A Personal Assistant Project

Informatics 201

SJSU SAN JOSÉ STATE UNIVERSITY

Esteban T. Lopez
Professor : Dr. Benoît

October 11, 2020

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Project

A personal decision support system (DSS) that is focused on personal maintenance and goal setting. This is an attempt to concentrate a series of my personal projects into a sort of personal content management system (CMS). This is also an attempt by me to move beyond hack writing to the deliberate analysis, planning, and organization of a complex project. (Hack writing means haphazardly engineering software without adequate planning or best practices, not the negative popular culture connotation of illegally breaking into systems). The personal projects I have been tinkering and hacking on are projects that many people have interest in. Though this is for personal use, this could eventually be expanded for public use. This design will be treated as a public or commercial software engineering project; Initially, it will be for personal use. The basic categories are health, finances, and professional/education. The design will allow for expansion of categories.

The initial concept of the user interface (UI) of this program is to have tiles in a key performance indicator (KPI) graphical user interface (GUI) as in a dashboard. The KPI tiles will show current states and trends. The central idea is to see weight trends and body fat % trends for the health category. Also debt and income trends for the financial category. For the professional/education things like progress on degrees or certificates and updating resume and job applications using education and career history. This is an ambitious, but feasible project. This will be tackled modularly and in phases; Taking advantage of hack writing skills but consolidated into a larger and managed project. The managed part is important not only for completing the project, but for future features. For example, a feature in health showing calories needed and meals eaten analyzed against the finances of cost and budget. This DSS will not be complete with this document, and by nature, will never be completed. The Unified Modeling Language (UML) will be attempted throughout this document. This document represents an expansion of planning and organization skills as well as the support skill of UML.

1.1.1 Approach

The philosophy of this project is to use the expertise and best practices of professionals and academics to form and present a design of a personal software project. The purpose is not to profess expertise in any methodology of a software development life cycle (SDLC), of writing pseudocode, or of diagramming. Basically, the approach here is to explain the software being developed, and using a personalized hybrid to get the project started, organized, and developed.

1.1.2 SDLC

The main 2 types of software development life cycle (SDLC) that will be used as models are Waterfall and Agile. In a sense, the existence of this document is about planning beforehand; Waterfall. Waterfall is a SDLC that emphasizes planning an entire project before beginning the programming. Agile approaches SDLC in an iterative fashion. A robust defense of Waterfall is not ever-present, usually Waterfall is invoked as a post against which other SDLC approaches are measured and justified. Agile does not espouse documentation. ”Working software over comprehensive documentation” is a specific declaration of the Agile Manifesto (Layton and Ostermiller, 2017, p. 22). Documentation definitely takes time away from writing working software. Documentation gives a structure and deliberate design to working software.

I have created many modules in many different languages with the intention of creating a central conglomeration of working software. Writing bits of code does produce working prototypes, but having a plan corrals the elements to be used together in an organized way. The plan here is to use Waterfall to organize and document this DSS. The software engineering will be tackled using Agile principles. Creating documentation and using UML probably violates Agile principles just a little bit, or a lot. One of the "platinium principles" of Agile is "resisting formality" (Layton and Ostermiller, 2017, p. 36). While there will be documentation, the project will be split up into Agile iterations. Agile splits up projects into iterations, parts of projects are prioritized into different iterations and handled in sprints (Layton and Ostermiller, 2017, p. 46). At the end of each sprint a fully functional piece of software is delivered. This will be the approach here, but documentation and optimization will be used since this is a personal project. This project design will be slowed down by learning to diagram in UML,and pseudocode, while learning Agile project management. The purpose is to not only organize this labor of love, but to expand design and software engineering experiences.

As a side note, I did typeset this document in LaTeX; I wanted to feel like I was programming while designing this document. It was a bit of a learning curve, still on the curve. LaTeX is open source, and allows for separation of style and content (Kottwitz, 2011, pp. 10-11).

