

**DIABETES MANAGEMENT SYSTEM**

INTR 8015 PROJECT IMPLEMENTATION PHASE



**Student ID:** R00076703

**Supervisor:** Karl Grabe

**Second Reader:** Noreen Gubbins

**Date:** 04 January 2015

**Version: 1.1**

**BSc (Hons) in IT Management**

**Department of Computing**

**Declaration**

I hereby certify that this material which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent, that such work has been cited and acknowledged within the text of my work.

I understand that my project documentation may be stored in the library at CIT, and may be referenced by others in the future.

**Signed: ­­­­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Table of Contents**

**PAGE**

**Executive Summary………………………………………4 – 4**

1. **Introduction…………………………………………………5 – 6** 
   1. Project Background**………………………………………**5 – 5
   2. Goals of the System**………………………………….**6 – 6
   3. Objective of the System**……………………………...**6 – 6
2. **Methodology………………………………………....86 – 103** 
   1. Revision Control Software**……………………………**
   2. Eclipse PHP Development Tools**….………………….**
   3. Lifecycle**………………………………………….......**
3. **Analysis and Design…………………………..……...42 – 85** 
   1. Problem Definition**………………………………….**
   2. The Stakeholders**……………………………………**
   3. Project Constraints**………………………………….**
   4. Relevant Facts and Assumptions**……………….......**
   5. Product Boundary Use Case Diagram**……………...**
   6. Project Use Case List**……………………………….**
   7. Project Use Case Narrative**…………………………**
   8. Summary of Prototyping**…………………………....**
      1. Functional Requirements*………….81 – 81*
      2. Non-Functional Requirements*……82 - 82*
   9. Data Dictionary**…………………………………….**
   10. Collaboration Diagrams**……………………………**
   11. Sequence Diagrams**………………………………..**
   12. Business Data Model / Class Diagrams**……………**
   13. Activity Diagram**…………………………………..**
   14. Database Model**……………………………………**
4. **Implementation…………………………….……..106 – 107** 
   1. User Manual**…………………………………**
   2. Validation Testing**…………………..……….**
5. **Evaluation…………………………………….……..106 – 107** 
   1. Revised System Objectives**……………………**
   2. Problems Encountered**…………………………**
   3. OutLook**……………………………………….**
6. **Conclusions…………………………………….……..106 – 107**
7. **Bibliography………………………………………....108 – 108**

**EXECUTIVE SUMMARY**

Over the past 50 years, there has been a dramatic increase in the prevalence of interrelated metabolic disease states such as obesity, insulin resistance, and type 2 diabetes. Diabetes Mellitus is a group of metabolic diseases characterized by high blood sugar (glucose) levels that result from defects in insulin secretion, or its action, or both. Diabetes occurs as a result of inability of cells to use insulin properly and efficiently or insufficient production of insulin. In other words, it is a common condition that affects millions of people worldwide. For instance, if diabetes is not treated, it can lead to serious complications such as blindness, amputations, and even death. However, before people develop diabetes, they always develop pre-diabetes. Healthy people and people with pre-diabetes may be able to prevent or delay type 2 diabetes from developing.

In order to prevent the risks of serious complications that can be caused by diabetes, the blood glucose level must be monitored. The blood glucose level or blood sugar concentration is the amount of glucose (sugar) present in the blood of a human. Therefore, to monitor the blood glucose level in the blood an online application known as the **Diabetes Management System** (DMS) is developed.

Diabetes Management System is a powerful and efficient way to track and monitor the management of diabetes over time. The system will be developed as an online application, where patients and doctors can track, monitor, and keep an accurate reading of blood glucose level tests.

The Diabetes Management System (DMS) will assist in the effective management of diabetes. It will provide a means of tracking, organising, and managing all of the activities involved in monitoring diabetes. DMS will work in parallel with the existing paper-based system i.e. the Diabetes Monitoring Diary. The DMS will make it possible to track patient’s data stored in the paper-based system. Moreover, the new system (DMS) can be used to replace the existing system (DMD), in order to make monitoring of diabetes easy for people with diabetes. See below.

**EXISTING** **NEW**

PAPER-BASED WEB-BASED

Mary Kate Mary Kate

**Existing System** – Diabetes Monitoring Dairy (DMD)

**New System** – Diabetes Management System (DMS)

1. **INTRODUCTION**
   1. PROJECT BACKGROUND

As a final year student in Information Technology Management (ITM), I undertook the effort to develop an interactive web-based Diabetes Management System. The application provides an easily-accessible means of managing an Endocrinology/Diabetes and its various Patients, Doctors and Nurses. This system is accessible by all stakeholders and mediates access via a security system. Diabetes Management System will transform the paper-based filling system i.e. Diabetes Monitoring Diary to a web-application system and enable simultaneous multi-user access to a central database.

My inspiration for the Diabetes Management System came from my Mum. She was recently diagnosed with TYPE1 Diabetes. As a result of this, she needs to take insulin by injection, measure her daily blood glucose level and record the result of her blood glucose levels in her Diabetes Monitoring Diary. The diary enables her to keep a record of her daily blood glucose level and any appointments with her diabetes nurse. Moreover, the information in the diary helps her nurse to make the right adjustments to her insulin doses.

In order to curb society’s risk for many serious health problems caused by Diabetes, I have decided to develop a **Diabetes Management System** (also known as the Diabetes Monitoring Application) for South Terrace Cork Hospital. The Diabetes Management System will be used for monitoring daily blood glucose level, making the right adjustments to insulin doses and keeping a record of any appointments with the diabetes nurse or doctor.

South Terrace Cork Hospital is the name of the business chosen to help gather more information on the requirement specifications needed for the Diabetes Management System. It is one of the leading diabetes clinics in Ireland, with the main clinic located in Northwood Business Campus Santry, Dublin 9 and seven other clinics located throughout Ireland. With the newly built System, the business will be able to provide an effective management of co-morbidities and diabetes for individuals and all groups at risk of developing health issues.

In addition, the system will be designed for hospitals and healthcare providers with the intention of preventing or delaying the short and long term complications caused by Diabetes. For instance, when a doctor defines a treatment plan for a diabetic based on his/her diabetes type, sex, age, etc. factors. The Diabetes Management System (DMS) can be used to send a reminder to patients to ensure better compliance with the treatment plan. And, once the patients, undergo tests, the data such as blood glucose level and insulin injections, is captured in the system. Then, doctors can view the progress of individual patient and take necessary interventions if needed.

* 1. GOALS OF THE SYSTEM
     1. The system will enable patients to enter their personal information.
     2. The system will be able to keep track of the patients’ insulin doses and any adjustments made by a doctor or nurse.
     3. The system will enable patients to update and view their personal information in real-time.
     4. The system will enable patients to keep track of their blood glucose level remotely. This process will reduce the time spent in filling out this information that is driven by the paper-based system i.e. the diabetes monitoring dairy.

With the Diabetes Management System, patients will be able to keep track of their blood glucose level readings at any time.

* + 1. The system will enable healthcare professionals to make appropriate insulin adjustments for their patients.
    2. The system will enable patients to leave comments and personal notes.
    3. The system will enable patients and doctors to register into the Diabetes Management System.
  1. OBJECTIVE OF THE SYSTEM
     1. The plan for the new system is to provide the capability for patients to update data in real-time. This process will reduce, if not eliminate some of the current problems that people with diabetes face i.e. patients’ blood glucose level test results will be recorded in the system, which will be up-to-date.
     2. To provide world ready Diabetes Management System that can be adapted to various purposes. The software will make diabetes management easier and more efficient than ever before.
     3. The aim is to design an online application that will be integrated into a mobile application. This application will help people with Type 1 diabetes and Type 2 diabetes manage the data associated with:

1. Records of actual insulin delivery
2. Statistical Evaluation.
3. Manual log entries for exercise and other factors.
4. Coaching of dose corrections.
5. Blood glucose level results from a glucose meter.
6. **METHODOLOGY**

In order to produce a world ready application, a combination of iterative and prototype-based development **methodologies** is used during the development of the Diabetes Management System.

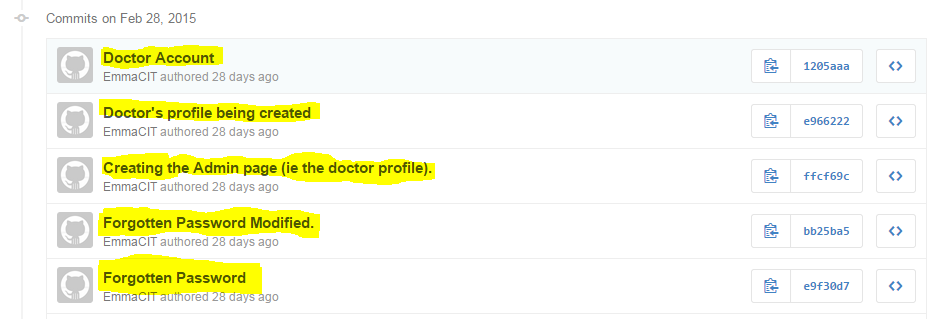
The **tools** that enabled the process for the completion of the project are as follows:

**Revision Control Software**

The revision control software also known as the source control software is a management tool used to help me build better software and also provided a great opportunity to work efficiently. I decided to use ***Git***, a distributed source control system, and its companion website ***GitHub***. GitHub is an instance of a hosting service that uses the Git revision control system. GitHub is a powerful mechanism tool that provided numerous features, which made code management effortless. Moreover, it is a web-based hosting service for software projects.

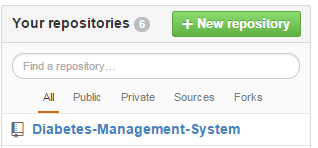
The basic unit of work in GitHub is the **commit**. A commit is a code change with an accompanying comment by its author. Whenever a change takes place, the details are immediately published on the GitHub website: who made the changes, when it was made, what lines were changed, and many more. And, a comprehensive history maintained with all the details of the changes made in an easily accessible way.

An example of this is shown below:



*My Commits published on GitHub*

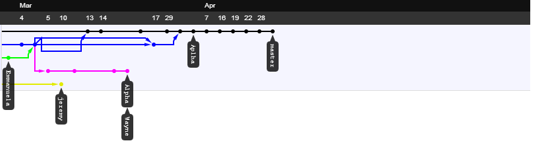
A collection of source code and its associated history is called a **repository** also known as **‘repo’** in GitHub. In order words, I created a repository that gave me full access, making it possible for me to make necessary changes at my spare time by **pushing** commits to the Git. In addition, these are simple commands triggered by a button press in the development tools.



*My Repository* ***“Diabetes-Management-System”*** *created on GitHub*

One of Git’s most important features is the ability to create ***branches***. Branches in GitHub are concurrent lines of development independent of the ‘**master**’ code, in which changes can be made without the fear of interference. Sometimes, I created a new branch for each feature, and worked on that branch at a time. If a feature does not work out, it was abandoned with no consequences. This process encouraged rapid experimentation and development of the project.

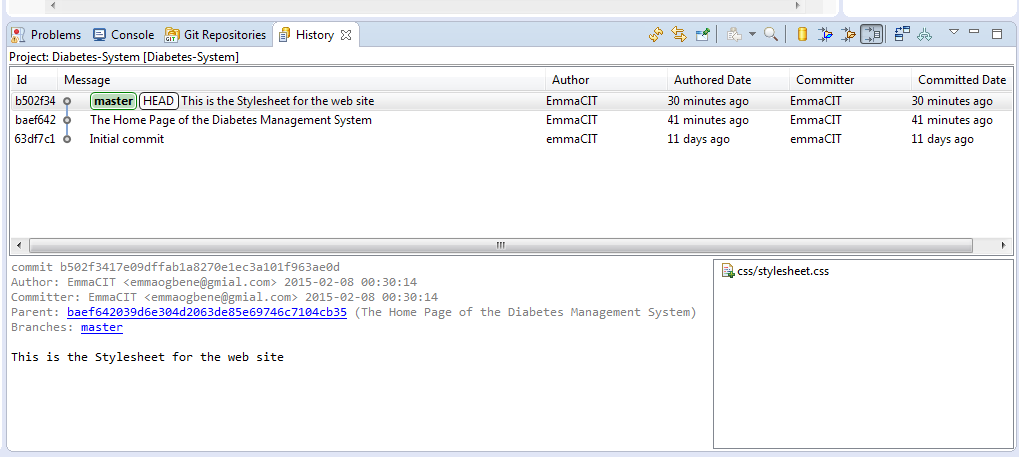
The most helpful feature of GitHub is ***merging***. When a feature is completed and tested, its branch can be merged back into the master branch to integrate the changes into it. The process of merging is mostly automatic, and conflicts are easily resolved manually. Since branches diverge from the master codebase, over time they grow out of sync as other work is completed and integrated. But merging the work in both ways (i.e. by merging the master branch into the feature branch), it can be brought up to date with the latest changes, without any interference to the main codebase.



*An excerpt from repository history:* ***dots are commits, lines are branches, and arrows signify merges***

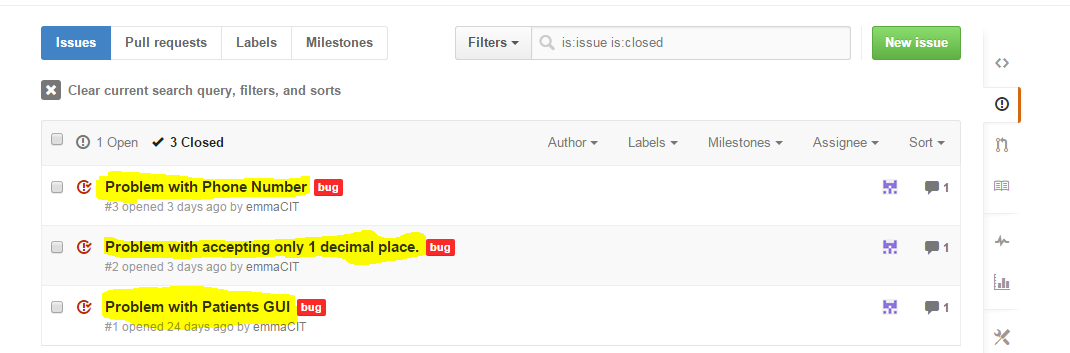
**Eclipse PHP Development Tool**

Git is completely integrated into Eclipse PHP, the development environment that I used for the project. It integrates all of Git’s functionality in a way that is quite simple to use.



*E.g. of Commits and branches in Eclipse’s history view*

In conclusion, GitHub provides an issue-tracking system which is integrated with the commit log. This was used to keep track of bugs I founded and to specify the features I needed. The issue tracker is also integrated with the commit history – commits can be marked as related to a given issue and they show up on the issue page on the website. This made it very simple for me to see what I have done about each problem as they are encountered and fixed.



*E.g. of a GitHub’s issue tracker*

**Life cycle**

During development of the application, I undertook an iterative/prototype based approach. One of the greatest aspects of using this model is that, anyone can see the development of the software on a step by step scale. The whole process is iterative and made it easy for me to demonstrate things even when some of the materials were temporary. It was easier to change and upgrade.

Additionally, both incremental and iterative model are main procedures in the chosen methodology, which was used in the development cycle of the Diabetes Management System. Underlying features required were developed in separate branches, tested, and then merged when it is suitable to be integrated into the main branch.

To sum up what I did, it closely relates to the iterative and incremental model.



The model allowed the addition of new features to be added quickly and easily.

For a specific instance of this process, the primary GUI components were considered: the tabs that were used to manipulate information in the database.

I started with the requirements laid out in the report. This provided designs for the appearance and general functionality of each tab. The functional requirements determined what was needed to be done at a finer level of detail.

Branches for each tab was created and implemented. Multiple tabs were developed simultaneously in different branches. As each tab was completed and tested in isolation, the related branch was merged into the master branch and integration testing was carried out. The design is quite elegant, featuring many reusable components.

A complete history of all development activity is visible on the Git repository on GitHub, line by line.

**Tools Used**

For Source control I used ***Git***.

The integrated development environment of choice was ***Eclipse PHP***, which has full Git integration. This is also an environment that I have used before for my programming modules.

For bug tracking, feature requests and staying up to date on development, I used ***GitHub***.

I used ***Google Docs*** and ***Microsoft Word*** for user document editing. This had already proven reliable in the past.

The HTTP server running on the Linux machine is ***Apache 2***.

The database server runs ***MySQL***.

For Unified Modelling Language (UML) diagrams, I used ***Visual Paradigm***. It is a software design tool tailored for agile software projects. Also, it supports BPMN, UML, DFD, ERD, and SysML. Visual Paradigm was used to model the Use Case Diagrams, Activity Diagrams, Sequence Diagrams, Collaboration Diagrams, Database Model and Domain Model of the Diabetes Management System.

For mock-up diagrams, I used ***Pencil***, which is a free and open-source GUI prototyping tool that people can easily install and use to create mock-ups in popular desktop platforms. It was used to mock-up the graphical user interface (GUI) design of the Diabetes Management System.

For the development of the application, the latest version of ***XAMPP*** was downloaded and installed. It consists of the following components such as **Apache HTTP Server, MySQL database, PHP Programming Language, Perl Programming Language, FTP Server** and **SMTP Server**. XAMPP helps in the installation process of the named components to be quick and easy. For instance, installing the components individually, factors such as PHP configuration and the setup of MySQL will not have to be considered. XAMPP is a free and open source cross-platform web server solution stack package.

***GoDaddy*** was used to host the Diabetes Management System. It is a publicly traded Internet domain registrar and web hosting company. GoDaddy sells e-business related software and services. It makes registering Domain Names fast, simple, and affordable. Hosting the application allows anyone to access the application anytime and anywhere, through internet connection.

Of all the tools I used, only GoDaddy.com, Visual Paradigm and Google Docs were proprietary. The rest are free and open source.

1. **ANALYSIS AND DESIGN**
   1. **PROBLEM DEFINITION**

Diabetes Management System is an online application that assists in the management of Diabetes. It provides a means of tracking, monitoring, organising, and managing all the activities of a Patient, Doctor and any other Stakeholders. The application allows for easy-system-evolution to accommodate expanding of the business. The application provides an easy to use user interface containing all the features of the system. Most importantly, the application has a helpful user-manual for the end-users.

In addition, the new system works in parallel with the existing paper-based filling system. It makes it possible to track patients’ data such as their blood glucose level records stored in paper-based system.

To facilitate this several features must be provided:

***Doctor*** must be registered before he/she can access the system. The Doctor must be able to login and add his/her patients. The Doctor must be able to view his/her patients’ blood glucose level records, adjust their insulin doses and notify them to come in for check-up. The Doctor sees his/her personal information. The Doctor also maintains reports regarding any information such as patient’s blood glucose level readings. The Doctor is responsible for scheduling any appointments with patients through email notifications.

***Patient*** must register before being authorized to use the system. After registering, the Patient can view his/her personal information, update their profile information as well as change their password. They can record their blood glucose level, view their records, update as well as delete their records. The Patient can leave personal notes. The Patient can book an appointment by emailing their doctor to schedule a date and time for them. Furthermore, the Patient can use the blood glucose level guidelines for assistance when reading their measurements. They should be able to track and monitor their blood glucose level statistics using a bar chart.

