we are implementing auto mode so through our software we inject the hormones insulin/glucagon automatically.

9 SAFETY PLAN

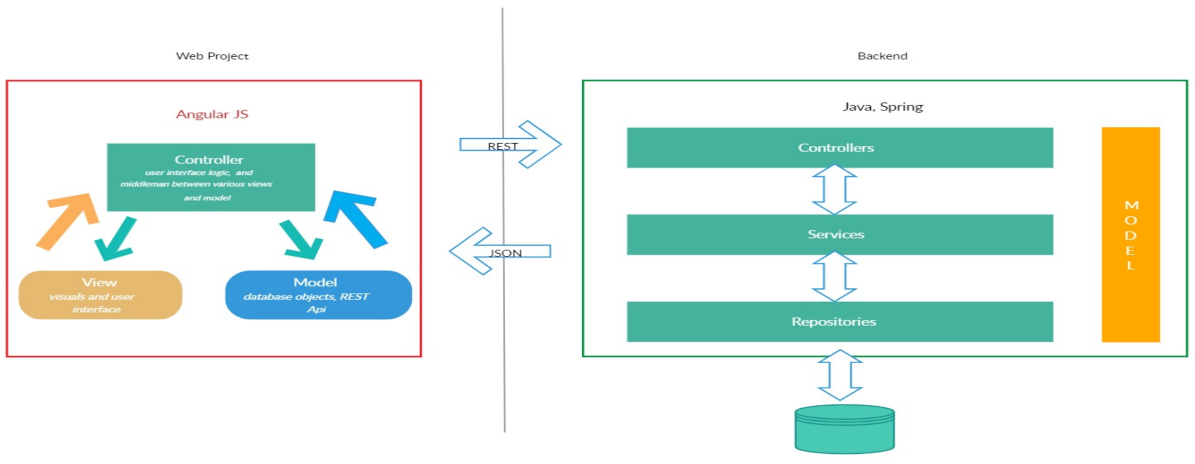
Following severe hazards were prioritized at the top as per the Safety Criticality and Process Model, which are resolved and put into safety plan

- Software Simulation System continuously monitors Pump's Battery Level and Reservoir's Status and alerts Patient in case of Low Battery and Reservoirs becoming empty.
- While entering the details, the Patient should also provide his/her dependent's details.
- The System will send an email to the dependent if any exception occurs with proper details. The exceptions can be:
  - When reservoirs are less than 10 percent.
  - Injected dosage doesn't work when BSL stays longer than a particular time.

10 DESIGN MODEL

10.1 ARCHITECTURE DIAGRAM

- We used JHipster framework which provided us front and back end basic connection of AngularJS 6 as front end and Spring Boot as backend with User account authentication and its basic functionalities like create account, login, log out and forget password.
- At the front end, using Angular Framwork, we created html page in which graph rendering and other fucntionalities implemented using html, css and typescript.
  Frontend Frameworks: angular6, html5, css3, jquery
  Design frameworks: Bootstrap, google material design
- At the back end, using Spring boot, we created REST services, which fetches the request and generates the response after running into Business logic which executes the mathematical model to evaluate the current Blood sugar level of the user/patient.
- At the front end, once response available it updates the GUI with appropriate values.
- In REST call it is mandatory to pass token to get the response. Token generation mechanism by JSON Web Token authentication way which provided by JHipster framework.