Chapter 2

DSS Overview

2.1 DSS

Any information system has a variety of models, DSS are no different. One model represents a DSS as having a user, user interface (UI), database management system (DBMS), and model base management system (MBMS) (Sauter, 2011). This high level abstraction will be used to present a UML model as an overview of this DSS. The decision maker is the reason for this particular DSS, especially the user creating the system in this case. While the decision maker and UI is not the only part of the DSS, the following UML use case diagram will be shown to illustrate a high level concept of this system. The following diagram does not only show the UI, it is emphasized. “To the decision maker the user interface is the DSS” (Sauter, 2011, p.215). To the designer all the components of the DSS are the whole system.

2.1.1 UML Use Case

This DSS will basically have 3 general initial options for the decision maker.

1. Create an account
2. Login to account
3. Use modules without logging in

To maximize the use of the system, having an account with logged data in the database will be required. However, the system will allow basic models be used and calculations be done without the benefit of previous data.The UI will have to be managed in order to organize and integrate all components of the DSS. As explained, the DSS has a decision maker that uses the UI to connect to the MBMS and with an account the DBMS.

The first entry point for a decision maker is to create an account. If the account is created the decision maker can then login and use personal historical information and enter new information. The decision maker with an account will have access to modules and KPI dashboard.

Another option is for a decision maker to use modules without logging in. They will not have access to database and historical information, and MBMS modules that depend on historical information. But they will be able to generate one time reports and perform some decision making analysis.

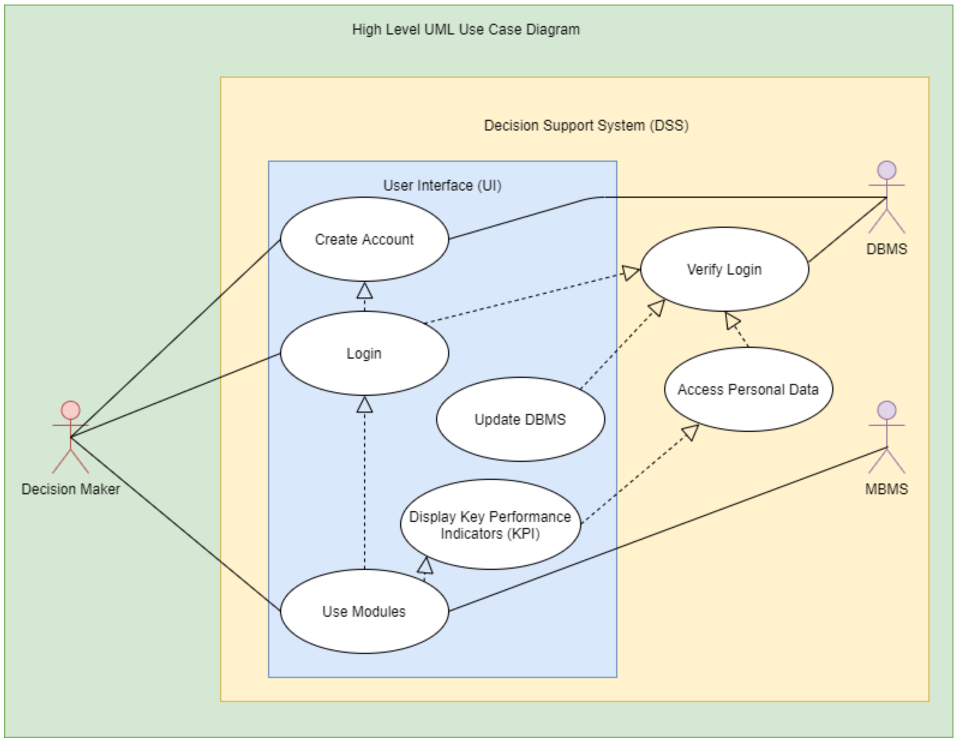

Figure 2.1: DSS UML Use Case Diagram

2.1.2 DSS Manager

Even though the DSS is represented at a high level as having a decision maker using the UI with the DBMS and MBMS, there needs to be a manager. We will call this the DSS manager. This is necessary to move beyond the high level design and actually engineer a real program that can be controlled. The high level design is very useful to diagram the DSS use case, but will probably also be useful to diagram the DSS activity at a high level.