**Feature Overview**

In summary, the diabetes management system should help in the tracking and monitoring of the following:

* Allowing patients and doctors to register into the system.
* Logging registered patients and doctors to the diabetes management system.
* Allowing patients to view and update their personal information.
* Patients uploading their profile image.
* Allowing users to change their password and recover their username and password.
* Patients recording and viewing their blood glucose level measurements as well as updating and deleting their records.
* Patients leaving, updating and deleting their personal notes.
* Providing information guidelines for patients to use when recording their blood glucose level readings.
* Allowing patients to view their blood glucose level statistics.
* Patients booking appointment to see their Doctor via email.
* Enabling the tracking of all doctors’ information.
* Doctors maintaining reports such as patients’ blood glucose level measurements and their insulin doses.
* Allowing doctors to make adjustments to patients’ insulin doses.
* Enabling doctors to notify their patients to come in for check-ups via email.
* Doctors viewing his/her patients’ blood glucose level measurements in order to make changes to their diet, exercise or insulin doses.
  1. **THE STAKEHOLDERS**
     1. The Client Dr Shirley Cotter, Manager of Diabetes Clinic Ltd.
     2. The User
        1. Manager
        2. Doctor
        3. Nurse
        4. Patient
        5. Secretary
     3. Other Stakeholders
        1. The Cork University Hospital
        2. The Diabetes Day Centre
        3. The Clinical Nutritionist
        4. The Eye Clinic
        5. The Diabetes Federation of Ireland
        6. The Diabetes Ireland Care Centre
     4. Maintenance Users and Service Technicians
        1. The Administrator
        2. IT Department Staff
  2. **PROJECT CONSTRAINTS**
     1. **Solution Constraints**

The product shall record patient’s blood glucose level, insulin injections, and blood pressure information and be able to retrieve the data to both the patient and the doctors’ clinic.

* + 1. **Integration Constraints**

Diabetes Management System shall provide an easily-accessible means of managing a Diabetes Clinic and its various patients, doctors, nurses, and secretary. The system is accessible by all users and mediates access via a security system.

The system enables simultaneous multi-user access to a secured central database.

* + 1. **Budget Constraints**

The budget for the project is €80,000, excluding costs incurred for attending regular meetings.

* + 1. **Schedule Constraints**

The system must be ready in 3 months, excluding the period of research. The software application must be implemented and tested before a final session of software testing. Furthermore, the application must work on majority of web-browsers such as **Internet Explorer**, **Google Chrome**, **Mozilla Firefox**, **Opera**, **Netscape**, and **Safari**, installed on any PCs with Windows, Vista, and Linux operating systems.

* 1. **Relevant Facts And Assumptions**
     1. **Relevant Facts**

Diabetes is now the commonest metabolic illness in the world. Meanwhile, Ireland has a growing population of Diabetes and is increasing at an alarming rate in tandem with rising obesity rates.

County Tipperary has the highest rate of diabetes people with 2.5% and County Kerry has the smallest number of diabetes people with 1%.

Moreover, research indicates that 9.6% of Irish population now have the disease. Diabetes increases the risk for many serious health problems such as Heart attack, Stroke, Kidney failure, Blindness, etc. However, with correct treatment and recommended lifestyle changes, many people with diabetes are able to prevent or delay the onset of complications.

* + 1. **Assumptions**

The graphical user interface (GUI) design must be available to the developers.

The chosen business must agree for the software to be developed by signing a contract.

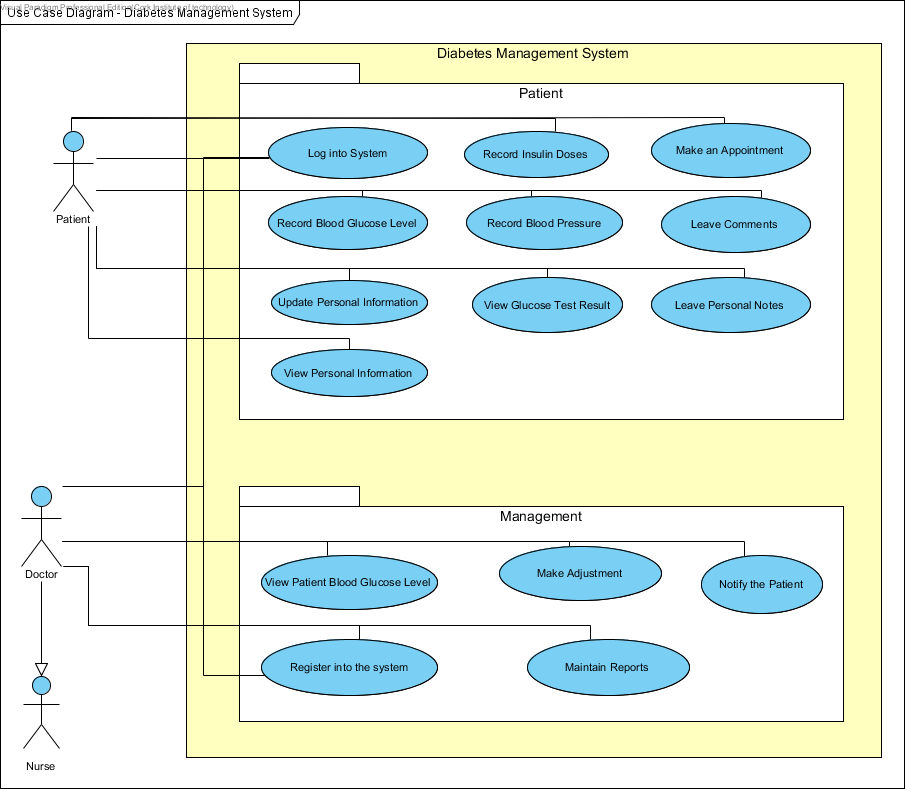
* 1. **PRODUCT BOUNDARY USE CASE DIAGRAM**

The Product Scope diagram identifies the boundaries between the users (Actors) and the Product. This diagram is a summary of all the Product Use Cases.

The Diabetes Management System use case diagram illustrated below shows the actors/users outside the product boundary (i.e. the rectangle). While the product Use Cases (PUCs) are the ellipses inside the boundary. And, the lines denote interfaces between the Product and an Actor. The actors can be either automated or human.

The most important aspect in modelling a system is to capture the dynamic behaviour. In order to capture this, I used Visual Paradigm to model the system of our application. This use case diagram consists of the actors, use cases and their relationships.

The diagram below (**See next page**) is used to gather the functional and non-functional requirements of the Diabetes Management System.



***Use Case Diagram for Diabetes Management System***

* 1. **PRODUCT USE CASE LIST**

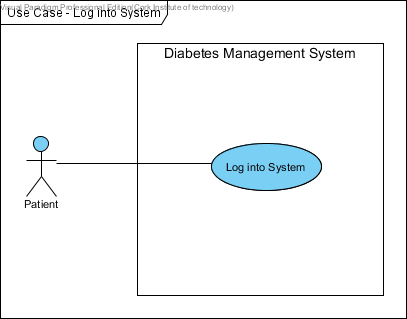
|  |  |
| --- | --- |
| **Use-Case Name** | **Brief Summary of Each Use-Case** |
| Log into System | This use case describes the Login process. |
| Update Personal Information | This use case describes the event of updating the patient and doctor profile information. |
| View Personal Information | This use case describes the event of viewing patients’ complete profile information. |
| Record Blood Glucose Level | This use case describes the event of documenting patients’ Blood Glucose Level. |
| Record Insulin Doses | This use case describes the event of documenting patients’ Insulin Doses. |
| View Blood Glucose Level Results | The use case describes the event of viewing blood glucose level results. |
| Record Blood Pressure | This use case describes the event of documenting patients’ blood pressure. |
| Make an Appointment | This use case describes the event of making an appointment to see the doctor and keeping a record of any appointments with the doctor. |
| Leave Personal Notes | This describes the process of leaving personal drafts, appointment dates, emergency contacts, etc. |
| Leave Comments/Special Events | This use case describes the process of leaving special events/comments based on the patients’ Blood Glucose Level measurement. |
| View Patient Blood Glucose Level Measurements | This use case describes the event of viewing patient’s blood glucose level records. |
| Register into the system | This use case describes the process of patients’ registering into the system. |
| Make Adjustments | This use case describes the event of making adjustments to patients’ Insulin Doses. |
| Notify Patients | This use case describes the event of notifying patients’ to come in for check-ups. |
| Maintain Reports | This use case describes the event of recording and maintaining any information such as patients’ record of blood sugar readings, record of Insulin adjustments, record of blood pressure, list of Insulin doses, etc. Also, it includes updating and deleting reports. |

* 1. **PRODUCT USE CASES NARRATIVE**

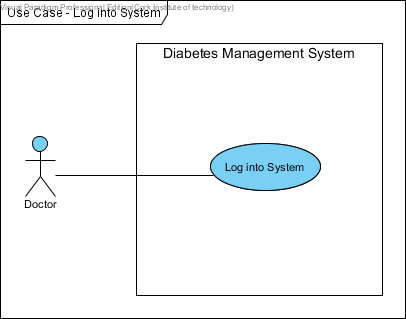
**Use-Case Narrative (Log into System)**

Full Description of use case ***Log into System***

|  |  |  |
| --- | --- | --- |
| **Use-Case #** | **1** | |
| **Use Case Name:** | Log into System | |
| **Scenario:** | User wants to log into the Diabetes Management System with user credentials (e.g. username and password). | |
| **Event (Trigger):** | User wants to enter into the Diabetes Management System to view or update their profile. | |
| **Brief Description:** | User provides required credentials. System validates credentials and logs user into the Diabetes Management System. | |
| **Actors:** | Patient, Doctor, Nurse. | |
| **Type:** | Essential | |
| **Preconditions:** | Patient, Doctor and Nurse are known by the System | |
| **Post conditions:** | Patient, Doctor and Nurse logged into the system. | |
| **Flow of Events**  **(Steps):** | **Actor Action** | **System Response** |
| 1. This use case begins when a user desires to log into the system. |  |
| 1. The user provides the username and password. | * 1. System validates the entered username and password and logs the user into the system. |
| **Alternative Flow of Events** | | |
| **Line 2.1:** Invalid username and password entered. Indicate error. Return to Step 2. | | |



***Patient Log into System***

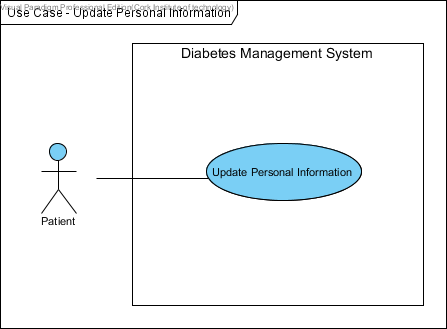


***Doctor Log into System***

**Use-Case Narrative (Update Personal Information)**

Full Description of use case ***Update Personal Information***

|  |  |  |
| --- | --- | --- |
| **Use-Case #** | **2** | |
| **Use Case Name:** | Update Personal Information | |
| **Scenario:** | User wants to update his/her profile information in the system. | |
| **Event (Trigger):** | User wants to update personal information. | |
| **Brief Description:** | User provides information (e.g. name, address, telephone number). System updates the personal information. | |
| **Actors:** | Patient, Doctor and Nurse. | |
| **Type:** | Essential | |
| **Preconditions:** | Patient, Doctor and Nurse are known by the System  Patient, Doctor and Nurse information must exist. | |
| **Post conditions:** | Patient, Doctor and Nurse information updated in the system. | |
| **Flow of Events**  **(Steps):** | **Actor Action** | **System Response** |
| 1. This use case begins when a user desires to update their personal information in the system. |  |
| 1. The user provides the username and password. | 2.1 System validates username and password and logs the user into the system.  2.2 System displays user’s personal information. |
| 1. The user modifies their personal information. (E.g. address, name, telephone number). | 3.1 System updates and displays updated user’s personal information. |
| **Alternative Flow of Events** | | |
| **Line 2.1:** Invalid username and password entered. Indicate error. Return to Step 2. | | |

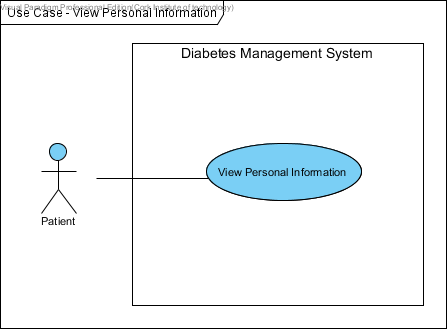


***Patient Update Personal Information***

**Use-Case Narrative (View Personal Information)**

Full Description of use case ***View Personal Information***

|  |  |  |
| --- | --- | --- |
| **Use-Case #** | **3** | |
| **Use Case Name:** | View Personal Information | |
| **Scenario:** | Patient wants to view his/her personal information. | |
| **Event (Trigger):** | User wants to view his/her personal information | |
| **Brief Description:** | Patient provides Login details. System displays patient’s personal information. | |
| **Actors:** | Patient and Doctor | |
| **Type:** | Essential | |
| **Preconditions:** | Patient and Doctor are known by the system.  Personal information must exist and be up to date. | |
| **Post conditions:** | None | |
| **Flow of Events**  **(Steps):** | **Actor Action** | **System Response** |
| 1. This use case begins when a patient wants to view their personal information in the system. |  |
| 1. The patient provides username and password. | * 1. System validates username and password and logs the patient into the system.   2. System displays patients’ personal information. |
| 1. Patient views their personal information. |  |
| **Alternative Flow of Events** | | |
| **Line 2.1:** Invalid username and password entered. Indicate error. Return to Step 2.  Patient not registered. Inform the Patient. Return to Step 2. | | |

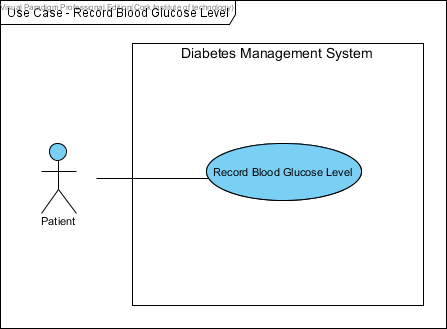


***Patient views Personal Information***

**Use-Case Narrative (Record Blood Glucose Level)**

Full Description of use case ***Record Blood Glucose Level***

|  |  |  |
| --- | --- | --- |
| **Use-Case #** | **4** | |
| **Use Case Name:** | Record blood glucose level | |
| **Scenario:** | This is the process whereby a patient wants to record his/her blood glucose level readings. | |
| **Event (Trigger):** | This use case is initiated when the patient selects this option from the User Interface. | |
| **Brief Description:** | Patient provides their blood glucose level readings. The system records patients’ blood glucose level readings. | |
| **Actors:** | Patient | |
| **Type:** | Essential | |
| **Preconditions:** | Patient is known by the system. | |
| **Post conditions:** | Blood glucose level is recorded in the system. | |
| **Flow of Events**  **(Steps):** | **Actor Action** | **System Response** |
| 1. This use case begins when a patient wants to record their blood glucose level. |  |
| 1. The patient provides username and password. | * 1. System validates username and password and logs the patient into the system. |
| 1. Patient enters the blood glucose level (mmol/l) information. This includes: before breakfast, 2 hours after breakfast, before lunch, 2 hours after lunch, before dinner, 2 hours after dinner, before supper or bed, during the night. | * 1. The system validates the information.   2. Patient’s blood glucose level readings are added into the system. |
| **Alternative Flow of Events** | | |
| **Line 2.1:** Invalid username and password entered. Indicate error. Return to Step 2.  Couldn’t find patient. Inform the patient. Return to Step 2.  **Line 3.1:** Invalid or Insufficient Information. Inform the patient. Return to Step 3. | | |

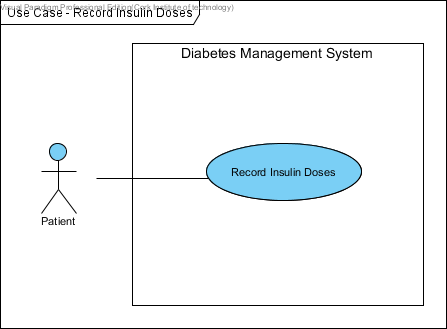


***Patient records blood glucose level***

**Use-Case Narrative (Record Insulin Doses)**

Full Description of use case ***Record Insulin Doses***

|  |  |  |
| --- | --- | --- |
| **Use-Case #** | **5** | |
| **Use Case Name:** | Record Insulin Doses | |
| **Scenario:** | This is the process whereby a patient wants to record his/her insulin dose. | |
| **Event (Trigger):** | This use case is initiated when the patient selects this option from the User Interface. | |
| **Brief Description:** | Patient provides their insulin doses. System keeps track of the patients Insulin Doses. | |
| **Actors:** | Patient | |
| **Type:** | Essential | |
| **Preconditions:** | Patient is known by the system. | |
| **Post conditions:** | Insulin Doses recorded in the system. | |
| **Flow of Events**  **(Steps):** | **Actor Action** | **System Response** |
| 1. This use case begins when a patient wants to record their insulin doses. |  |
| 1. The patient provides username and password. | * 1. System validates username and password and logs the patient into the system. |
| 1. Patient enters the insulin doses (units) information, which includes: type of insulin, pre-Breakfast (am), pre-Lunch (am), pre-Dinner (pm), before Bed (pm). | * 1. The system validates the information.   2. Patients’ insulin dose is added to the system. |
| **Alternative Flow of Events** | | |
| **Line 2.1:** Invalid username and password entered. Indicate error. Return to Step 2.  **Line 3.1:** Invalid or Insufficient Information. Inform the patient. Return to Step 3. | | |
|  | | |

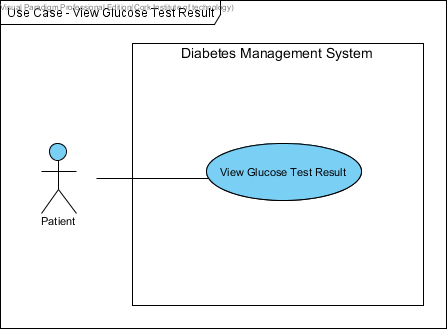


***Patient records Insulin Doses***

**Use-Case Narrative (View Blood Glucose Level Results)**

Full Description of use case ***View Blood Glucose Level Results***

|  |  |  |
| --- | --- | --- |
| **Use-Case #** | **6** | |
| **Use Case Name:** | View Blood Glucose Level Results | |
| **Scenario:** | Patient wants to view his/her blood glucose level results. | |
| **Event (Trigger):** | A Patient wants to view his/her blood glucose level results. | |
| **Brief Description:** | Patient requests to view blood glucose level results. System displays patients’ blood glucose level results. | |
| **Actors:** | Patient | |
| **Type:** | Essential | |
| **Preconditions:** | Patient is known by the system.  Blood Glucose Level results must exist and be up to date. | |
| **Post conditions:** | None | |
| **Flow of Events**  **(Steps):** | **Actor Action** | **System Response** |
| 1. This use case begins when a patient wants to view his/her blood glucose level results. |  |
| 1. The patient provides username and password. | * 1. System validates username and password and logs the patient into the system. |
| 1. Patient requests to view their blood glucose level results. | * 1. System displays patient’s blood glucose level results table. |
|  | 1. Patient views the blood glucose level results table and analyses the results. |  |
| **Alternative Flow of Events** | | |
| **Line 2.1:** Invalid username and password entered. Indicate error. Return to Step 2.  **Line 3.1:** Blood glucose level results not up-to-date. Inform the patient. Return to Step4. | | |