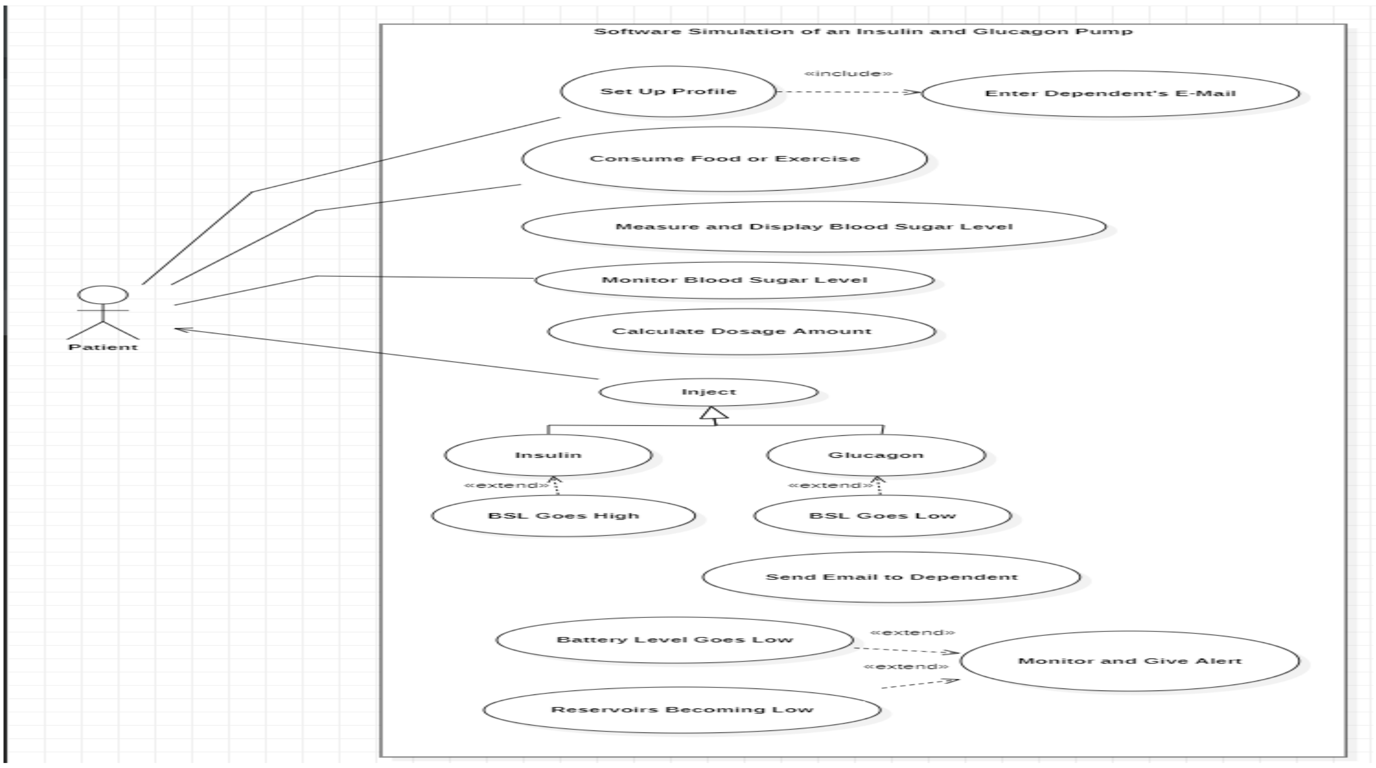

Web Project
Angular JS
Controller
user interface logic, and middleman between various views and model
View
visuals and user interface
Model
database objects, REST Api
REST
JSON
Backend
Java, Spring
Controllers
Services
Repositories
MODEL