2.1.3 DSS Pseudocode

The pseudocode below is just to give an overall idea of how the DSS is accessed. Treatment and project approach will be addressed.

Algorithm 1: DSS UI Entry

```
if !Account then
  Create Account
else if Account then
  Login
  Verify Account
  Load Database Info
  Display KPI
  Allow Logging New Info to Database
  Use Modules With Database Info
else {!Login}
  Use Modules (MBMS) Without Benefit of Database Info
end if
```

Chapter 3

Agile Iteration 1

3.1 DSS Foundation

In a sense, Iteration 1 flies in the face of Agile principles. There will not really be a completely useful piece of software after this iteration. It is a personal project and ensuring that the components of the DSS are in place is essential. The most important part, in order to be able to use historical information to see trends and model results is having a database. This is also important to scale up if this is made commercial, or even just to share with friends and family. I have previously created a database to track substitute teaching earnings and paydays, I did not plan ahead to add the ability for different users apart from myself. It still needs refining, but this is an example of why planning ahead for different possibilities is beneficial.

3.1.1 DBMS

DBMSs are usually modeled using Entity-Relationship (ER) models, but can be modeled using UML class diagrams (Elmasri and Navathe, 2011, pp. 199-200). UML class diagrams have object-oriented programming in mind (Hamilton and Miles, 2006, p. 65). The idea of having data tables and having procedural language in the same graphical image is interesting. Since this is a DSS, the procedural language feature might violate the conceptual separation between DBMS and MBMS (see Figure 2.1). A simple UML class diagram (see Figure 3.1) in order to set up a simple database is all that is needed for the DBMS in Iteration 1. There are things to consider for future iterations, such as user address and phone number, user recovery email. But this is not necessary for Iteration 1.

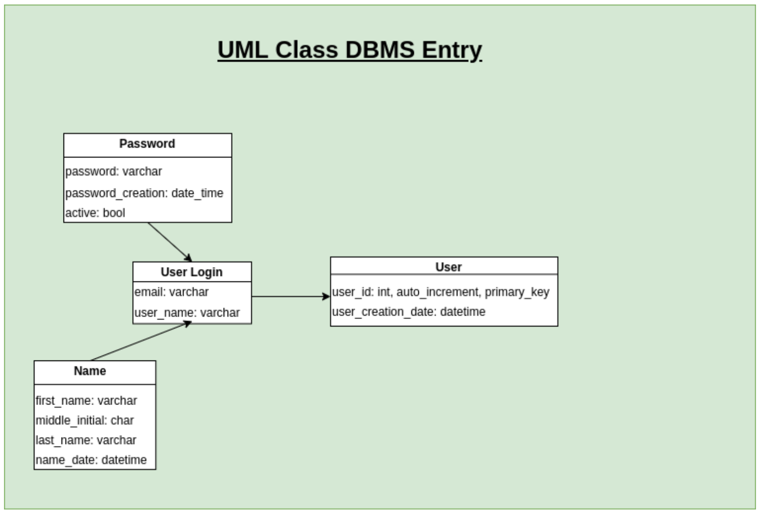

Figure 3.1: UML Class DBMS Entry Diagram

3.1.2 UI and MBMS

In order to make sure the framework is complete, a simple UI and MBMS need to be set up to test the DBMS. Simple tests using HTML forms with PHP should be sufficient, possibly even database connection tests using Python. The DBMS procedural language will be considered separate, and will be written in separate scripts; It really is part of the DBMS, but does make sense as a feature of the MBMS. The advantage of modular components is being able to select different languages that users can use to interact with the database and the models.

3.1.3 Summary of Iteration 1

In order to set up the DSS and get through Iteration 1 quickly the following languages and systems will be used. The UI will be developed using localhost browsers, so it will be developed using HTML, CSS, and Javascript at the front-end. The back-end for the MBMS will primarily be PHP, but since the goal is to have a fully functional DSS Python will be incorporated. Python is used for analytics and in order to make the program powerful, Python (or C++) will have to be implemented. The final part will be to use MySQL as the DBMS because a Linux operating system will be used and MySQL is pretty standard. There is no real usable software, which violates Agile principles; This is a very important component to get the rest going while keeping it organized.