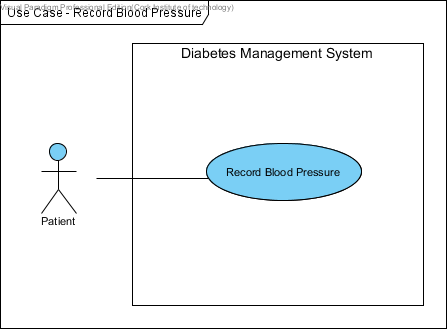


***Patient views Blood Glucose Level Results***

**Use-Case Narrative (Record Blood Pressure)**

Full Description of use case ***Record Blood Pressure***

|  |  |  |
| --- | --- | --- |
| **Use-Case #** | **7** | |
| **Use Case Name:** | Record Blood Pressure | |
| **Scenario:** | This is the process whereby a patient wants to record his/her blood pressure. | |
| **Event (Trigger):** | A Patient wants blood pressure readings to be added to existing records. | |
| **Brief Description:** | Patient provides blood pressure readings. System updates the patients’ blood pressure records. | |
| **Actors:** | Patient | |
| **Type:** | Essential | |
| **Preconditions:** | Patient is known by the system. | |
| **Post conditions:** | Blood pressure readings added in the system. | |
| **Flow of Events**  **(Steps):** | **Actor Action** | **System Response** |
| 1. This use case begins when a patient wants to record their blood pressure. |  |
| 1. The patient provides username and password. | * 1. System validates username and password and logs the patient into the system. |
| 1. Patient enters their blood pressure test result. | * 1. The system validates the information.   2. Patients’ blood pressure is added to the system. |
| **Alternative Flow of Events** | | |
| **Line 2.1:** Invalid username and password entered. Indicate error. Return to Step 2.  **Line 3.1:** Invalid or Insufficient Information. Inform the patient. Return to Step 3. | | |

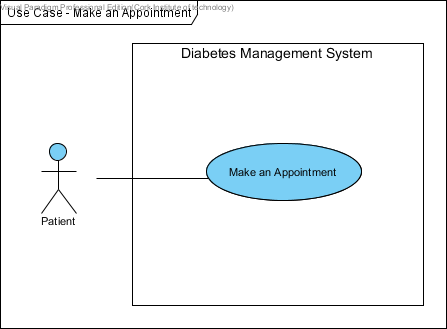


***Patient records Blood Pressure***

**Use-Case Narrative (Make an Appointment)**

Full Description of use case ***Make an Appointment***.

|  |  |  |
| --- | --- | --- |
| **Use-Case #** | **8** | |
| **Use Case Name:** | Make an appointment | |
| **Scenario:** | A patient wants to make an appointment. | |
| **Event (Trigger):** | New appointment. | |
| **Brief Description:** | Patient provides information on appointment details. System verifies appointment. On completion, the system displays appointment details. | |
| **Actors:** | Patient | |
| **Type:** | Essential | |
| **Preconditions:** | Patient is known by the system. | |
| **Post conditions:** | Appointment details added in the system. | |
| **Flow of Events**  **(Steps):** | **Actor Action** | **System Response** |
| 1. This use case begins when a patient wants to make an appointment to see his/her Doctor. |  |
| 1. The patient provides username and password. | * 1. System validates username and password and logs the patient into the system. |
| 1. Patient provides the appointment date and time. | * 1. The system validates the information and displays a confirmation screen indicating appointment booked. |
|  | 1. Patient confirms appointment date and time. | * 1. Patients’ appointment date and time is added to the system. |
| **Alternative Flow of Events** | | |
| **Line 2.1:** Invalid username and password entered. Indicate error. Return to Step 2.  **Line 3.1:** Date and Time provided is already taken. Inform the patient. Return to Step 3. | | |

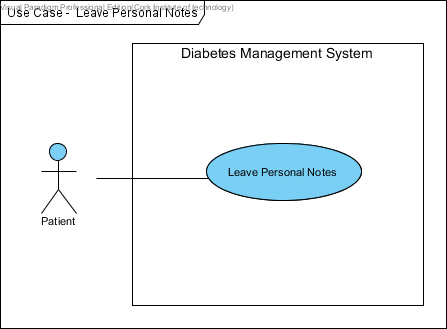


***Patient makes an Appointment***

**Use-Case Narrative (Leave Personal Notes)**

Full Description of use case ***Leave Personal Notes***

|  |  |  |
| --- | --- | --- |
| **Use-Case #** | **9** | |
| **Use Case Name:** | Leave Personal Notes | |
| **Scenario:** | A patient wants to leave personal note in the Diabetes Management System. | |
| **Event (Trigger):** | The use case is initiated when the patient wants to leave personal notes. | |
| **Brief Description:** | Patient provides information on what to leave in the system such as drafts, appointment dates, emergency contacts, etc. | |
| **Actors:** | Patient | |
| **Type:** | Essential | |
| **Preconditions:** | Patient is known by the system. | |
| **Post conditions:** | New personal notes is created and added in the system. | |
| **Flow of Events**  **(Steps):** | **Actor Action** | **System Response** |
| 1. This use case begins when a patient wants to leave personal notes. |  |
| 1. The patient provides username and password. | * 1. System validates username and password and logs the patient into the system. |
| 1. The patient provides the appointment dates, emergency contacts or useful telephone numbers, personal drafts, title of the note, date created, etc. | * 1. Adds personal notes to the system.   2. The system displays patients’ personal notes. |
| **Alternative Flow of Events** | | |
| **Line 2.1:** Invalid username and password entered. Indicate error. Return to Step 2. | | |

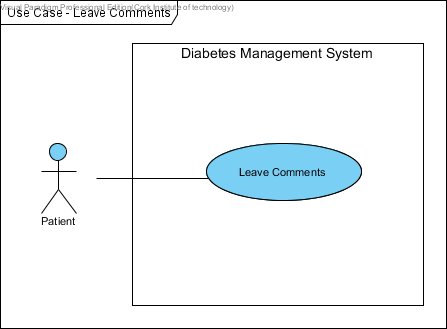


***Patient leaves Personal Notes***

**Use-Case Narrative (Leave Comments)**

Full Description of use case ***Leave Comments***

|  |  |  |
| --- | --- | --- |
| **Use-Case #** | **10** | |
| **Use Case Name:** | New Comments or Special Events | |
| **Scenario:** | A patient wants to leave comments in the Diabetes Management System. | |
| **Event (Trigger):** | This use case is initiated when the patient selects this option from the User Interface. | |
| **Brief Description:** | Patient wants to leave comments or special events in the system. The patient provides information on what to leave in the system such as: the events that happened after taking insulin injection. | |
| **Actors:** | Patient | |
| **Type:** | Essential | |
| **Preconditions:** | Patient is known by the system. | |
| **Post conditions:** | Comments are added in the system. | |
| **Flow of Events**  **(Steps):** | **Actor Action** | **System Response** |
| 1. This use case begins when a patient wants to leave comments in the system. |  |
| 1. The patient provides username and password. | * 1. System validates username and password and logs the patient into the system. |
| 1. The patient provides the special events. This may include an event that occurred after insulin injection. | * 1. Adds comments to the system.   2. The system displays patients’ comments or events. |
| **Alternative Flow of Events** | | |
| **Line 2.1:** Invalid username and password entered. Indicate error. Return to Step 2. | | |

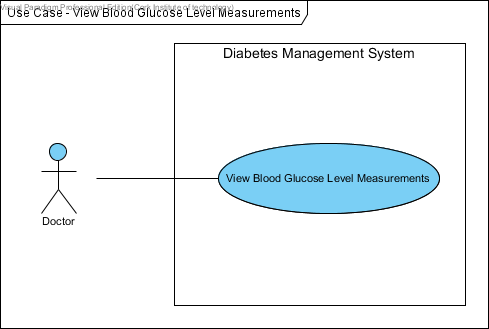


***Patient leaves Comments***

**Use-Case Narrative (View Blood Glucose Level Measurements)**

Full Description of use case ***View Blood Glucose Level Measurements***

|  |  |  |
| --- | --- | --- |
| **Use-Case #** | **11** | |
| **Use Case Name:** | View Patients’ Blood Glucose Level Measurements | |
| **Scenario:** | Doctor wants to view his/her patients’ blood glucose level results. | |
| **Event (Trigger):** | New patients’ blood glucose level results. | |
| **Brief Description:** | Doctor requests to view patients’ blood glucose level results. System displays patients’ blood glucose level results. | |
| **Actors:** | Doctor | |
| **Type:** | Essential | |
| **Preconditions:** | Doctor and Patient are known by the system.  Patients’ Blood Glucose Level results must exist and be up to date. | |
| **Post conditions:** | None | |
| **Flow of Events**  **(Steps):** | **Actor Action** | **System Response** |
| 1. This use case begins when a doctor wants to view his/her patients’ blood glucose level results. |  |
| 1. The doctor provides username and password. | * 1. System validates username and password and logs the doctor into the system. |
| 1. Doctor requests to view patients’ blood glucose level results. | * 1. System displays patients’ blood glucose level results table. |
| 1. Doctor views his/her patients’ blood glucose level results table and analyses the results to make adjustments. |  |
| **Alternative Flow of Events** | | |
| **Line 2.1:** Invalid username and password entered. Indicate error. Return to Step 2. | | |

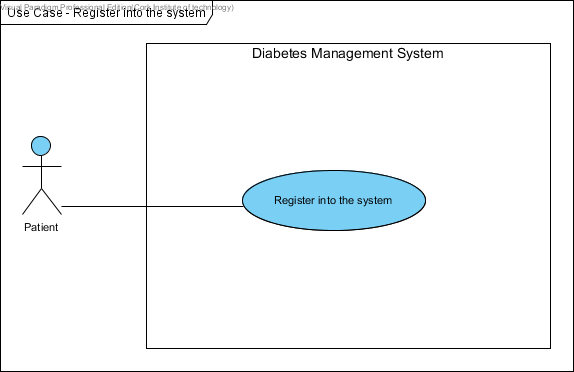


***Doctor views Blood Glucose Level Measurements***

**Use-Case Narrative (Register into the system)**

Full Description of use case ***Register into the system***

|  |  |  |
| --- | --- | --- |
| **Use-Case #** | **12** | |
| **Use Case Name:** | Register into the system | |
| **Scenario:** | A patient or a doctor wants to register into the system. | |
| **Event (Trigger):** | New Patient or Doctor. | |
| **Brief Description:** | The patient or doctor provides personal details. System adds patient or doctor to the system. | |
| **Actors:** | Patient and Doctor | |
| **Type:** | Essential | |
| **Preconditions:** | Patient and Doctor details must exit. | |
| **Post conditions:** | Both patient and doctor are registered into the system. | |
| **Flow of Events**  **(Steps):** | **Actor Action** | **System Response** |
| 1. This use case begins when a user wants to register for the system. |  |
| 1. The user provides his/her personal information such as first name, last name, username, password, email address. | * 1. System validates users’ personal details. |
| 1. The user indicates that the entry of information is complete. | * 1. The system generates and assigns a unique username and password to the user.   2. The user is added to the system.   3. Then, an automated email is sent to the user to activate their account. |
| **Alternative Flow of Events** | | |
| **Line 2.1:** Invalid Information provided. Inform the user. Return to Step 2. | | |

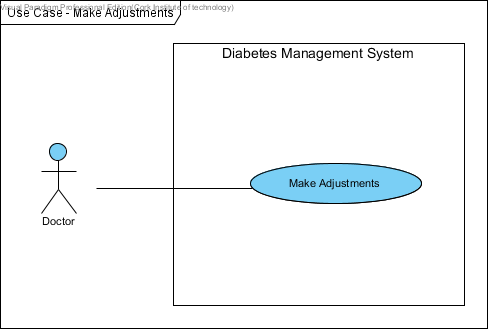


***Patient registers into the system***

**Use-Case Narrative (Make Adjustments)**

Full Description of use case ***Make Adjustments***

|  |  |  |
| --- | --- | --- |
| **Use-Case #** | **13** | |
| **Use Case Name:** | Make Adjustments | |
| **Scenario:** | Doctor wants to make adjustments to patients Insulin Doses. | |
| **Event (Trigger):** | New Adjustment to patients Insulin Doses. | |
| **Brief Description:** | Doctor requests to make adjustment to his/her patients’ Insulin Doses. Doctor provides information on adjustment modification. On completion, the system adds new adjustment and sends automated email to the patient. | |
| **Actors:** | Doctor | |
| **Type:** | Essential | |
| **Preconditions:** | Doctor is known by the system.  Patient must exist.  Patients Insulin Doses must exist and be valid. | |
| **Post conditions:** | Patients Insulin Doses updated. | |
| **Flow of Events**  **(Steps):** | **Actor Action** | **System Response** |
| 1. This use case begins when a doctor wants to make adjustments to the patients Insulin Doses. |  |
| 1. The doctor provides username and password. | * 1. System validates username and password and logs the doctor into the system. |
| 1. Doctor requests to modify patients Insulin Doses. | * 1. System displays patients’ Insulin Doses. |
| 1. Doctor makes necessary adjustments to the patients Insulin Doses. | 4.1. System validates and updates the information.  4.2. System displays the new adjustments to the doctor.  4.3. System sends an automated email to patients. |
| **Alternative Flow of Events** | | |
| **Line 2.1:** Invalid username and password entered. Indicate error. Return to Step 2.  **Line 3.1:** Invalid Insulin Adjustments. Indicate error. Return to Step 4. | | |

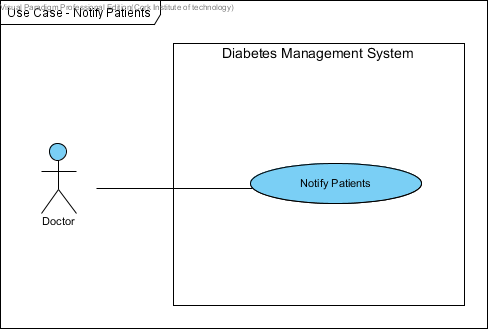


***Doctor makes Adjustments***

**Use-Case Narrative (Notify Patients)**

Full Description of use case ***Notify Patients***.

|  |  |  |
| --- | --- | --- |
| **Use-Case #** | **14** | |
| **Use Case Name:** | Notify Patients | |
| **Scenario:** | Doctor wants to notify his/her patients to come in for check-ups. | |
| **Event (Trigger):** | New Check-up. | |
| **Brief Description:** | Doctor provides the necessary information for the notification. System sends an automated email to the patient. | |
| **Actors:** | Doctor | |
| **Type:** | Essential | |
| **Preconditions:** | Doctor is known by the system.  Patient must exist.  Patients’ blood glucose level readings must exist and be up-to-date. | |
| **Post conditions:** | Patients Insulin Doses updated. | |
| **Flow of Events**  **(Steps):** | **Actor Action** | **System Response** |
| 1. This use case begins when a doctor wants to notify patients to come in for check-ups. |  |
| 1. The doctor provides username and password. | * 1. System validates username and password and logs the doctor into the system. |
| 1. Doctor requests to view patients’ blood glucose level readings. | * 1. System displays patients’ blood glucose level readings. |
| 1. Doctor reviews the results and requests to send an automated email to patients. | 4.1. System validates and sends an automated email to patients. |
| **Alternative Flow of Events** | | |
| **Line 2.1:** Invalid username and password entered. Indicate error. Return to Step 2. | | |

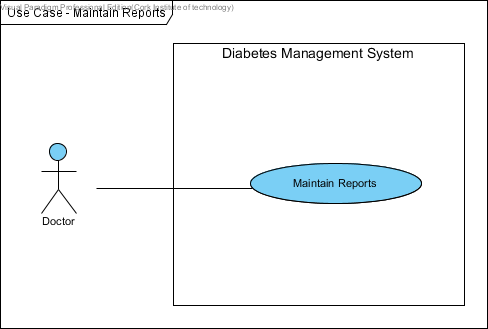


***Doctor notifies Patients***

**Use-Case Narrative (Maintain Reports)**

Full Description of use case ***Maintain Reports***.

|  |  |  |
| --- | --- | --- |
| **Use-Case #** | **15** | |
| **Use Case Name:** | Maintain Reports | |
| **Scenario:** | Doctor creates patients report. | |
| **Event (Trigger):** | New Reports | |
| **Brief Description:** | This use case allows the Doctor to maintain any information such as patients’ record of blood sugar readings, record of insulin adjustments, record of blood pressure, list of insulin doses, etc. Also, the Doctor can delete the report from the system. | |
| **Actors:** | Doctor | |
| **Type:** | Essential | |
| **Preconditions:** | The Doctor must be logged into the system in order for this use case to begin. | |
| **Post conditions:** | None | |
| **Flow of Events**  **(Steps):** | **Actor Action** | **System Response** |
| 1. This use case is initiated when a Doctor wants to maintain reports. |  |
| 1. The Doctor provides his/her username and password. | * 1. System validates username and password and logs the Doctor into the system. |
| 1. The Doctor specifies the report criteria: report type (either patients’ information, patients’ record of blood sugar readings, record of insulin adjustments), dates for the report, patients name(s). | * 1. The system displays the report criteria for the Doctor. |
| 1. Doctor provides the information such as report title. | * 1. The system provides a report satisfying the report criteria. |
| 1. Doctor indicates everything is complete. | * 1. The system saves the report to the specified name and location. |
|  | 1. Doctor confirms the decision to print the report. | * 1. Prints updated report. |
| **Alternative Flow of Events** | | |
| **Line2.1:** Invalid username and password entered by the Doctor. Indicate error. Return to Step 2.  **Line 3.1:** Invalid format or insufficient information. Inform Doctor. Return to Step 3.  **Line4.1:** Requested information unavailable. Inform Doctor. Return to Step 4. | | |



***Doctor maintains Reports***

* 1. Summary of Prototyping
     1. **Functional Requirements** 
        1. The system shall enable the patient and doctor to log in with their credentials username and password.
        2. The system shall enable the patient to view their personal information.
        3. The system shall enable the patient to update their personal information.
        4. The system shall enable the patient to view their blood glucose level records.
        5. The system shall enable the patient to record insulin doses.
        6. The system shall enable the patient to record their blood glucose level.
        7. The system shall enable the patient to record their blood pressure.
        8. The system shall allow the patient to make an appointment.
        9. The system shall allow the patient to leave personal notes.
        10. The system shall allow the patient to leave comments.
        11. The system shall enable the doctor to view their patients’ blood glucose level measurements.
        12. The system shall enable the patient and doctor to register into the system.
        13. The system shall enable the doctor to make adjustments to patients’ insulin doses.
        14. The system shall allow the doctor to notify patients.
        15. The system shall enable the doctor to maintain reports.
     2. **Non-Functional Requirements** 
        1. **Operational Requirements**
           1. The system shall run on different web browsers (such as Google Chrome, Mozilla Firefox, and Internet Explorer) and mobile devices that are used by stakeholders.
           2. The system shall interface with the business management system.
           3. The system shall be flexible and easily adaptable to changes saving up to 50 percent of its operating cost.
           4. The system shall be a platform and accessible from any devices and from any remote areas.
        2. **Performance Requirements**
           1. The system shall support management staff of approximately 150 management people.
           2. The system shall enable patients to update their personal information and blood glucose level records within 2 minutes.
           3. The user shall get a system response within 10 seconds.
           4. The system shall be able to register up to 100 diabetic patients.
           5. Responses to report queries shall take less than two seconds.
        3. **Security Requirements**
           1. No patient can access any other patients’ personal information or blood glucose records.
           2. The system shall be secured to prevent unauthorized access from patients’ data.
        4. **Availability Requirements**
           1. Information about blood glucose records shall be available at any time to the patient.
           2. A list of diabetic patients’ shall be made available at any time to their doctors.
  2. **DATA DICTIONARY**

This data dictionary is a textual list of all concepts that are defined during the analysis. The aim of the data dictionary is to define a vocabulary that will be common to all the users of the Diabetes Management System.