10.2 USE CASE DIAGRAM

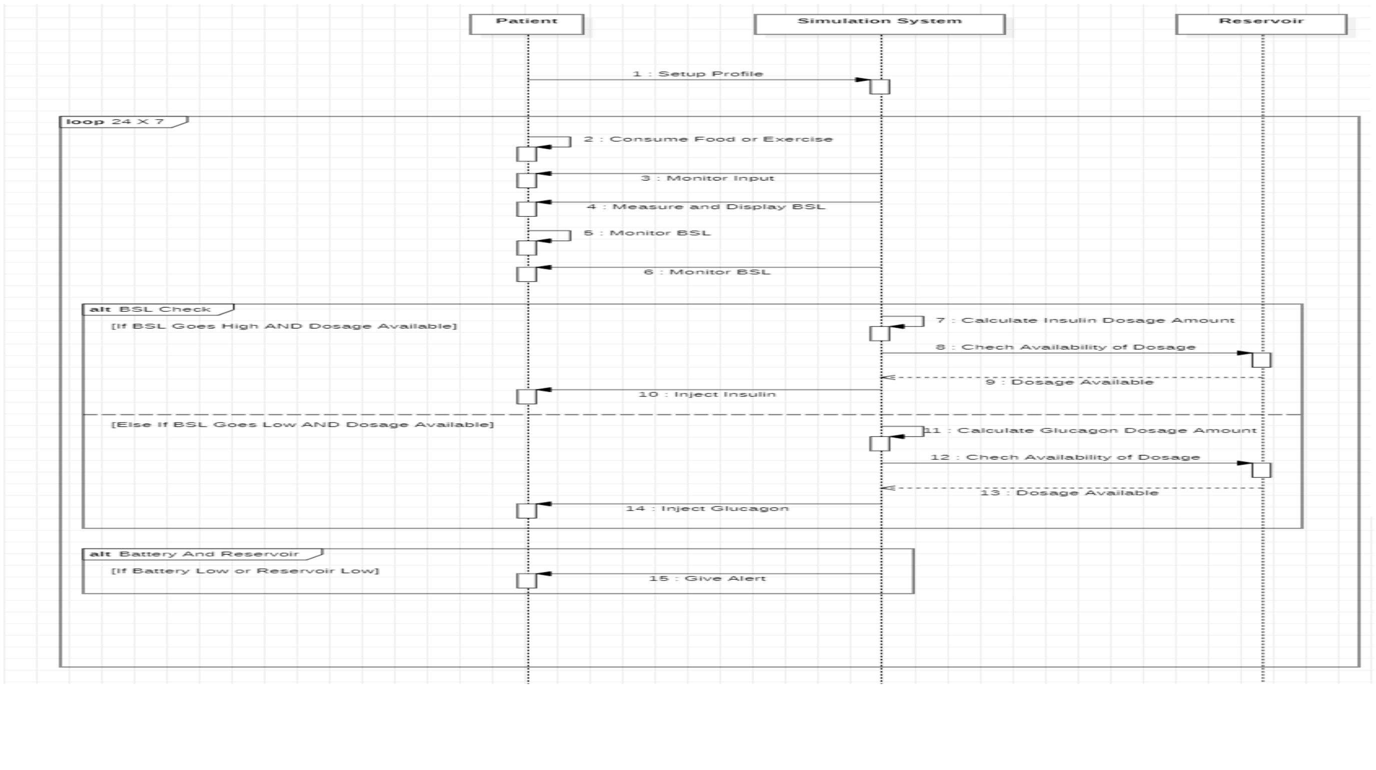
Following are the major use cases of our System:

- Set Up Profile
- Take Input or Be Idle
- Measure and Display Blood Sugar Level
- Monitor Blood Sugar Level
- Calculate Dosage Amount
- Inject Insulin/Glucagon
- Send Email to Dependent
- Monitor and Give Alert

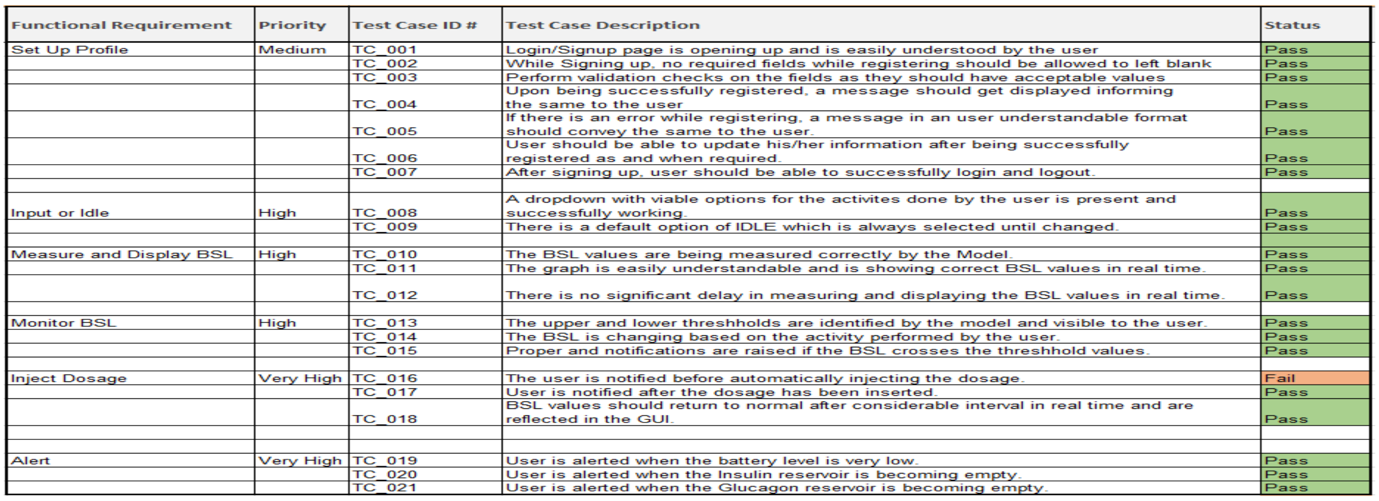
For more details, refer the GitHub Link given in Section 13.