- UI
  - HTML
  - CSS
  - Javascript
- DBMS
  - MySQL
- MBMS
  - PHP
  - Python
- Development Operating System
  - Linux (Ubuntu)

Chapter 4

Agile Iteration 2

4.1 Health Module

Without going into the details of goal setting, this module is being created with the intention of monitoring and making decisions on weight loss. This could be used for weight gain or maintenance as well. The idea is to have software that allows weight to be entered, along with body fat%, height, weight, age, and sex. The output would show a range of body fat%, lean body mass, fat only weight, body mass index, basal metabolic rate. The DSS portion would allow for sensitivity analysis by seeing what caloric needs would be based on the basal metabolic rates at different body weights, by seeing the weights at different body fat percentages and body mass index.

4.1.1 Pseudocode for Initial Health Module

Algorithm 2 gives an overview of the information that would be provided to the DSS and what the output should be. Not all the information is needed for this iteration of the health module. The essential information is $Weight$ and $Body\ Fat\ Percentage$ to calculate $Fat\ Only\ Weight$, $Lean\ Body\ Mass$, and $Weight\ at\ Range\ of\ Body\ Fat\ Percentages$. Algorithm 3 shows the formulas of how to arrive at $Fat\ Only\ Weight$ and $Lean\ Body\ Mass$. Algorithm 4 displays the way a list of $Weight\ at\ Range\ of\ Body\ Fat\ Percentages$ is created based off of $Lean\ Body\ Mass$. An Agile meeting would be able to help fill out the rest.

Algorithm 2: Initial Health Module Overview

```
ENTER:
$Weight$
$Body\ Fat\ Percentage$
$Height$
$Age$
$Sex$
OUTPUT:
$Weight\ at\ Range\ of\ Body\ Fat\ Percentages$
$Body\ Mass\ Index$
$Lean\ Body\ Mass$
$Fat\ Only\ Weight$
$Basal\ Metabolic\ Rate$
```

Algorithm 3: Body Composition

```
Require: $Weight > 0 \wedge Body\ Fat\ Percentage > 0$
  DATA:
  $Weight$
  $Body\ Fat\ Percentage$
  CALCULATE FAT:
  $Fat\ Only\ Weight = Weight * (Body\ Fat\ Percentage/100)$
  $Lean\ Body\ Mass = Weight - Fat\ Only\ Weight$
  return $Fat\ Only\ Weight \wedge Lean\ Body\ Mass$
```

Algorithm 4: Range of Body Fat Percentages

```
Require: $Lean\ Body\ Mass > 0$
  DATA:
  $Lean\ Body\ Mass$
  CALCULATE BODY FAT PERCENTAGES AND FILL
  ARRAY OR LIST:
  for $i \leftarrow 1$ to $100$ do
  |  $Body\ Fat\ Array[i] = Lean\ Body\ Mass * (i/100)$
  end
  return $Body\ Fat\ Array[]$
```

4.1.2 Future Iterations on Health Module

Concepts: Workout, blood pressure, diabetes, pain, temperature, oxygen, sleep patterns, medication log, nutrition log, etc. Combined with financial module and coordinating nutrition with expenses. The health module can be a complete DSS, but as a personal assistant it would be nice if the program enveloped many aspects of personal life; especially tedious things like health and finances.

4.2 Final Thoughts

4.2.1 Experience

Learning how to diagram, write pseudocode, and design versus hack writing has been an enlightening experience. I spent the last year substitute teaching and design skills are very useful, and I believe can translate to teaching young students and older students alike. I see parallel skills between teaching and designing to direct and help software teams. I definitely learned alot about typesetting in LaTeX, I believe I can connect more deeply with the plight of 2nd graders struggling learn Google docs. While I am proud of being able to write code, I believe helping others understand and work in tandem would be euphoric. This experience has catapulted my technology skills into a world that substitute teaching tugged me into. I appreciate diagramming, pseudocode, and planning more than I ever have. I also appreciate the art of inforgraphics and the deceptive ease with which they convey information. I want to excel at conveying information to anybody, whether in teaching, managing projects, or informatics. I am not there yet, but aspire to be.