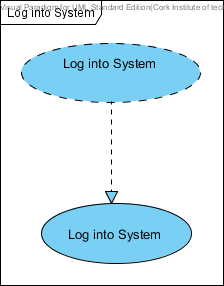
|  |  |
| --- | --- |
| **Name** | **Description** |
| **Patient ID** | This is a unique automated Identification Number, which is assigned to each patient when registered by the system. This will enable them to log in into the system. |
| **Username** | This is an Identification Name of the user account name. This can be combination of the user names and any other names or special characters the account will know. |
| **Password** | A string of characters that allows access to the system i.e. it is used by the user to gain access to the system. The password enforces the privacy of the users account. |
| **First Name** | The first name of the user as known to the system. |
| **Last Name** | The users’ last name as known to the system. |
| **Address** | This is the current postal address of the users. These are home or office address of the user. |
| **Phone Number** | This is the contact phone number of the user. |
| **Email Address** | This is an active and current email address of the user. The system uses these email addresses to send email to the user on any occasions including registration. |
| **DOB** | This is the date of birth of the patient. |
| **Gender** | It is the sex of the user of the application. |
| **Diabetes** | This is a disorder of the metabolism causing excessive thirst and the production of large amounts of urine. |
| **Height** | This is the height of the patient. |
| **Kgs** | This measure the unit of the weight. |
| **Medical History** | This is the medical history of the patient such as: Allergies, heart condition and general patient’s health status. |
| **Blood Pressure** | This is the pressure of the blood in the circulatory system. Also, it is measured for diagnosis since it is closely related to the force and rate of the heartbeat and the diameter and elasticity of the arterial walls. |
| **Blood Glucose Level** | This is the amount of glucose in the blood. Glucose is a sugar that comes from the foods we eat and it’s also formed and stored inside the body. |
| **Insulin Doses** | This is the number of Units given in each injection (Pre-Breakfast, Pre-Lunch, Pre-Dinner, and Before Bed). |
| **Comments** | These are special events that happen when the patient takes the insulin injections. The patient can leave comments such as: “hype during the night”. |
| **Personal Notes** | This is some text written by the patient that can contain information such as appointment dates, checklist for diabetes day care unit. It is updated frequently. |
| **Profile Image** | This is a picture of the user. |
| **Aerobic Exercise** | Activity involving large muscles, done for an extended period of time. Aerobic exercise can be used to improve insulin sensitivity and assist in diminishing elevated blood glucose levels into the normal range. Examples of aerobic exercise include walking, biking, jogging, swimming, aerobic classes and cross country skiing. |
| **Database Security** | Connecting directly to a database leaves the system vulnerable to hackers. Referencing ‘Database Security’ means connecting to the database by first sending information through a script and then to the database. |
| **Database** | This is where data is stored such as patients’ data. |
| **Doctor** | This is a user that monitors patients’ activities. He/she is qualified to treat people who are ill. Doctor can notify patient when their blood sugar level readings are low. |
| **Nurse** | This is someone trained to take care of the sick or infirm especially in a hospital. A nurse is also a user capable of using the system. |
| **Patient** | A user of the system that can view and update their information. |
| **User ID** | Special number to identify different users in the system. |
| **Log in** | An interface where the user must input their username and password before they can use the application. It is the bridge between the system and the user. |
| **Performance Graph** | A graph representation to illustrate patients’ blood sugar level performance. |
| **Register** | To be able to use the system, the user must register. This process allows patients and doctors to access their information that is stored in the database. |
| **Diabetic Patient** | A person that has diabetes. |

* 1. **COLLABORATION DIAGRAM**

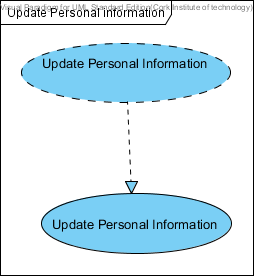
Collaboration Diagrams are used to model an alternate view of the Sequence Diagram. It shows how objects involved in a situation interact with each other, instantiating a particular class in the Diabetes Management System. Also, it gives the modeller the ability to show detail, such as visibility. Collaboration diagrams are used to understand all of the effects on any given object and are also used for procedural design.

Collaboration diagrams are used to model every scenario found in the Diabetes Management System.

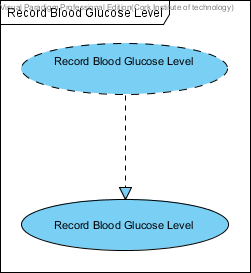
The use case diagram **Log into System** is collaborating with the Use case Log into System.



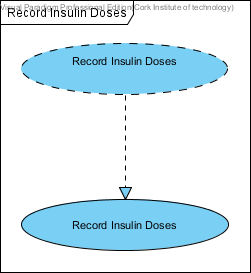
The use case diagram **Update Personal Information** is collaborating with the Use case Update Personal Information.



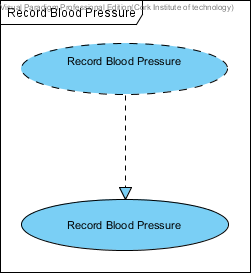
The use case diagram **Record Blood Glucose Level** is collaborating with the Use case Record Blood Glucose Level.



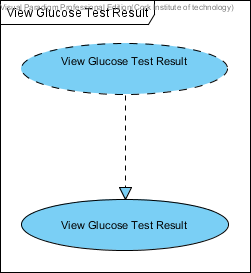
The use case diagram **Record Insulin Doses** is collaborating with the Use case Record Insulin Doses.



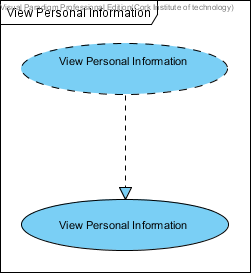
The use case diagram **Record Blood Pressure** is collaborating with the Use case Record Blood Pressure.



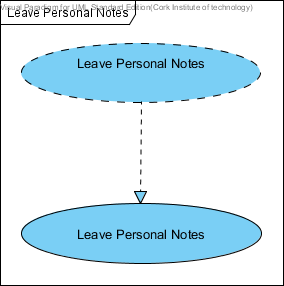
The use case diagram **View Blood Glucose Level Result** is collaborating with the Use case View Blood Glucose Level Result.



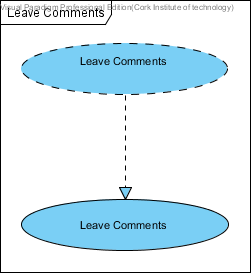
The use case diagram **View Personal Information** is collaborating with the Use case View Personal Information.



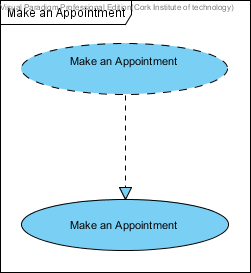
The use case diagram **Leave Personal Notes** is collaborating with the Use case Leave Personal Notes.



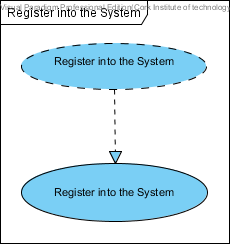
The use case diagram **Leave Comments** is collaborating with the Use case Leave Comments.



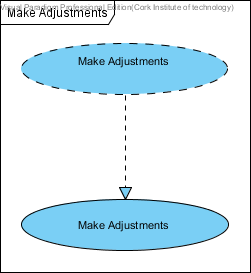
The use case diagram **Make an Appointment** is collaborating with the Use case Make an Appointment.



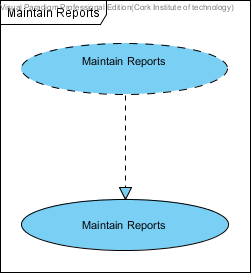
The use case diagram **Register into the system** is collaborating with the Use case Register into the system.



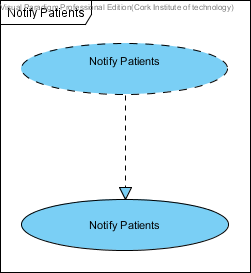
The use case diagram **Make Adjustments** is collaborating with the Use case Make Adjustments.



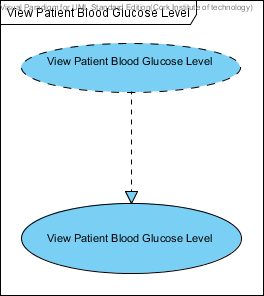
The use case diagram **Maintain Reports** is collaborating with the Use case Maintain Reports.



The use case diagram **Notify Patients** is collaborating with the Use case Notify Patients.



The use case diagram **View Patient Blood Glucose Level** is collaborating with the Use case View Patient Blood Glucose Level.

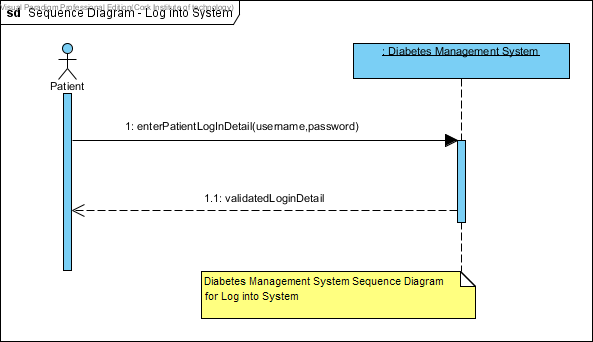


* 1. **SEQUENCE DIAGRAMS**

Sequence diagrams display the interaction between objects arranged in a time sequence. An interaction sequence diagram (also known as a sequence diagram) is one of many kinds of UML interaction diagram. The sequence diagram is semantically equivalent to a communication diagram for simple interactions.

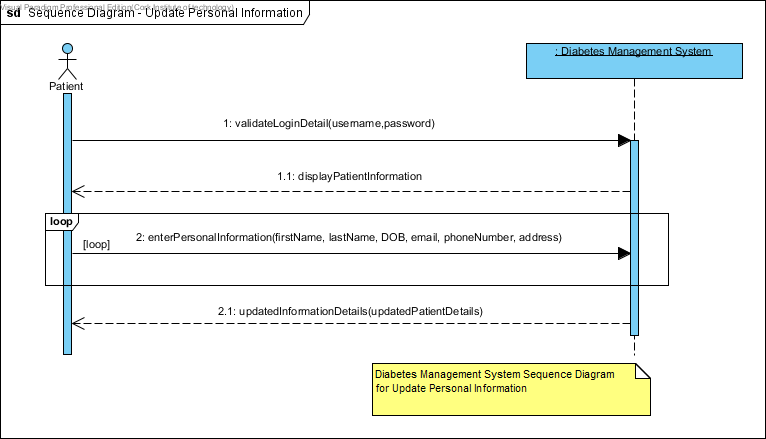
Sequence diagrams can be modelled at different levels of information and to meet different purposes at several stages in the development lifecycle. The most common use of a sequence diagram is to show the specific object interaction that occurs mainly for one use case or one application. For instance, when an interaction sequence diagram is used to design the dynamic behaviour of a use case, it can be seen as a detailed specification of the use case.

The sequence diagram below is used to represent the detailed object interaction that occurs for the use case **Log into System**. The diagram also shows how a **Patient, Doctor and a Nurse** mayinteract with the **Diabetes Management System**.



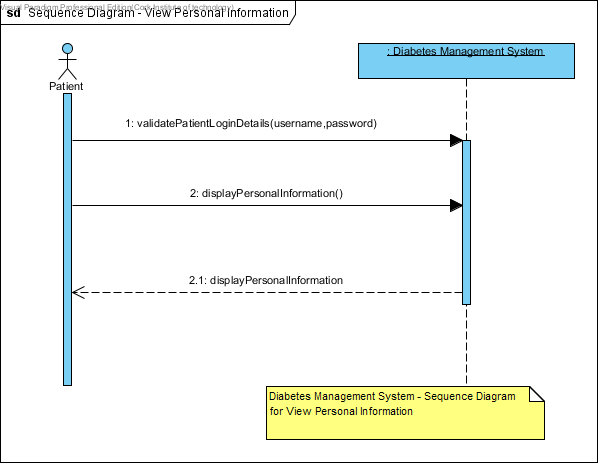
Sequence Diagram for the use case **Log into System**

The sequence diagram below is used to represent the detailed object interaction that occurs for the use case **Update Personal Information**. The sequence diagram below shows how a **patient and a Doctor** interact with the **Diabetes Management System**.



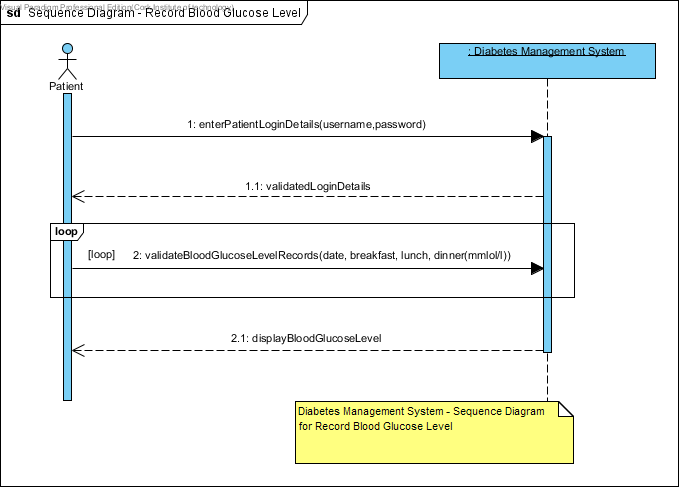
Sequence Diagram for the use case **Update Personal Information**.

The sequence diagram below is used to represent the detailed object interaction that occurs for the use case **View Personal Information**. The sequence diagram below shows how a **patient** interacts with the **Diabetes Management System**.



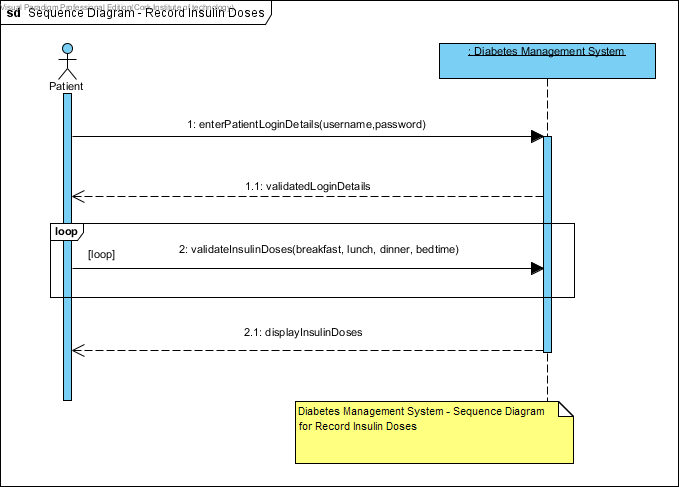
Sequence Diagram for the use case **View Personal Information.**

The sequence diagram below is used to represent the detailed object interaction that occurs for the use case **Record Blood Glucose Level**. The sequence diagram below shows how a **patient** interacts with the **Diabetes Management System**.



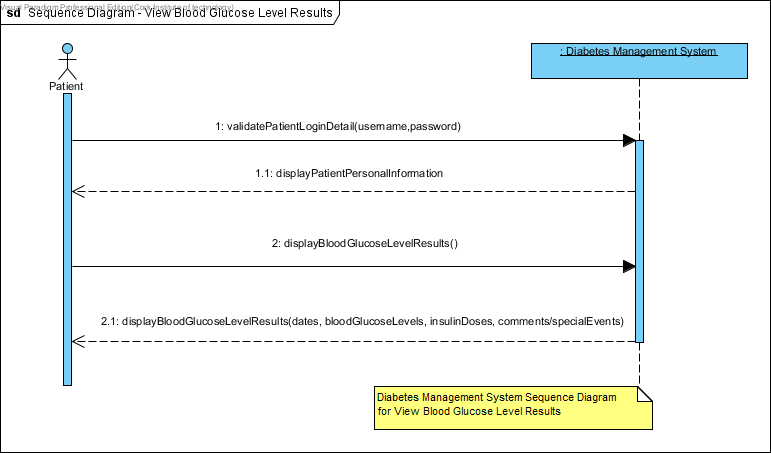
Sequence Diagram for the use case **Record Blood Glucose Level.**

The sequence diagram below is used to represent the detailed object interaction that occurs for the use case **Record Insulin Doses**. The sequence diagram below outlines how a **patient** interacts with the **Diabetes Management System**.



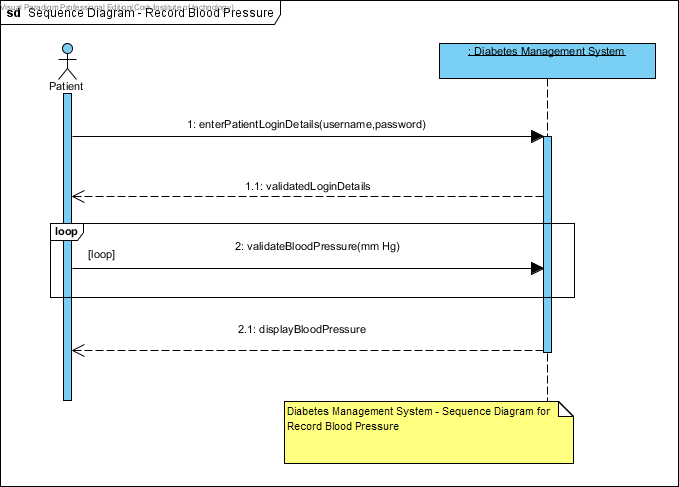
Sequence Diagram for the use case **Record Insulin Doses.**

The sequence diagram below is used to represent the detailed object interaction that occurs for the use case **View Blood Glucose Level Results**. The sequence diagram below outlines how a **patient** interacts with the **Diabetes Management System**.



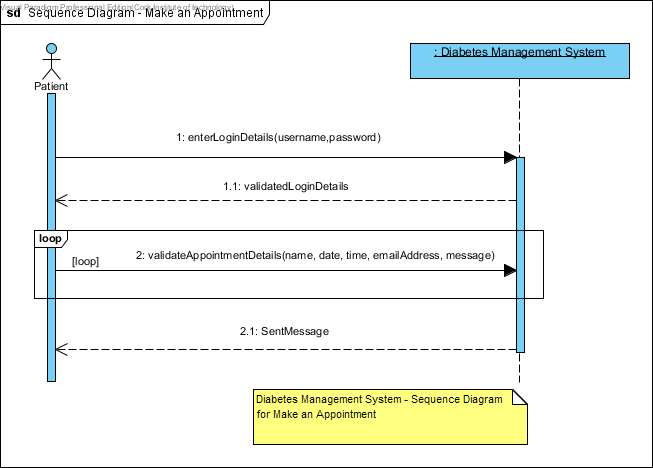
Sequence Diagram for the use case **View Blood Glucose Level Results.**

The sequence diagram below is used to represent the detailed object interaction that occurs for the use case **Record Blood Pressure**. The diagram also shows how a **patient** interacts with the **Diabetes Management System**.