10.3 SEQUENCE DIAGRAM


Patient
Simulation System
Reservoir
1 : Setup Profile
loop 24 X 7
2 : Consume Food or Exercise
3 : Monitor Input
4 : Measure and Display BSL
5 : Monitor BSL
6 : Monitor BSL
alt BSL Check
[If BSL Goes High AND Dosage Available]
7 : Calculate Insulin Dosage Amount
8 : Chech Availability of Dosage
9 : Dosage Available
10 : Inject Insulin
[Else If BSL Goes Low AND Dosage Available]
11 : Calculate Glucagon Dosage Amount
12 : Chech Availability of Dosage
13 : Dosage Available
14 : Inject Glucagon
alt Battery And Reservoir
[If Battery Low or Reservoir Low]
15 : Give Alert

11 VERIFICATION AND VALIDATION

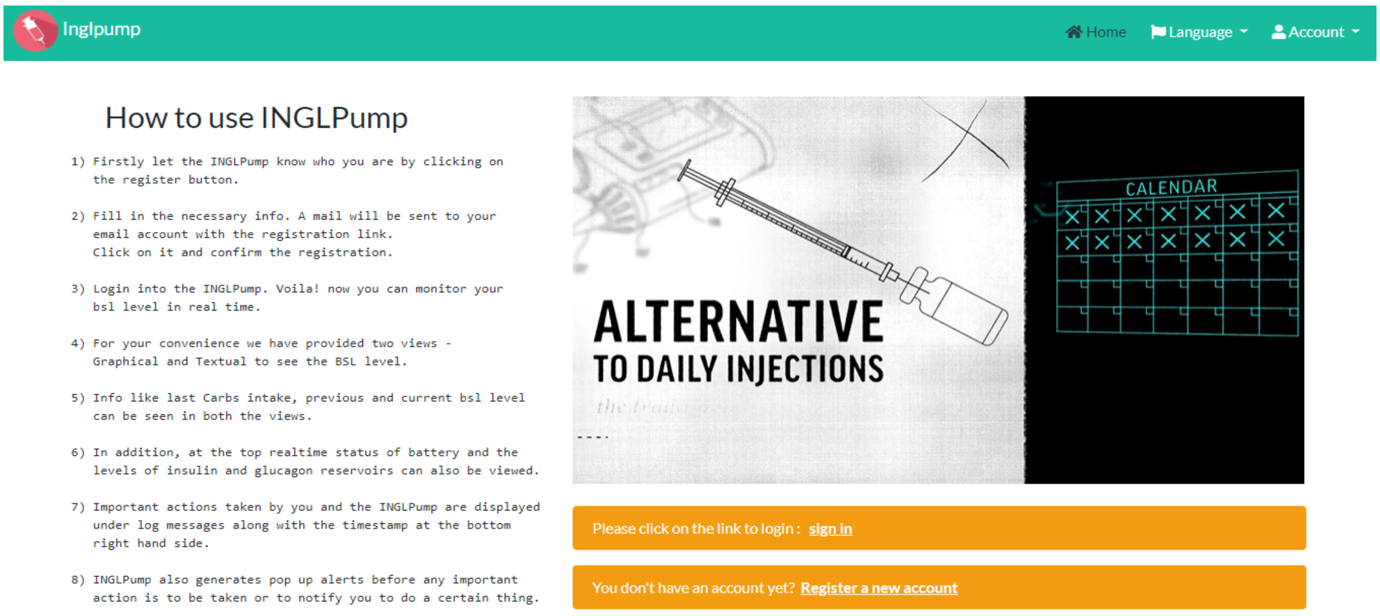


Functional Requirement	Priority	Test Case ID #	Test Case Description	Status
Set Up Profile	Medium	TC_001	Login/Signup page is opening up and is easily understood by the user	Pass
		TC_002	While Signing up, no required fields while registering should be allowed to left blank	Pass
		TC_003	Perform validation checks on the fields as they should have acceptable values	Pass
		TC_004	Upon being successfully registered, a message should get displayed informing the same to the user	Pass
		TC_005	If there is an error while registering, a message in an user understandable format should convey the same to the user.	Pass
		TC_006	User should be able to update his/her information after being successfully registered as and when required.	Pass
		TC_007	After signing up, user should be able to successfully login and logout.	Pass
Input or Idle	High	TC_008	A dropdown with viable options for the activites done by the user is present and successfully working.	Pass
		TC_009	There is a default option of IDLE which is always selected until changed.	Pass
Measure and Display BSL	High	TC_010	The BSL values are being measured correctly by the Model.	Pass
		TC_011	The graph is easily understandable and is showing correct BSL values in real time.	Pass
		TC_012	There is no significant delay in measuring and displaying the BSL values in real time.	Pass
Monitor BSL	High	TC_013	The upper and lower threshholds are identified by the model and visible to the user.	Pass
		TC_014	The BSL is changing based on the activity performed by the user.	Pass
		TC_015	Proper and notifications are raised if the BSL crosses the threshhold values.	Pass
Inject Dosage	Very High	TC_016	The user is notified before automatically injecting the dosage.	Fail
		TC_017	User is notified after the dosage has been inserted.	Pass
		TC_018	BSL values should return to normal after considerable interval in real time and are reflected in the GUI.	Pass
Alert	Very High	TC_019	User is alerted when the battery level is very low.	Pass
		TC_020	User is alerted when the Insulin reservoir is becoming empty.	Pass
		TC_021	User is alerted when the Glucagon reservoir is becoming empty.	Pass

Requirements Traceability Matrix

12 HUMAN MACHINE INTERACTION

- The figure below shows the initial screen which has a help text and instructional video. The patient can login or register from here.