References

Elmasri, R. and Navathe, S. B. (2011). *Fundamentals of Database Systems (6th Ed.)*. Addison-Wesley, Boston, MA.

Hamilton, K. and Miles, R. (2006). *Learning UML 2.0*. O'Reilly Media, Inc., Sebastopol, CA.

Kottwitz, S. (2011). *LaTeXBeginner's Guide: Create high-quality and professional-looking texts, articles and books for business and science using LaTeX*. Packt Publishing, Birmingham, UK.

Layton, M. C. and Ostermiller, S. J. (2017). *Agile Project Management for Dummies (2nd Ed.)*. John Wiley & Sons, Inc., Hoboken, NJ.

Sauter, V. (2011). *Decision support systems for business intelligence (2nd Ed.)*. Retrieved on September 13, 2020, from, https://learning.oreilly.com/library/view/decision-support-systems/9780470433744/.

Glossary

LaTeX Is a mark up language specially suited for scientific documents. 3, 12

Agile The Agile software development life cycle (SDLC) is a project management style that delivers fully functional modules of software. Agile prioritizes software features and delivers those first. 2, 3, 7, 9, 10

content management system A content management system (CMS) is software that controls how information is created, stored, and retrieved. 1

dashboard A dashboard is a graphical user interface (GUI) that usually displays key performance indicators (KPI's). 1, 5

database management system A database management system (DBMS) is a database with a SQL query system, and usually provides some sort of procedural language support.. 4

decision support system A decision support system (DSS) is an information system that gives a decision maker support. 1

graphical user interface A graphical user interface (GUI) is a user friendly way to interact with software. 1

key performance indicator A key performance indicator (KPI) is a measure and is usually used as a tile with many KPI's on a dashboard. A few KPI examples for this project is: how much do I weigh today, or the weight average for the last week. 1

model base management system A model base management system (MBMS) is just where the models and programs are stored. This is an entirely new concept to this author as of this writing. It is unclear if there is any comprehensive MBMS, for example is Keras or Tensorflow considered an MBMS. 4

pseudocode Conceptual code that is written to convey the overall concept of an algorithm or piece of code. This is not strict coding language, but is widely used to explain concepts and design in the manner of code. Many beginning programming and algorithm textbooks use pseudocode, there is no universal standard; that would defeat the purpose. Textbooks usually explain their own standard. 2, 3, 6, 12

software development life cycle A software development life cycle (SDLC) is a project management approach to developing software. There are many SDLC approaches. Two prominent frameworks are waterfall and agile. 2

Unified Modeling Language The Unified Modeling Language (UML) is an attempt to standardize a general purpose visualization and diagramming of programming and softwarehardware systems. 2

user interface A user interface (UI) is the way a user interacts with software. A popular UI is a web browser with buttons, usually called a graphical user interface (GUI). Not all UI's are GUI's. 1, 4

Waterfall The Waterfall software development life cycle (SDLC) is a project management style that has every step planned out with a substantial amount of detail before beginning a software project. This approach produces and delivers a complete software project. 2

List of Acronyms

CMS content management system. 1, *Glossary:* content management system

DBMS database management system. 4–9, *Glossary:* database management system

DSS decision support system. 1, 2, 4–8, 10, 12, *Glossary:* decision support system

GUI graphical user interface. 1, *Glossary:* graphical user interface

KPI key performance indicators. 1, 5, 6, *Glossary:* key performance indicator

MBMS model base management system. 4–9, *Glossary:* model base management system

SDLC software development life cycle. 2, *Glossary:* software development life cycle

UI user interface. 1, 4–6, 8, 9, *Glossary:* user interface

UML Unified Modeling Language. 2–4, 7, *Glossary:* Unified Modeling Language