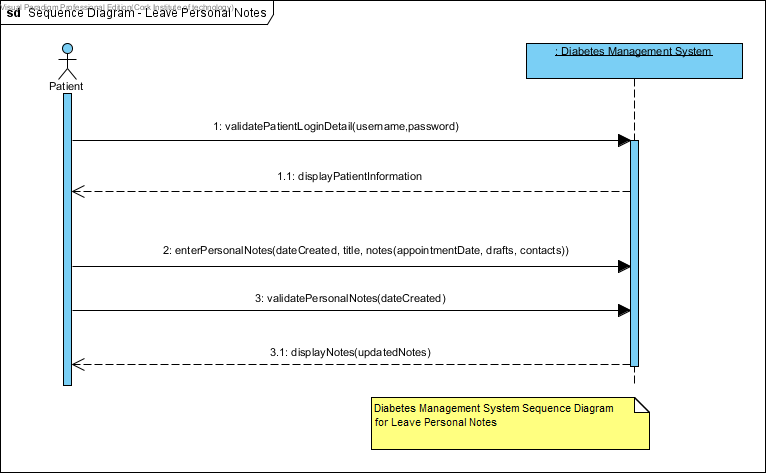
Sequence Diagram for the use case **Record Blood Pressure.**

The sequence diagram below is used to represent the detailed object interaction that occurs for the use case **Make an Appointment**. The diagram below also shows how a **patient** interacts with the **Diabetes Management System**.



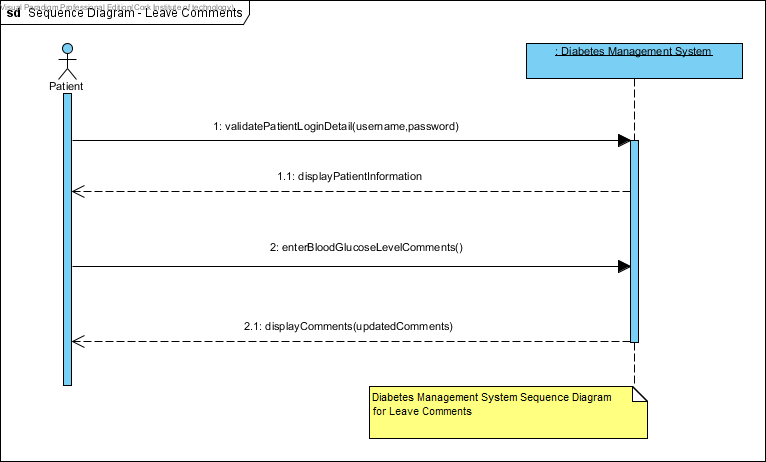
Sequence Diagram for the use case **Make an Appointment.**

The sequence diagram below is used to represent the detailed object interaction that occurs for the use case **Leave Personal Notes**. The diagram outlines how a **patient** interacts with the **Diabetes Management System**.



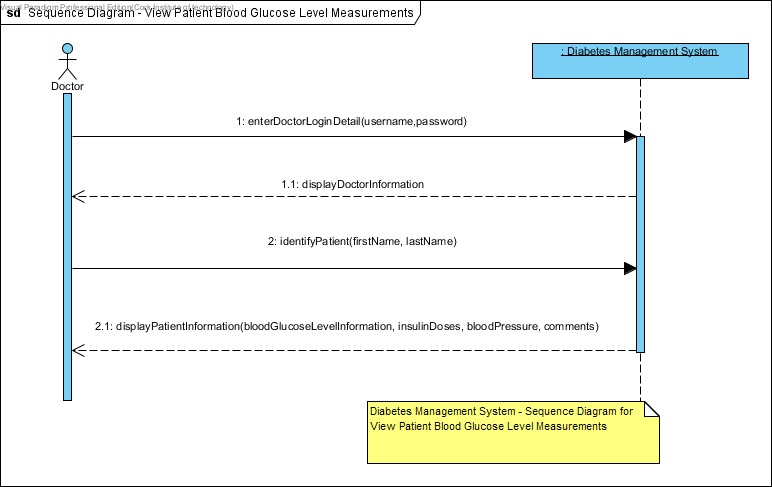
Sequence Diagram for the use case **Leave Personal Notes.**

The sequence diagram below is used to represent the detailed object interaction that occurs for the use case **Leave Comments**. Here, it outlines how a **patient** interacts with the **Diabetes Management System**.



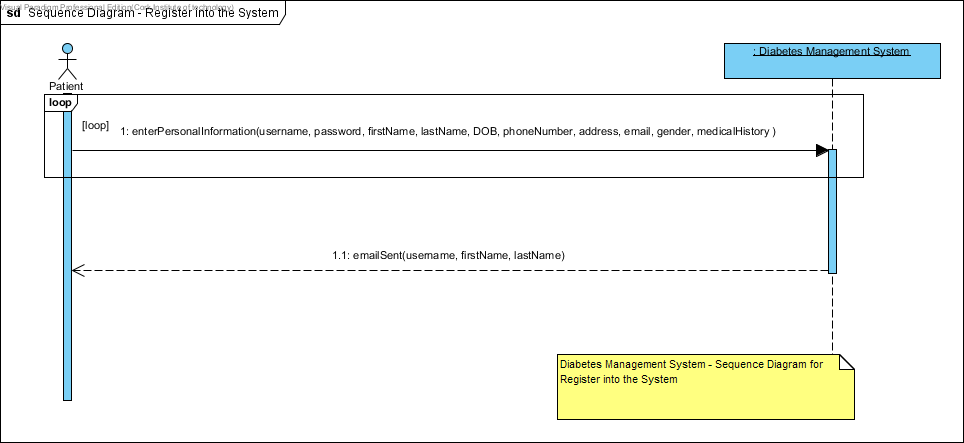
Sequence Diagram for the use case **Leave Comments.**

The sequence diagram below is used to represent the detailed object interaction that occurs for the use case **View Patient Blood Glucose Level Measurements**. The diagram also shows how a **Doctor** interacts with the **Diabetes Management System**.



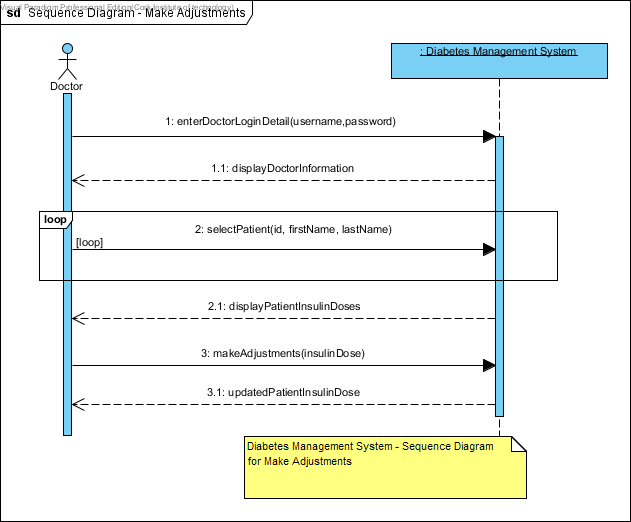
Sequence Diagram for the use case **View Patient Blood Glucose Level Measurements**.

The sequence diagram below is used to represent the detailed object interaction that occurs for the use case **Register into the System**. The diagram also shows how a **patient** and a **doctor** interact with the **Diabetes Management System**.



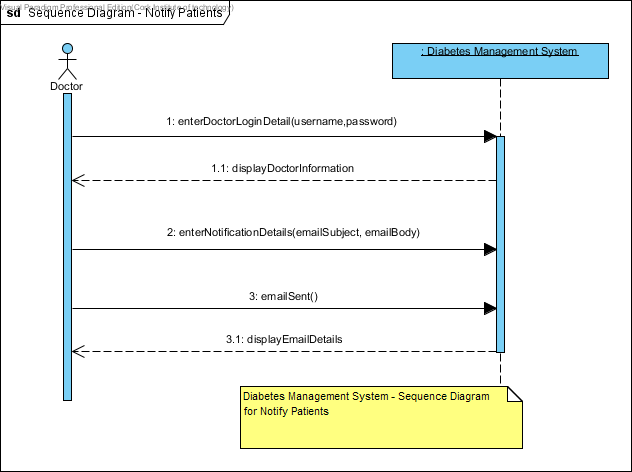
Sequence Diagram for the use case **Register into the System**.

The sequence diagram below is used to represent the detailed object interaction that occurs for the use case **Make Adjustments**. The diagram also shows how a **Doctor** interacts with the **Diabetes Management System**.



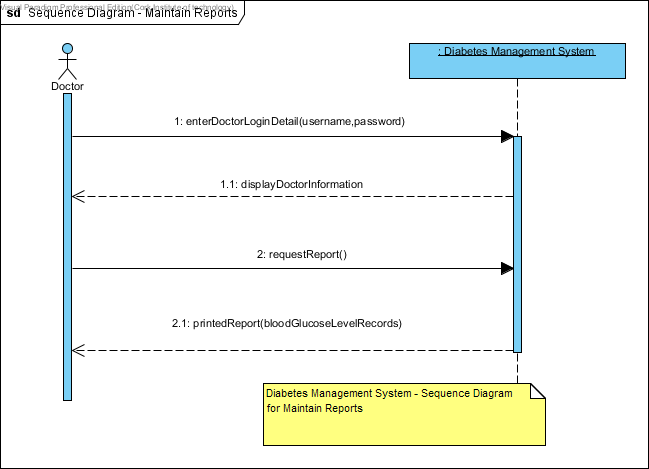
Sequence Diagram for the use case **Make Adjustments**.

The sequence diagram below is used to represent the detailed object interaction that occurs for the use case **Notify Patients**. The diagram also shows how a **Doctor** interacts with the **Diabetes Management System**.



Sequence Diagram for the use case **Notify Patients**.

The sequence diagram below is used to represent the detailed object interaction that occurs for the use case **Maintain Reports**. The diagram also shows how a **Doctor** interacts with the **Diabetes Management System**.



Sequence Diagram for the use case **Maintain Reports**.

* 1. **BUSINESS DATA MODEL**

**Class Diagrams**

Class Diagrams in UML (Unified Modelling Language) is a static diagram that describes the structure of the Diabetes Management System being modelled. It shows a collection of the System’s Classes (i.e. Class Names), their Attributes, Operations, and the Relationships among Objects. Moreover, it is the main building block of object-oriented modelling. Class diagrams can be used to illustrate data model of the Diabetes Management System.

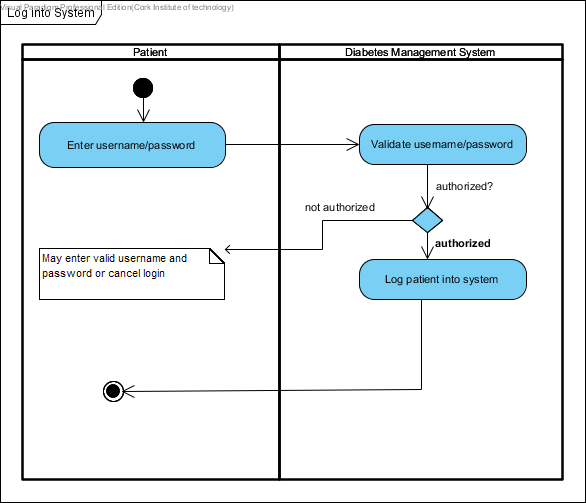
Class Diagrams will be used to model the general static view of the Diabetes Management System and detailed modelling that translate the models into programming codes.

***Domain/Conceptual Model for Diabetes Management System***

* 1. **ACTIVITY DIAGRAM**

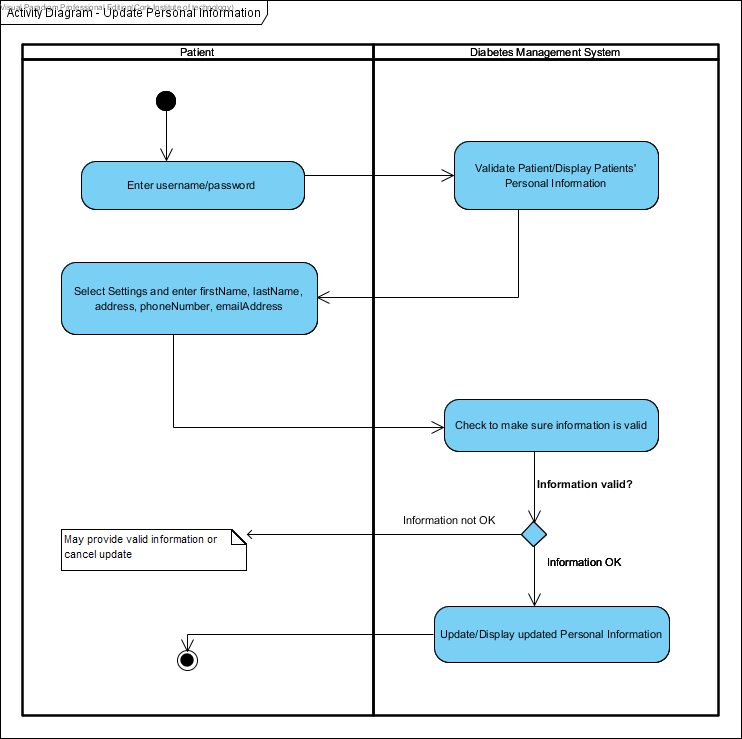
Activity diagrams are used to model different aspects of the Diabetes Management System. It is used to model a system function represented by a use case, possibly using object flows to show which objects that are involved in each use case. This is done during the phase of the lifecycle when requirements are being elaborated. Activity Diagrams are used at a low level to model the detail of how a particular operation is carried out, and are likely to be for this purpose in later analysis or system design activities. Moreover, activity diagrams at their simplest are used to show action flow from one activity to another, which can be seen in the Diabetes Management System.

The activity diagram below is used to illustrate the system function that is represented by the use case **Log into System.** Here, it describes the activity flow from the Patient and Doctor to the Diabetes Management System.



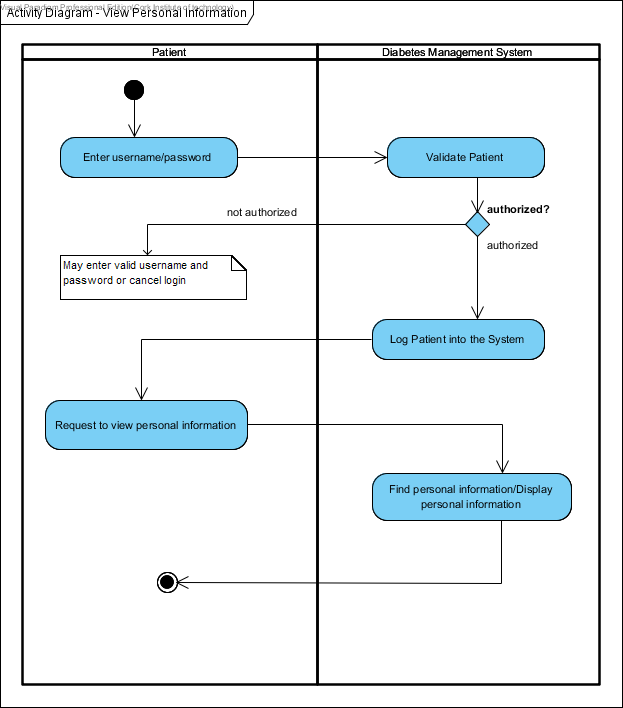
*Activity Diagram for the use case* ***Log into System***

The activity diagram below is used to illustrate the system function that is represented by the use case **Update Personal Information.** Here, it describes the activity flow from the Patient and Doctor to the Diabetes Management System.



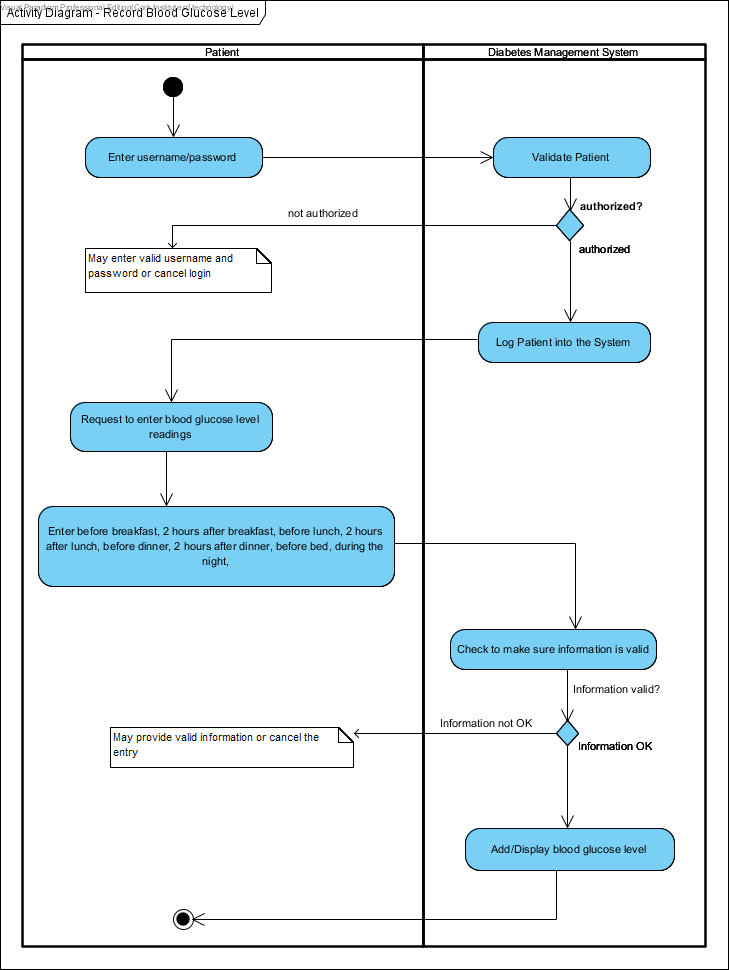
*Activity Diagram for the use case* ***Update Personal Information***

The activity diagram below is used to illustrate the system function that is represented by the use case **View Personal Information.** Here, it describes the activity flow from the Patient to the Diabetes Management System.



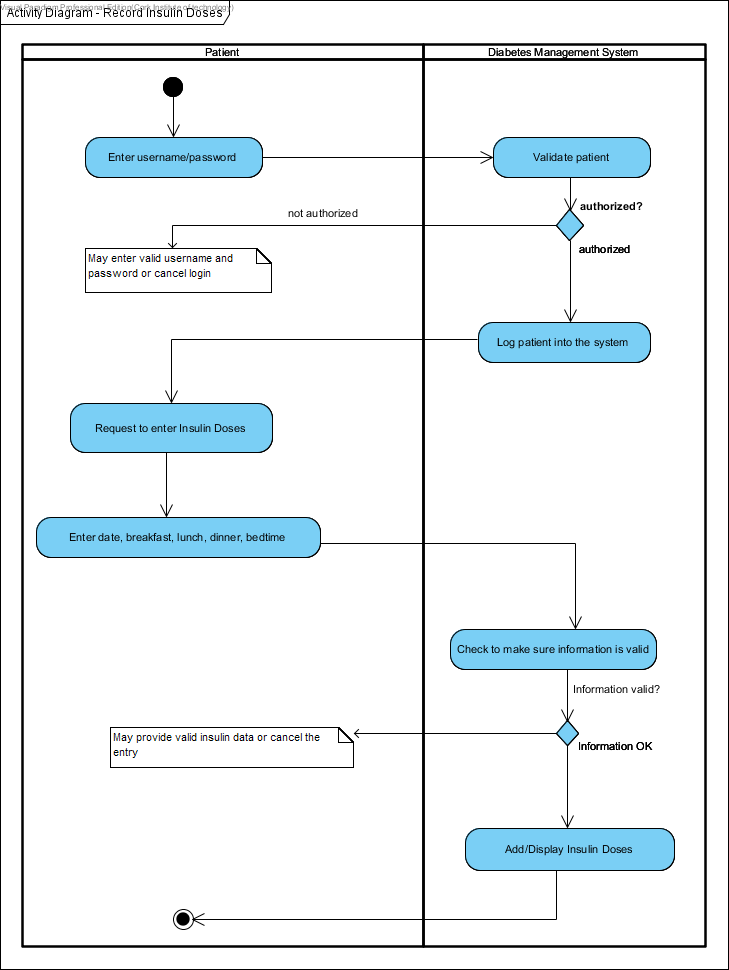
*Activity Diagram for the use case* ***View Personal Information***

The activity diagram below is used to illustrate the system function that is represented by the use case **Record Blood Glucose Level.** Here, it describes the activity flow from the Patient to the Diabetes Management System.