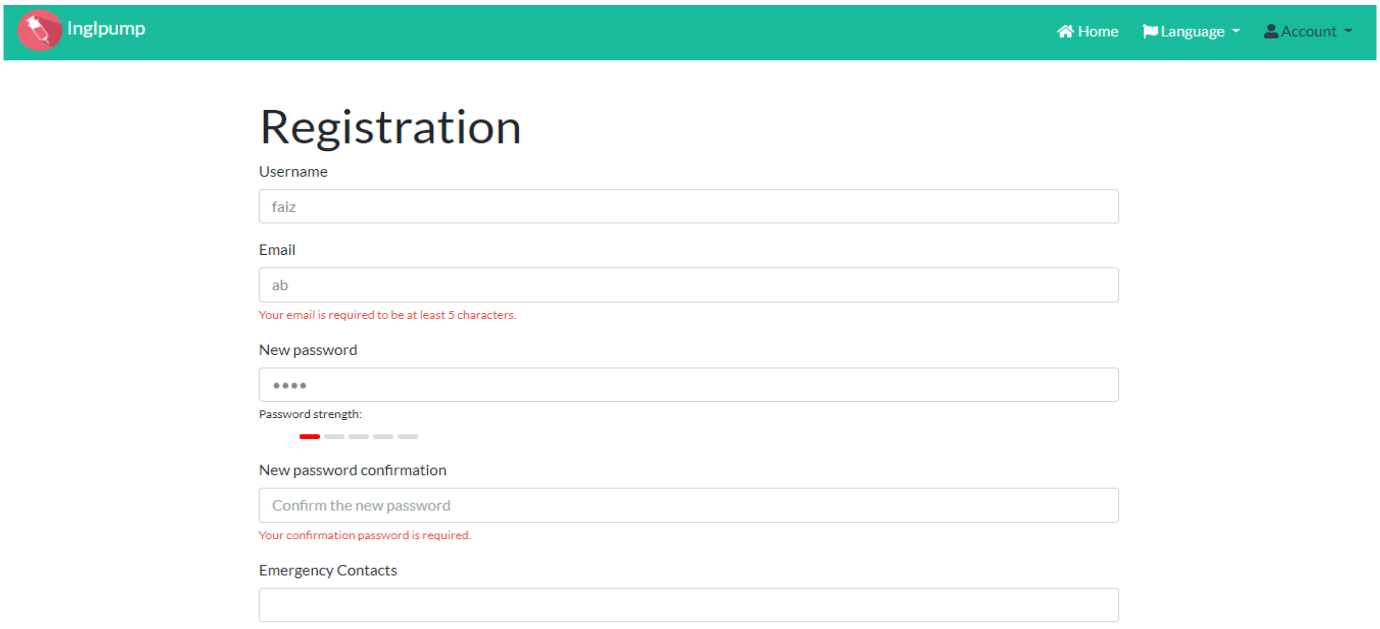
How to use INGLPump

1) Firstly let the INGLPump know who you are by clicking on the register button.
2) Fill in the necessary info. A mail will be sent to your email account with the registration link. Click on it and confirm the registration.
3) Login into the INGLPump. Voila! now you can monitor your bsl level in real time.
4) For your convenience we have provided two views - Graphical and Textual to see the BSL level.
5) Info like last Carbs intake, previous and current bsl level can be seen in both the views.
6) In addition, at the top realtime status of battery and the levels of insulin and glucagon reservoirs can also be viewed.
7) Important actions taken by you and the INGLPump are displayed under log messages along with the timestamp at the bottom right hand side.
8) INGLPump also generates pop up alerts before any important action is to be taken or to notify you to do a certain thing.

Please click on the link to login : sign in

You don't have an account yet? Register a new account

- The figure below shows the registration screen in which the patient has to enter the r elevant information.



Registration

Username

faiz

Email

ab

Your email is required to be at least 5 characters.

New password

••••

Password strength:

