Code.Hub

Software Engineering Bootcamp
Java & Angular

powered by Pfizer

Sacchon

A Diabetes Management
Web Application

Software Requirements Specification Document

October 2020

Abstract

The final deliverable in the Software Engineering Bootcamp is the group project software, *Sacchon*, a diabetes management web application. Each team will create their own implementation based on a Software Requirements Specification (SRS) document. The SRS is common to all, but each team can follow their own assumptions and approaches. Sacchon is divided into two distinct subsystems, the *Sacchon Rest-API* which is the backend and the *Sacchon Angular App*, which is the frontend. To the end user, the web app consists of three major components: the Repository of Medical Data (*MediDataRepo*), the Doctor Advice Services System (*DoctorAdvice*), and the Reporting Services (*Reporter*).
MediDataRepo keeps track of the users' blood glucose level, daily consumed carbohydrates, and medication intake. In the DoctorAdvice section a team of doctors provide advice to patients on a per-call basis. Reporter provides a series of aggregation operations for creating tables and charts. Further details on the business requirements of each component will be discussed in the class.

Code.Hub

Table of Contents

1 Introduction

This Software Requirements Specification (SRS) document gives an overview of Sacchon, a diabetes management web application. It names what is required to be analyzed, designed, implemented, tested, and presented. It also describes the high-level requirements of the stakeholders based on their needs and expectations, i.e. the desired product features.

2 Project scope

The project aim is to develop the Sacchon app and deliver it ready to be released. The requirements are given by the Sacchon Consulting Enterprise (a fictional company).

The information system must contain

1. A User Interface to submit and view medical data by the patients and doctors.
2. An Administration Console with user management, monitoring, and reporting capabilities of the system usage.

For the above features to be considered complete, the following functionality must be provided

1. From the backend view, a corresponding call with a valid JSON object containing all needed information must be available, according to the specification of each case.
2. From the frontend view, a corresponding page consuming the service from the backend and presenting the suitable information must be available.

3 Deliverables

The deliverable of this project is the integrated platform of the back and the front system of the Sacchon. This means that the following three subsystems must be delivered:

- **MediDataRepo,** the Repository of Medical Data)
- **DoctorAdvice**, the Doctor Advice Services System
- **Reporter**, the Reporting Services

In the following sections, all needed functionality for each subsystem is defined.

4 Functional Requirements

4.1 Overview

The patient uses the system to store their data at any time. After a month of data recording has elapsed, a doctor can review this data and provide advice to the patient for the next month. This process is repeated until the patient or the doctor unregisters from the system.

The Chief Doctor is an officer who can view reports which monitor the activity of patients and doctors in the system.

The following sections list the functionality needed to be present for the system to be considered complete.

4.2 MediDataRepo

The patient can
- manage their account
  - sign up for an account
  - remove the account
- store their data
  - blood glucose level (date, time, measured in mg/dL)
  - carb intake (measured in grams)
- view
  - their average daily blood glucose level over a user- specified period
  - their average carb intake over a user-specified period
  - the current and past consultations from doctors
- update
  - modify incorrect submitted data
  - delete incorrect submitted data

4.3 DoctorAdvice

The doctor can
- manage their account
  - sign up for an account
  - remove the account
- view patient record
  - browse the data for a single patient (patient data and consultations)
- search
  - find patients that have not had a consultation in the last month
- consult
  - provide advice to a patient for the upcoming month (name of medication and dosage)

update
modify a consultation to a patient

Initially a patient has not doctor consulting them. As soon as a month of data recording has passed, the patient is ready to be consulted by a doctor.

Doctors search for patients for which advice is pending. Each doctor can see all the patients that they consult and the patients that are new in the system. Doctors cannot see patients that are consulted by other doctors. All doctors can see the new patients.

When a doctor selects to consult a new patient, the patient becomes managed by this doctor exclusively.

When a doctor leaves the system, their patients become available again for other doctors to consult. Past consultations of these patients continue to refer to the doctor who is leaving the system.

If a doctor modifies a consultation, the patient must see a warning as soon as they enter the system, so that they know that some important information must be reviewed.

A month is the period from the day of consultation to the same date next month (say, from the 15th of June to the 14th of July). It is not a calendar month. The next month for a patient starts as soon as a doctor has provided a consultation. Until the new consultation arrives, the previous one is considered valid.

4.4 Reporter

This section is available only to the Chief Doctors in the application. The reports available are:

1. The information submissions (personal monitor data) of a patient over a time range
2. The information submissions (consultations) of a doctor over a time range
3. The list of the patients who are waiting for a consultation and the time elapsed since they needed to have one
4. The list of the patients with no activity over a time range
5. The list of the doctors with no activity over a time range

5 Non-Functional Requirements

5.1 Sacchon app

The Sacchon app must be able to run in any current major web browser.

Performance
To guarantee a good user experience, all calls to the backend must have a maximum response time of 1 second (excluding network response time).

Version control
All code must make use of the Git version control system. Each team must create its own single GitHub code repository.

User interface

A basic UI design with open source images will be used.

5.2 Sacchon application code

Development environment

The following set up is required:

1. The backend part of the system must be developed in Java 8 or greater.
2. Any IDE can be used for the implementation; JetBrains IntelliJ IDEA is preferred (but not required).
3. The frontend will be developed using the Angular Framework.
4. The underlying database system must be the Microsoft SQL Server Developer Edition.
5. Maven must be the tool to manage dependencies and the software development lifecycle.

Software interfaces

All functionality must be implemented by using HTTP calls returning information in JSON format.

A software interface document must be created. This is to be read by external software developers who may want to interface with the system. The document must provide the following information:

- Give the list of communications including the description of the data that is being exchanged.
- Specify any constraints.
- Describe application or other software interface characteristics, including component names and versions, databases, operating systems, libraries, tools, etc. that must be known to the developers.

Testing

The system must be fully backed up by unit tests.

For testing the REST calls, a third-party tool should be utilized. Postman (https://www.getpostman.com) is recommended as a commonly-used tool in this area. Swagger can also interface with the deliverable for testing. Automating REST calls testing is highly recommended.

Logging

A well-defined logging policy maintaining all information produced by the software when running is mandatory. The logging policy must be documented (e.g. name the specific log files, directories, the rotation policy etc.).

Exception handling

When handling software exceptions, every call must be guaranteed to return a valid JSON document. This includes the cases where something went wrong in the backend execution due to either development bugs or system causes. The end-user must never see default error pages on the web browser. Depending on the type of the call (the various CRUD actions), the proper HTTP code must be always returned from the backend for further processing up front.

Deliverables

The application code must be delivered in a single project repository on GitHub, on the master branch of the project. The required software interface document must be included as project documentation in the same repository.