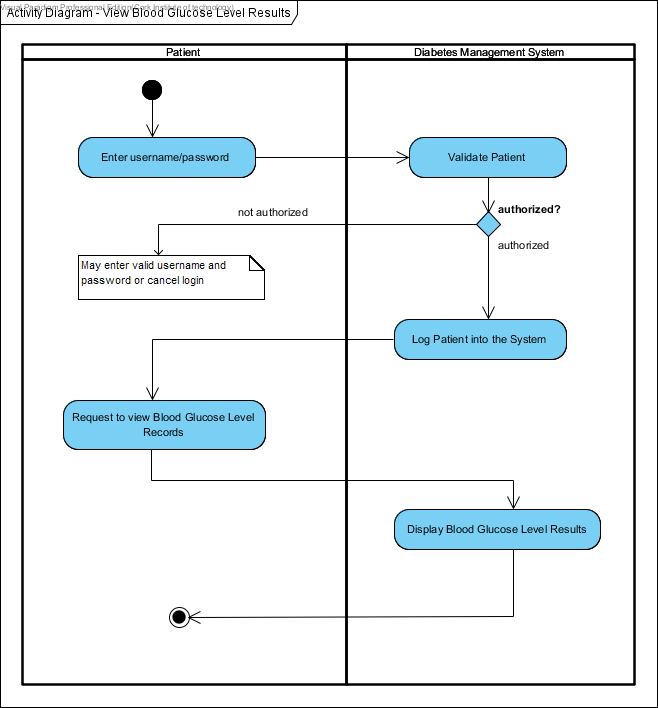
*Activity Diagram for the use case* ***Record Blood Glucose Level***

The activity diagram below is used to illustrate the system function that is represented by the use case **Record Insulin Doses.** Here, it describes the activity flow from the Patient to the Diabetes Management System.



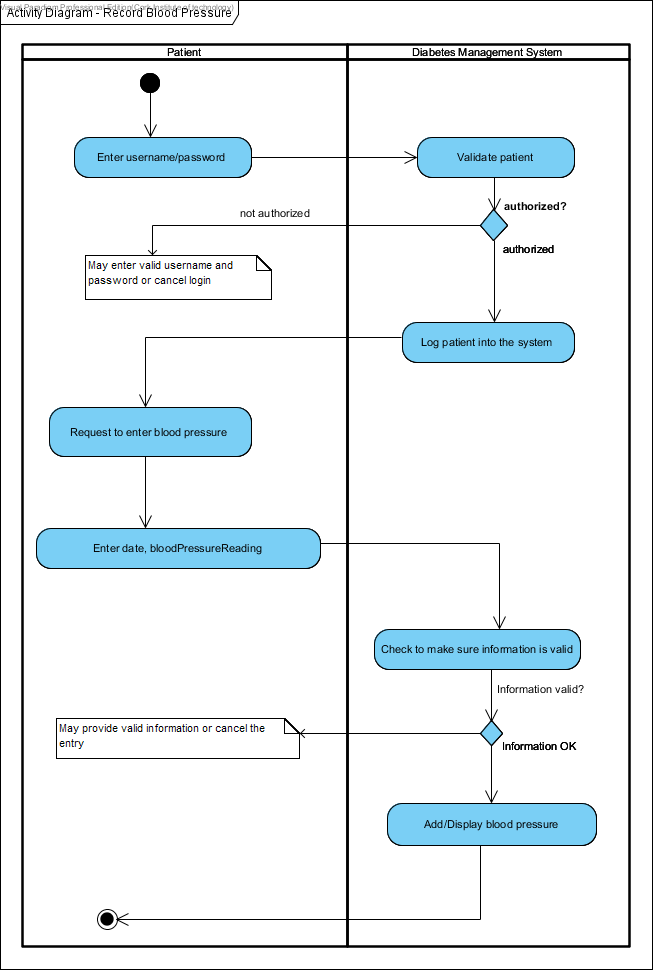
*Activity Diagram for the use case* ***Record Insulin Doses***

The activity diagram below is used to illustrate the system function that is represented by the use case **View Blood Glucose Level Results.** Here, it describes the activity flow from the Patient to the Diabetes Management System.



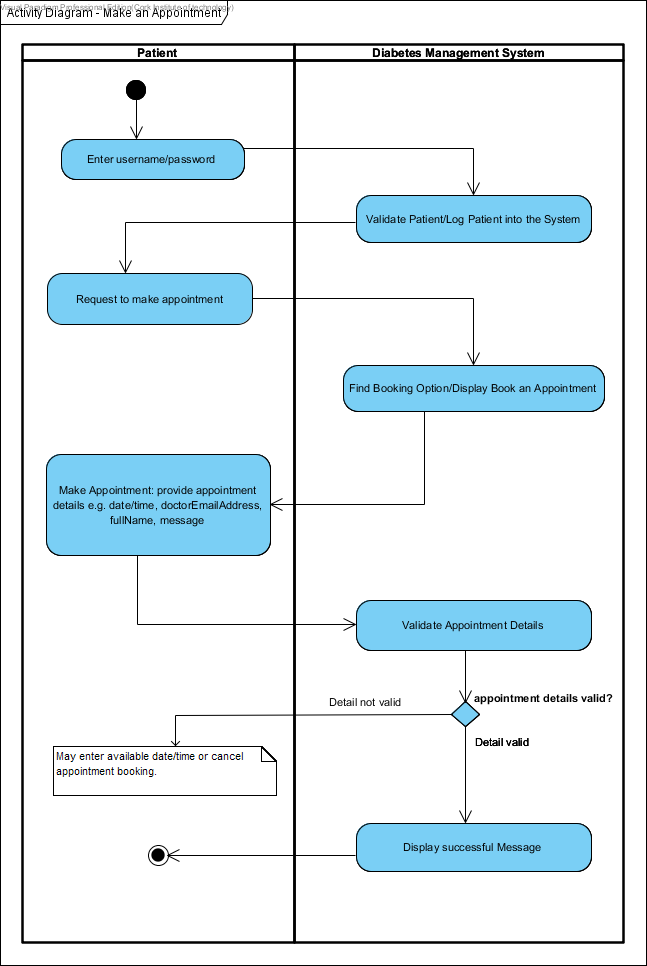
*Activity Diagram for the use case* ***View Blood Glucose Level Results***

The activity diagram below is used to illustrate the system function that is represented by the use case **Record Blood Pressure.** Here, it describes the activity flow from the Patient to the Diabetes Management System.



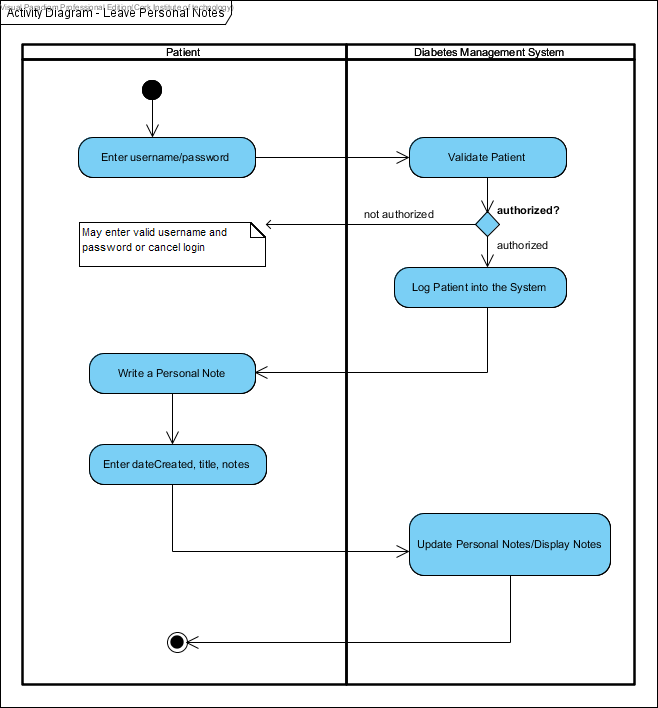
*Activity Diagram for the use case* ***Record Blood Pressure***

The activity diagram below is used to illustrate the system function that is represented by the use case **Make an Appointment.** Here, it describes the activity flow from the Patient to the Diabetes Management System.



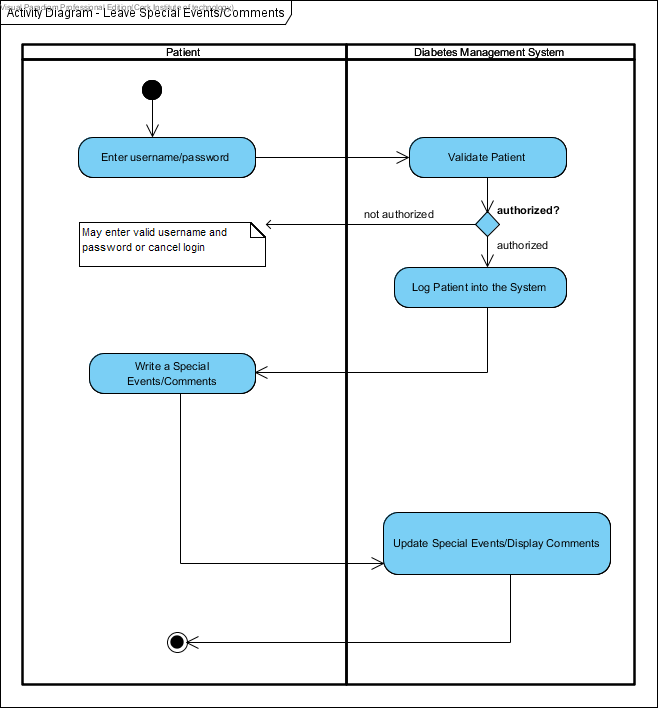
*Activity Diagram for the use case* ***Make an Appointment***

The activity diagram below is used to illustrate the system function that is represented by the use case **Leave Personal Notes.** Here, it describes the activity flow from the Patient to the Diabetes Management System.



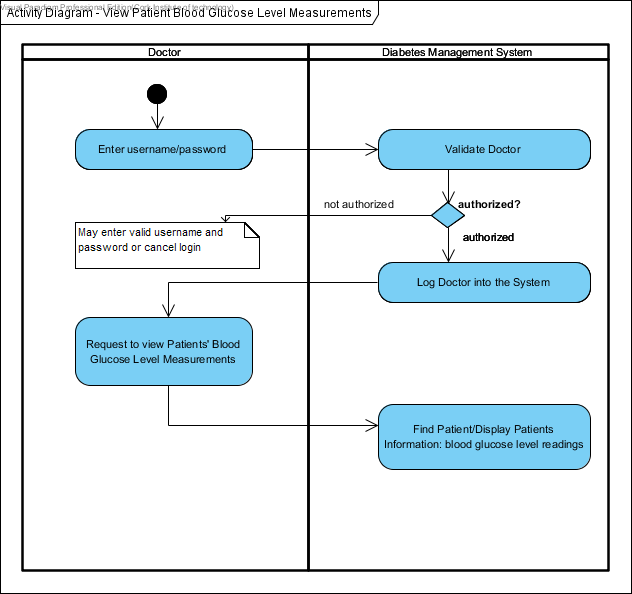
*Activity Diagram for the use case* ***Leave Personal Notes***

The activity diagram below is used to illustrate the system function that is represented by the use case **Leave Special Events/Comments.** Here, it describes the activity flow from the Patient to the Diabetes Management System.



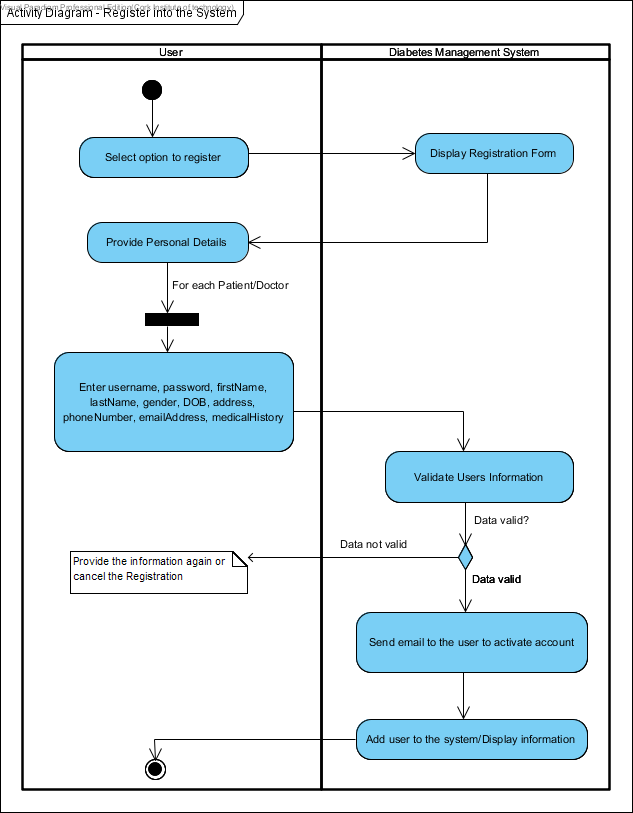
*Activity Diagram for the use case* ***Leave Special Events/Comments***

The activity diagram below is used to illustrate the system function that is represented by the use case **View Patient Blood Glucose Level Measurements.** Here, it describes the activity flow from the Doctor to the Diabetes Management System.



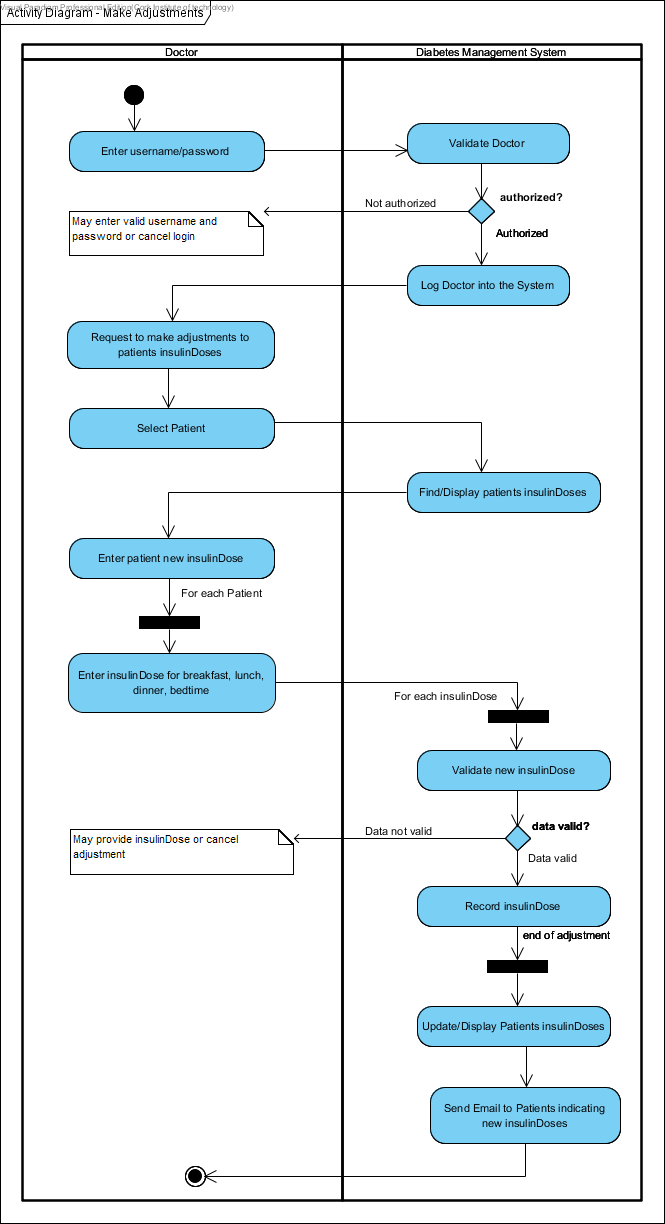
*Activity Diagram for the use case* ***View Patient Blood Glucose Level Measurements***

The activity diagram below is used to illustrate the system function that is represented by the use case **Register into the System.** Here, it describes the activity flow from the Patient and the Doctor to the Diabetes Management System.



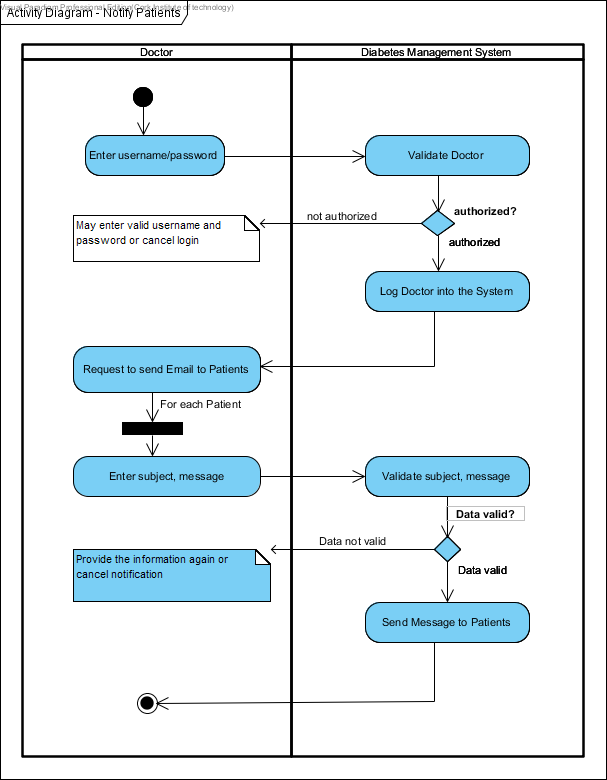
*Activity Diagram for the use case* ***Register into the System***

The activity diagram below is used to illustrate the system function that is represented by the use case **Make Adjustments.** Here, it describes the activity flow from the Doctor to the Diabetes Management System.



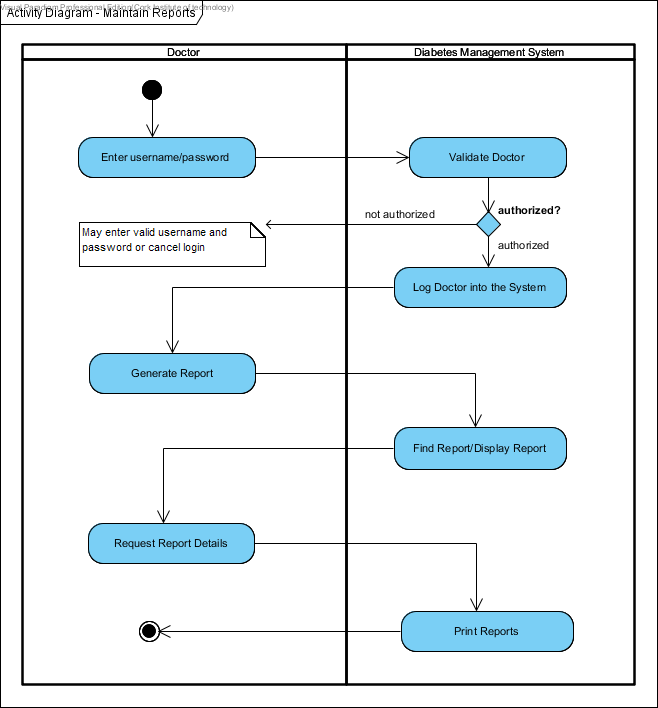
*Activity Diagram for the use case* ***Make Adjustments***

The activity diagram below is used to illustrate the system function that is represented by the use case **Notify Patients.** Here, it describes the activity flow from the Doctor to the Diabetes Management System.