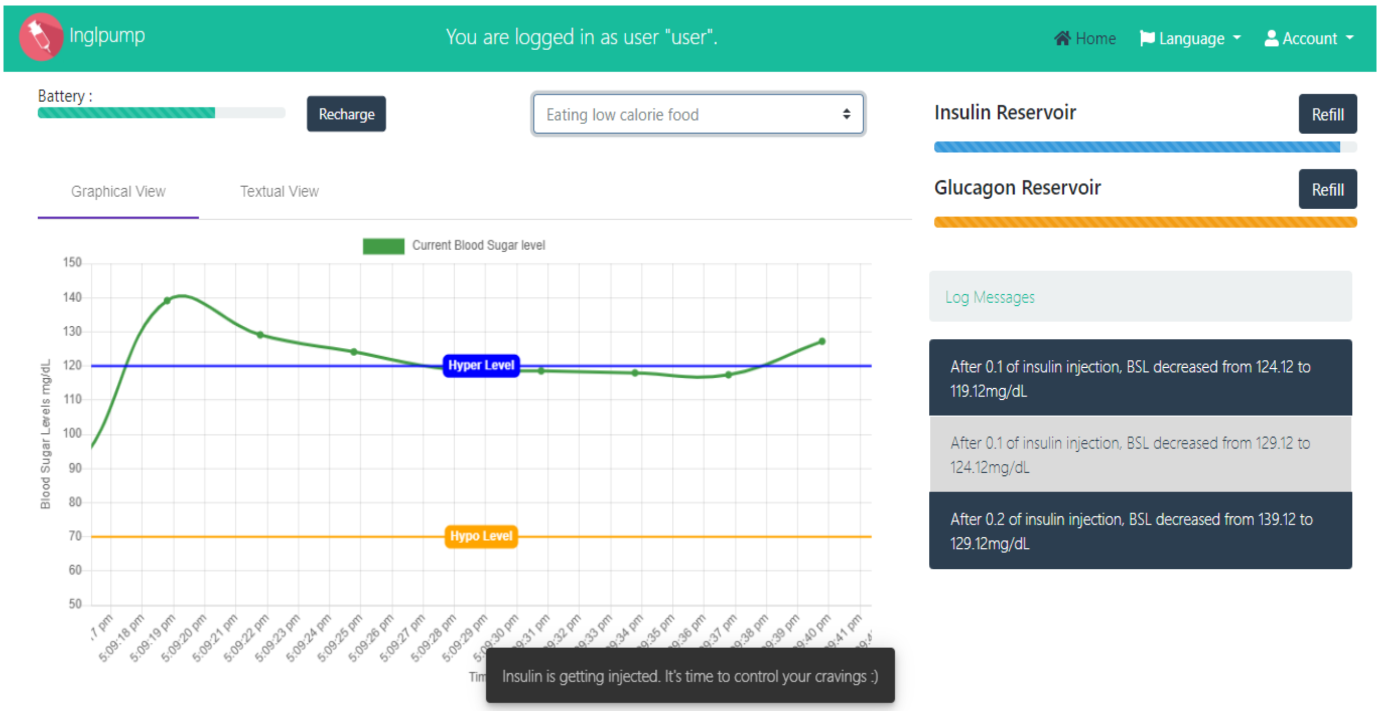
New password confirmation

Confirm the new password

Your confirmation password is required.

Emergency Contacts

- The figure below shows the BSL monitoring screen in the Graphical view. An alert at the bottom is being generated because the bsl level crossed the hyper mark along with the dosage of insulin being injected displayed in the log messages.



You are logged in as user "user".

Eating low calorie food

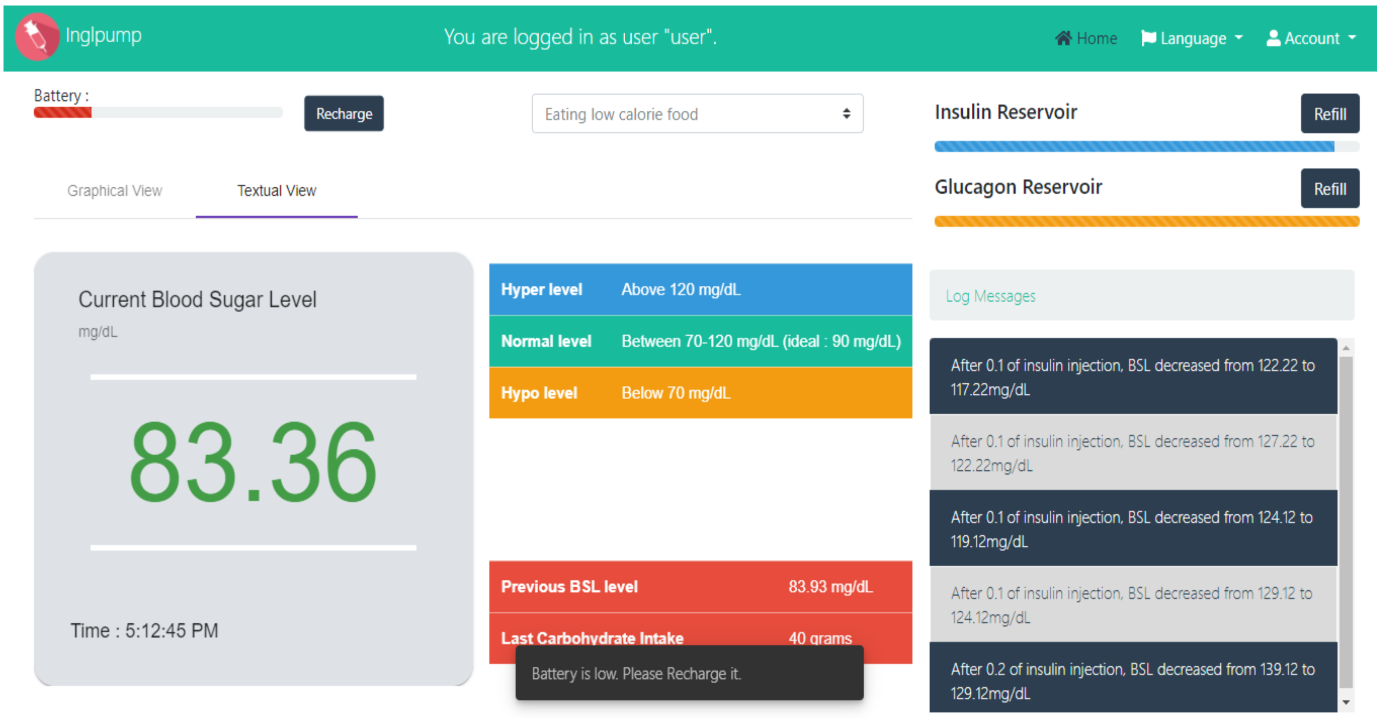
After 0.1 of insulin injection, BSL decreased from 124.12 to 119.12mg/dL