*Activity Diagram for the use case* ***Notify Patients***

The activity diagram below is used to illustrate the system function that is represented by the use case **Maintain Reports.** Here, it describes the activity flow from the Doctor to the Diabetes Management System.



*Activity Diagram for the use case* ***Maintain Reports***

* 1. **DATABASE MODEL**

I used MySQL to implement the database, running on a Linux server remotely accessible from anywhere. The full SQL table definitions can be found in the soft-copy hand up under the **core** folder. Most SQL queries used by the program are integrated into its source code in the core package.

The diagram below describes the basic database model, arranged by key relations.

PK indicates a primary key.

FK indicates a foreign key

U indicates a unique key.

varChar indicates variable Character.

1. **IMPLEMENTATION**
   1. **USER MANUAL**

**Purpose of the User Manual**

The user manual provides instructional support and guidance to authorised personnel who use the Diabetes Management System to track and monitor blood glucose level records.

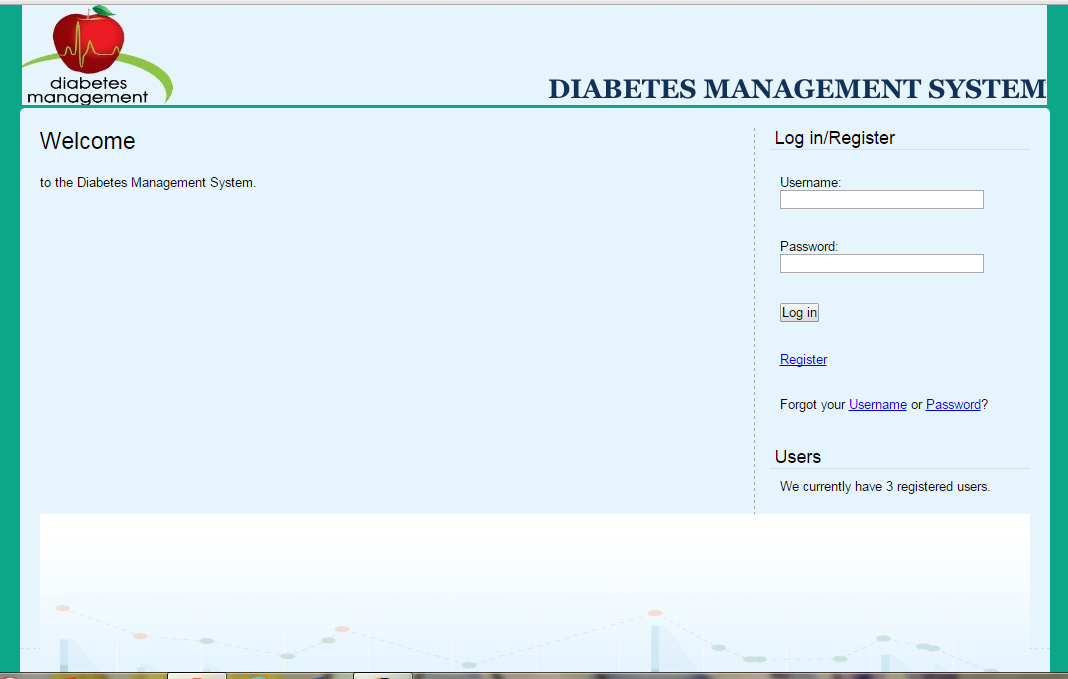
This user manual focuses on access, navigation, use and management of the Diabetes Management System.

* + 1. **PATIENT GUIDE**

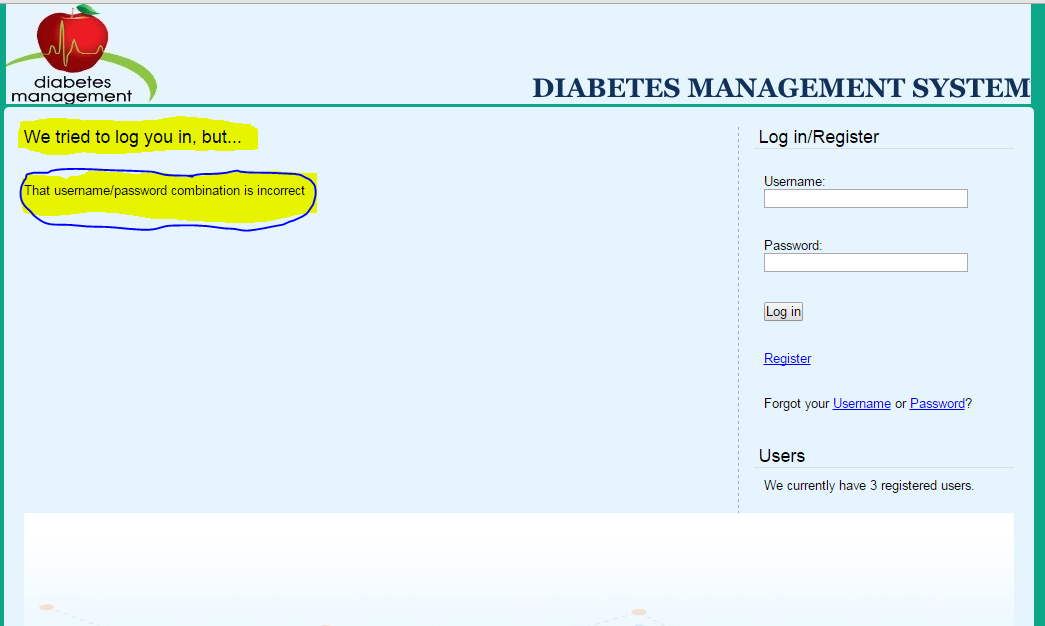
**Log into System**

The following steps below indicate how a user logs into the Diabetes Management System.

1. Firstly, the user accesses the Diabetes Management System via the URL **“www.diabetes-management-system.com”** on their browser.
2. The user enters their login credentials: **username** and **password** in the “Login/Register” section.
3. User logs in successfully or provided some assistance otherwise.



***The Diabetes Management System Login Screen***



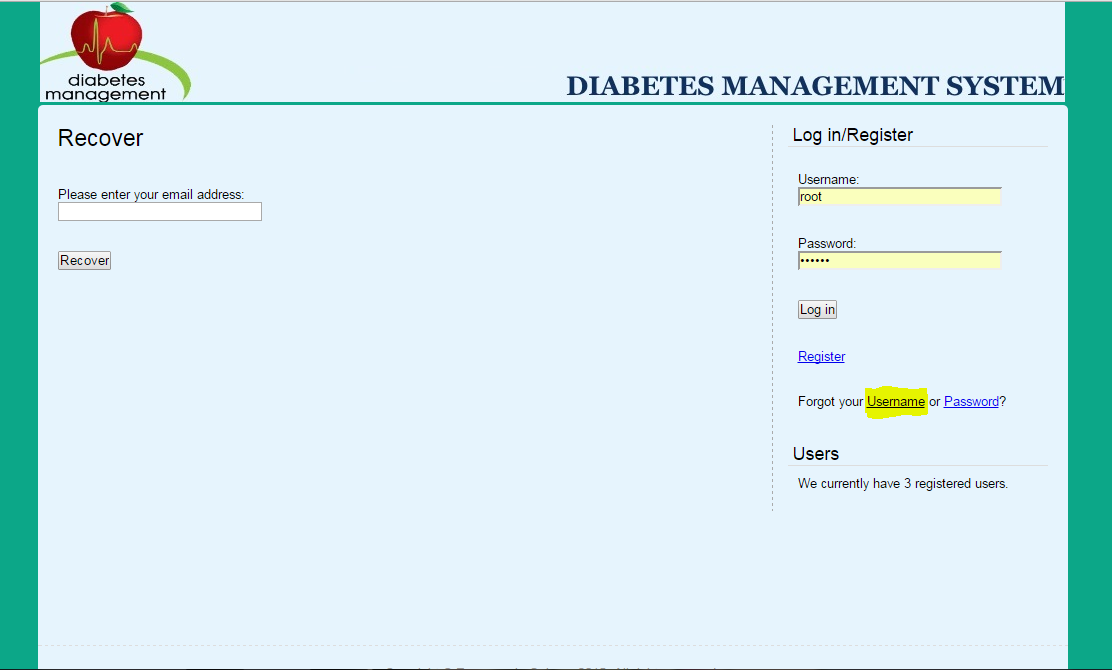
Here, the user is prompt to re-enter their credentials again.

For assistance, the user can **recover** their username or password via the following steps:

**Recover username**

Steps to recover username are as follows:

1. Select the **“Username”** option from the **“Forgot your Username or Password?”** on the home page.
2. The **“Recover”** panel will be prompted.
3. Enter your **“Email address”** in the email address field provided.
4. Click the **“Recover”** button.
5. An email will be sent to the email address provided in **step 3** above. Therefore, check the email to recover your username.

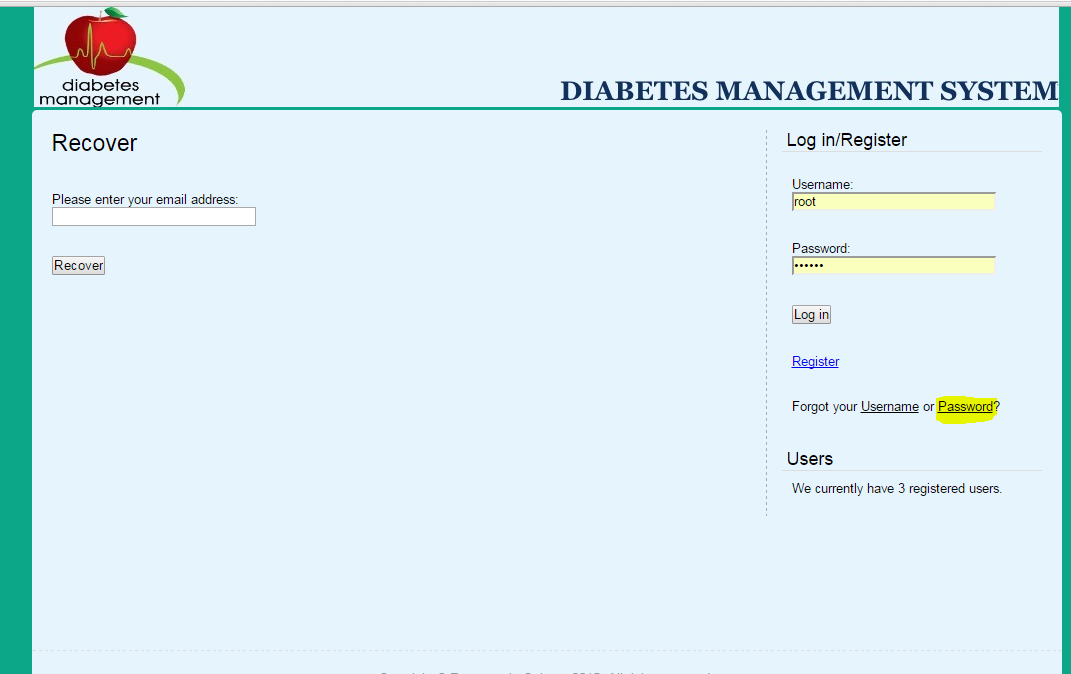


*Here*, *the user is prompt to enter their email address to recover their username.*

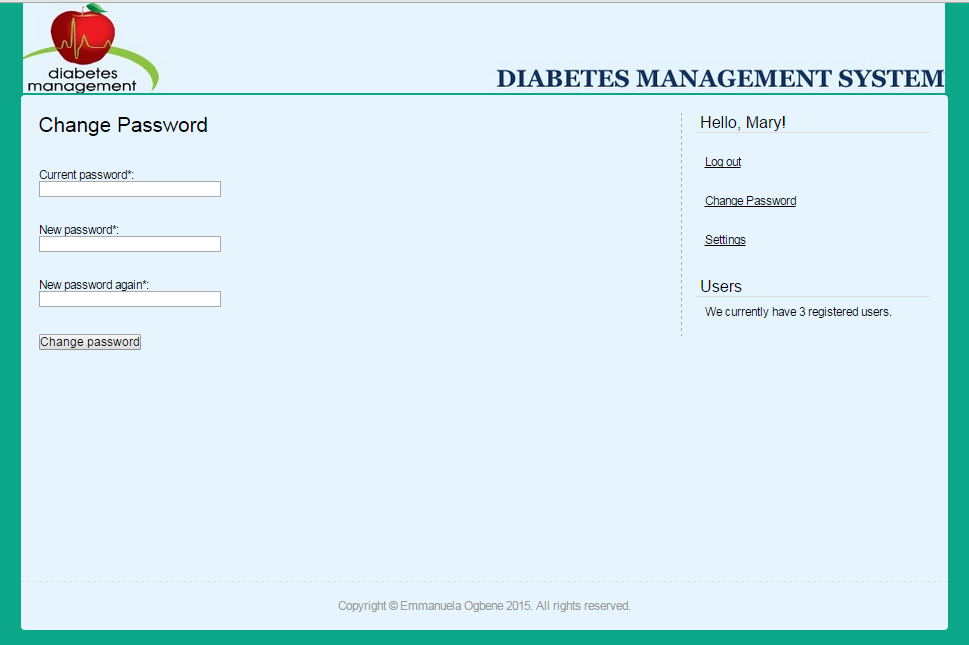
**Recover Password**

Steps to recover password are as follows:

1. Select the **“Password”** option from the **“Forgot your Username or Password?”** on the home page.
2. The **“Recover”** panel will be prompted.
3. Enter your **“Email address”** in the email address field provided.
4. Click the **“Recover”** button.
5. An email will be sent to the email address provided in **step 3** above. Therefore, check the email to recover your **new** password.
6. Then, log into the system by entering your username in the **“Login/Register”** panel. After, copy and paste the new password sent in your email address into the **password field** under the **“Login/Register”** panel. Click the **“Login”** button.
7. You will be re-directed to the **“Change Password”** panel as shown in the screen shot below.
8. Copy the new password sent in your email into the field **“Current password”**. Then, enter your new password into the field **“New password”**. Re-enter the password again in the field **“New password again”**. Afterwards, click the **“Change password”**.



*Here*, *the user is prompt to enter their email address to recover their password.*



*Here*, *the user is prompt to change their password.*

**View Personal Information**

The view personal informationauthorizes patients to view their personal information specified in the Use Case Narrative.

The steps below show how patients can view their personal information:

1. The patient logs into the system by providing their login credentials.
2. When the patient has successfully logged into the system, he/she is presented with their personal details.
3. Otherwise, they can click on the **“Profile”** tab which will display the patients’ information including their username, first name, last name, telephone number, email address, date of birth (DOB), address, and gender.

**Update Personal Information**

The update personal information enables a user to make desirable changes to their profile information. This includes any of the information specified in the Use Case Narrative. The following steps show how a patient and a doctor update their personal information in the Diabetes Management System.

1. Firstly, log into the Diabetes Management System.
2. The system should display your profile information.
3. Then, click on the **“Settings”** link that is located at the right hand side of the profile panel.
4. The **“Settings”** panel will be prompted. Here, make the necessary changes to the personal information and save the information by clicking on the **“Update”** button.
5. After, a confirmation page will be prompted to confirm that the information is updated successfully.

**Register into the System**

This use case allows anyone to sign up for the Diabetes Management System. The following steps show how a user can sign up for the system.

1. First and foremost, enter the applications URL as mentioned above into your browsers address bar.
2. The **login/Register** page will be prompted. Click on the **“Register”** link on the home page to display the **“Register”** page.
3. Provide the information required into the field and click on the **“Register”** button.
4. A message will be prompted indicating that the user is registered into the system. Also, an email will be sent to the user to activate their page.
5. Click the link in the email to redirect you to the home page where you login.

**Record Blood Glucose Level**

This describes the process of keeping track of Patients Blood Glucose Level in the system.

The steps below shows a patient guide in tracking their blood glucose level.

1. The patient logs into the system by providing his/her credentials.
2. The patient navigates to the **“Record Details”** tab as shown below. Click on the **“Enter your Records”** button on this page to record the daily blood glucose level.
3. The **“Blood Glucose Level Record Entry”** will be prompted. Here, the patient can fill in the requested fields and click the **“Save Report”** button.
4. After, click the **“View Blood Glucose Level Records”** button to view the blood glucose level readings for the day.

***Updating the Blood Glucose Level Records***

1. Navigate to the **“Record Details”** tab. Here, you should be able to see your records being displayed.
2. Make desirable changes to the record fields and click the **“Update”** button to save the updated information into the database system. **Note:** changes are made to the record based on the date displayed.
3. At this stage, the new detail is saved in the system.

***Deleting the Blood Glucose Level Records***

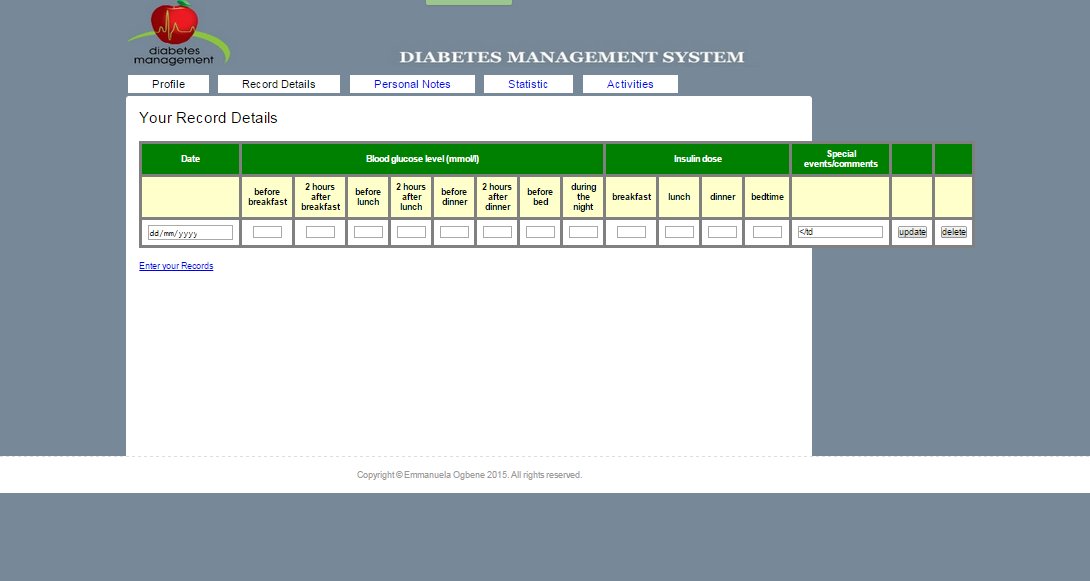
1. In order to delete a record, navigate to the **“Record Details”** tab and choose the record to delete.
2. Click on the **“Delete”** button next to the record that you want to delete. This will delete the records from the system as shown in the screenshot below.
   1. **VALIDATION TESTING**

The application was tested intensively. There were more than twenty issues encountered and recorded on GitHub. Here, I will mention some of the issues and how it was resolved.