After 0.1 of insulin injection, BSL decreased from 129.12 to 124.12mg/dL

After 0.2 of insulin injection, BSL decreased from 139.12 to 129.12mg/dL

Insulin is getting injected. It's time to control your cravings :)

- The figure below shows the BSL monitoring screen in the Textual view. An alert at the bottom is being generated because the battery level is low.



You are logged in as user "user".

Eating low calorie food

Current Blood Sugar Level
mg/dL
83.36
Time : 5:12:45 PM

Hyper level	Above 120 mg/dL
Normal level	Between 70-120 mg/dL (ideal : 90 mg/dL)
Hypo level	Below 70 mg/dL

Previous BSL level	83.93 mg/dL
Last Carbohydrate Intake	40 grams

Battery is low. Please Recharge it.

```
After 0.1 of insulin injection, BSL decreased from 122.22 to 117.22mg/dL
After 0.1 of insulin injection, BSL decreased from 127.22 to 122.22mg/dL
After 0.1 of insulin injection, BSL decreased from 124.12 to 119.12mg/dL
After 0.1 of insulin injection, BSL decreased from 129.12 to 124.12mg/dL
After 0.2 of insulin injection, BSL decreased from 139.12 to 129.12mg/dL
```

13 SOURCE CODE AND DOCUMENTATION URL

Clich here for GitHub Link: https://github.com/kshitijyelpale/SCS-Simulation-of-Insulin-Pump

14 CONCLUSION

In conclusion, this project was a great learning experience for the entire team. We got an understanding regarding all the different phases of software development life cycle from a safety critical perspective. The work environment was just like regular industry project and there were challenges like deadlines, regular presentations, working in a new team etc. Overall, overcoming these issues helped us grow as professionals and better understand the theoretical concepts that we learned during the course of this lecture.

14.1 FUTURE SCOPE

There is a lot of scope for improvements that could not be implemented in this project due to time and effort constraints, some of them are:

- Multiple perspectives could be implemented, for example: doctor, nurse, care-taker etc
- A manual mode could be introduced for the pump.
- Currently we have only alert notifications, some sound-based alerts could be introduced related to low battery or reservoir levels.
- In case of emergency, email alert could be sent to doctor
- While calculating the blood glucose levels and insulin/glucagon dosage, factors like weight, age, metabolism could be taken into account so that the calculations are more realistic.
- HMI could be made more intuitive and interactive, for example: more messages displayed to the user

15 REFERENCES

[1] Zhensheng Guo, Claudia Hirschmann: “An Integrated Process for Developing Safety-critical Systems using Agile Development Methods”. Available: https://pdfs.semanticscholar.org/f4e5/5be15193e5cea38a
[Access date: 09/02/2020]

[2] Carlos Estela: “Blood Glucose Levels”. Available:
https://scholarcommons.usf.edu/ujmm/vol3/iss2/12/ [Access date: 09/02/2020]