Bug Report 1

|  |  |
| --- | --- |
| **Bug Logging Form** |  |
| **Product Information** |  |
| Product: | Diabetes Management System |
| Version (include backup for example): | 4.1.1 |
| Component: | Patients GUI |
| Reported Date: | 09/04/2015 |
|  |  |
| **Defect Information** |  |
| Steps to Reproduce: | Patient Page/ Record Details |
| Actual Results: | It seems that the table is overflowing in the container. Therefore, it needs to be resized to fix the container it is placed in. The table is not cutting off anything past the 280px limit being enforced. |
| Expected Results: | The table should not overflow. It should be hidden. |
| Attachments/shots: | **See Below** |
|  |  |
| **Defect Severity** |  |
| Severity: | 3 |
| Priority: | 5 |
|  |  |
| **Fix Information** |  |
| Fixed Date: | 09/04/2015 |
| Verified Build: | 4.1.1 |
| How it was fixed: | Fixed now by adding **display:block**; and **overflow: auto**; to the tablestyle. |
|  |  |
| **Bug Status** |  |
| Status: | Resolved |
| Resolution: | Fixed |

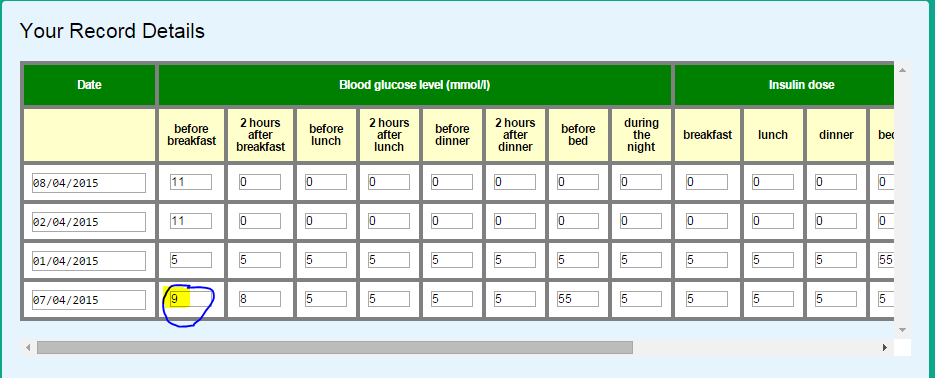
**Attachments/shots:**



Bug Report 2

|  |  |
| --- | --- |
| **Bug Logging Form** |  |
| **Product Information** |  |
| Product: | Diabetes Management System |
| Version (include backup for example): | 4.1.1 |
| Component: | Patients GUI |
| Reported Date: | 30/04/2015 |
|  |  |
| **Defect Information** |  |
| Steps to Reproduce: | Record Details on the Patients Page |
| Actual Results: | The blood glucose level record accepts integer values only. |
| Expected Results: | It should also be able to accept decimal values. |
| Attachments/shots: | **See Below** |
|  |  |
| **Defect Severity** |  |
| Severity: | 3 |
| Priority: | 5 |
|  |  |
| **Fix Information** |  |
| Fixed Date: | 30/04/2015 |
| Verified Build: | 4.1.1 |
| How it was fixed: | Fixed the problem by changing the database field from integer to varchar in order to accept decimal values for the blood glucose level readings. |
|  |  |
| **Bug Status** |  |
| Status: | Resolved |
| Resolution: | Fixed |

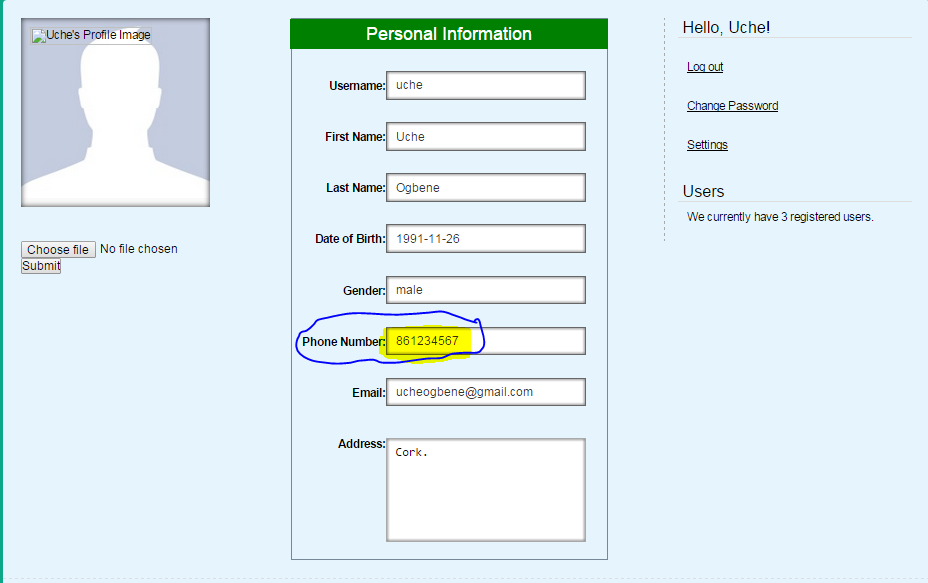
**Attachments/shots:**



Bug Report 3

|  |  |
| --- | --- |
| **Bug Logging Form** |  |
| **Product Information** |  |
| Product: | Diabetes Management System |
| Version (include backup for example): | 4.1.1 |
| Component: | Patients GUI |
| Reported Date: | 30/04/2015 |
|  |  |
| **Defect Information** |  |
| Steps to Reproduce: | Profile Information on the Patients Page |
| Actual Results: | The phone number is not accepting the correct format. It is missing a zero at the beginning of the number and also accepting characters. |
| Expected Results: | The phone number should accept only integer numbers and must be 10 digits valid phone number. |
| Attachments/shots: | **See Below** |
|  |  |
| **Defect Severity** |  |
| Severity: | 2 |
| Priority: | 3 |
|  |  |
| **Fix Information** |  |
| Fixed Date: | 30/04/2015 |
| Verified Build: | 4.1.1 |
| How it was fixed: | Was able to fix this issue by changing the structure of the phone number entity in the database to varchar (10). And, changing the field in the form from *tel* to *number* since Chrome and Firefox does not support this input type yet. |
|  |  |
| **Bug Status** |  |
| Status: | Resolved |
| Resolution: | Fixed |

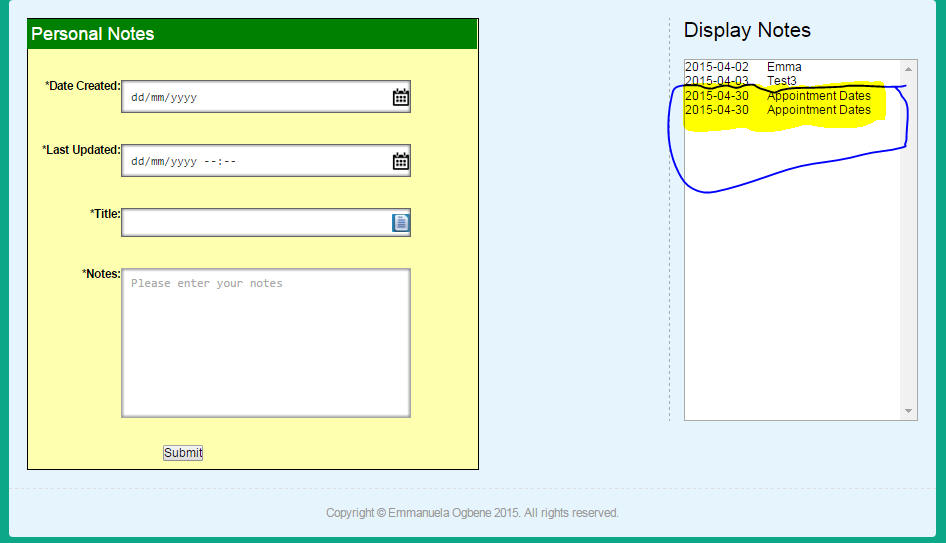
**Attachments/shots:**



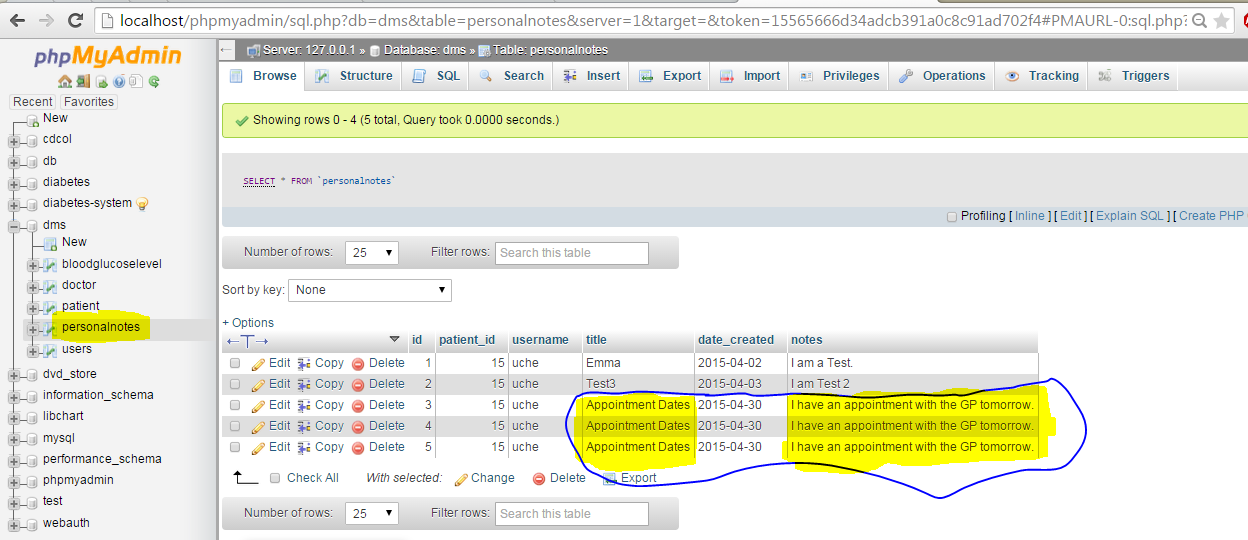
Bug Report 4

|  |  |
| --- | --- |
| **Bug Logging Form** |  |
| **Product Information** |  |
| Product: | Diabetes Management System |
| Version (include backup for example): | 4.1.1 |
| Component: | Patients GUI |
| Reported Date: | 30/04/2015 |
|  |  |
| **Defect Information** |  |
| Steps to Reproduce: | Problem with Personal Notes on the Patients Page |
| Actual Results: | When the browser is refreshed the second time, it displays the recently saved personal note twice, both in the database and in the GUI. |
| Expected Results: | Notes should only appear once. |
| Attachments/shots: | **See Below** |
|  |  |
| **Defect Severity** |  |
| Severity: | 3 |
| Priority: | 3 |
|  |  |
| **Fix Information** |  |
| Fixed Date: | 30/04/2015 |
| Verified Build: | 4.1.1 |
| How it was fixed: | Fixed the issue by changing the closing bracket of the validation section. |
|  |  |
| **Bug Status** |  |
| Status: | Resolved |
| Resolution: | Fixed |

**Attachments/shots on the patients GUI:**



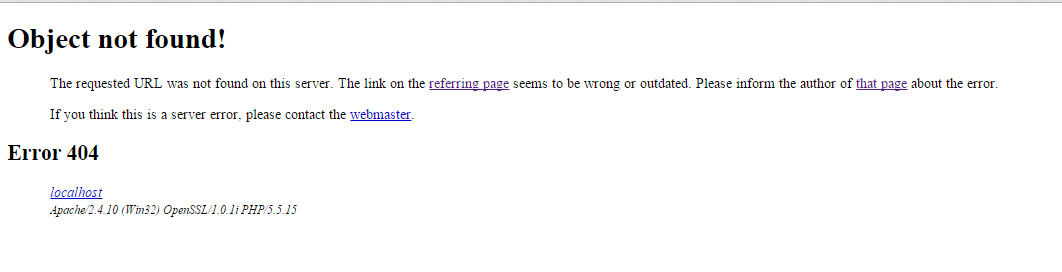
**Attachments/shots on the Database:**



Bug Report 5

|  |  |
| --- | --- |
| **Bug Logging Form** |  |
| **Product Information** |  |
| Product: | Diabetes Management System |
| Version (include backup for example): | 2.1.0 |
| Component: | Login Page |
| Reported Date: | 18/03/2015 |
|  |  |
| **Defect Information** |  |
| Steps to Reproduce: | Log in Page |
| Actual Results: | When a user provides his/her login credentials, they are not redirected to the right interface. |
| Expected Results: | Patients should be redirected to the patients' GUI, while Doctors should be redirected to the Doctors GUI. |
| Attachments/shots: | **See Below** |
|  |  |
| **Defect Severity** |  |
| Severity: | 5 |
| Priority: | 5 |
|  |  |
| **Fix Information** |  |
| Fixed Date: | 18/03/2015 |
| Verified Build: | 2.1.0 |
| How it was fixed: | It was fixed by providing a field *type* in the database, which determines who logs in. |
|  |  |
| **Bug Status** |  |
| Status: | New |
| Resolution: | Fixed |

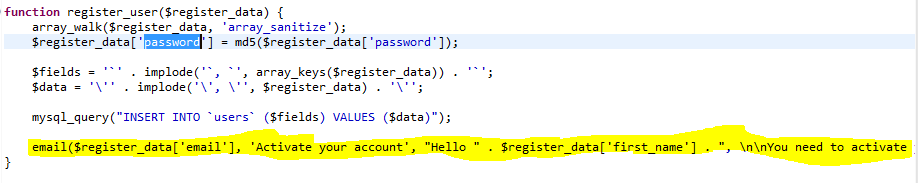
**Attachments/shots:**



Bug Report 6

|  |  |
| --- | --- |
| **Bug Logging Form** |  |
| **Product Information** |  |
| Product: | Diabetes Management System |
| Version (include backup for example): | 2.1.0 |
| Component: | Email Activation |
| Reported Date: | 18/03/2015 |
|  |  |
| **Defect Information** |  |
| Steps to Reproduce: | Email Activation |
| Actual Results: | When a user register for the system, email is not sending for the user to activate their page. |
| Expected Results: | A patient or doctor should receive an email verifying that, they have registered into the system and should be able to activate their page. |
| Attachments/shots: | **See Below** |
|  |  |
| **Defect Severity** |  |
| Severity: | 5 |
| Priority: | 5 |
|  |  |
| **Fix Information** |  |
| Fixed Date: | 18/03/2015 |
| Verified Build: | 2.1.0 |
| How it was fixed: | By modifying the **register\_user ($register\_data)** function, users are notified when they register for the system in order to activate their page. |
|  |  |
| **Bug Status** |  |
| Status: | New |
| Resolution: | Fixed |

**Attachments/shots:**



Bug Report 7

**Attachments/shots:**

1. **EVALUATION**

**Revised System Objectives**

First and foremost,

**Problems Encountered**

During the course of completing the ***Diabetes Management System***, I encountered the following problems such as:

1. **Log in/Registration Problem**

In the course of completing the login and registration process, I encountered some problems such as .

1. **GitHub Problem**

Because we are new to GitHub, we encountered many conflicts when we were merging our work into the Mater Repository. Due to the fact, we were working on the same file sometimes and in addition the two programmers were working on different speed. We resolved this by only allowing one person (the creator) to upload into the GitHub master repository, whereas the second programmer has to email his own code to the creator for integration.

1. **IDE - (Integrated Development Environment)**

There was a deliberation on which development platform to use and which of the IDE Platform will be of most benefit to the development of the desktop java application. Later, we chose NetBeans because it provided better qualities when using the available NetBeans tools to build our automated user interface. It was due to not having the time to write a java code for the design of the application user interface. In order words, we would have chosen Eclipse IDE to develop our java application.

1. **Coding & Debugging**

There were some tasks we found difficult to carry out during the completion of the software product such as uploading a profile-image of a Member, Instructor, and the Duty manager. In addition, assigning a member to a specific Instructor and there were some bugs that took quite some time to fix, e.g. changing or updating the content of a combo box.

1. **Accessing the Server:**

We used Wayne personal server for the implementation of our database. But not having direct access to the server prevented the other programmer from testing some codes quickly. Even though some stuffs were temporary.

1. **Time Constraint:**

Due to limited time available, it was quite difficult to experiment new ideas. Adding to this was the other course modules we participated.

**OutLook**

Because the product is only a desktop application we hope that, in the future a web-interface can be incorporated to the application, making the system very interactive with the end users, which was the main objective for the development of the Health Management System.

If more time and more were invested, the product could even be changed to act as a management system for areas such as an office, football or rugby club. The reason is that; the internal code is extremely flexible, generalizable and reusable.

In addition, businesses can also implement other features such as a Duty Manager personal information GUI. It can be possible because, the system is significantly flexible and adaptable to any environment.

With more time in the future, the payment use case could be implemented. In this way, the system will be able to keep a record of member payment and send them notice via automated email when their subscriptions are in arrears or when their payment is due. Moreover, adding a feature that will not grant them access to the system if they have reached their maximum payment allowed being in arrears.

**Schedule**

The original schedule was as follows:

**Week 1:** Research means of synchronising work and keeping in contact remotely.

**Week 2:** Research database systems for use in the project. What is viable?

Build a prototype interface design. (Non-programmatically)

**Week 4:** Implement database and link it to program prototype.

**Week 5:** Meeting Scheduled to analysis what was done so far.

**Week 10:** Working prototype GUI application and database interactivity. Documents

**Week 11:** Fully working prototype application. Validation Testing

**Week 13:** Completed Project and Documentation

It proved accurate enough. Most of the time we were actually well ahead of schedule: by week 6 we had already designed and created the database. By week 8, we had already created a working prototype, and by week 10, the program was feature-complete for the vertical slice we chose to implement.

1. **CONCLUSION**

In conclusion, I learned so much in the process of developing this online application.

I learned the useful source control tool Git. *Git* and *Eclipse PHP* have now become integral tools in my development processes, and most importantly is the best tool used. The reason is because; GitHub provides social networking functionality like network graph that displayed how I worked on my versions of my repository: **“DMS”**.

I learned new techniques of creating a secured application and creating a secured password for the end users of the application. This prevents unauthorized users from accessing confidential information.

Carrying out this project has greatly improved my programming skills; I take pride in the code architecture I created and the techniques I developed for it. I feel it will make a good showcase of my skills in the future and provide a useful basis for many other projects, academic and otherwise.

Moreover, I discovered the importance of destructive testing. Attempts to break parts of the program late in the development period were often successful due to the absence of this kind of testing earlier on. This almost proved disastrous in the case of the SQL injection vulnerabilities, but consistent use of good architectural design principles made the situation recoverable without much incident.

Additionally, the importance of planning became obvious early on. Had I not had a solid idea of what to implement, development would have been short-sighted and many features would now be absent due to implementation difficulty. For this reason, planning gives you foresight.

Also, the importance of quality requirement gathering cannot be emphasized enough. Due to quality requirements gathering, I had clear ideas of what needs to be done and how to get it done, in order to have a complete application. I was also able to integrate PHP to interact with the *MySQL* database. As a result, it improved the quality of the application I envisaged.