**REFLECTIVE REPORT**

**Z02DT – SOFTWARE AND dATABASES**

**Student Id: 14536106**

**Student Name: Roman Monsengwo Iyualeke**

Table of Contents

**No table of contents entries found.**

# List of Illustrations

**No table of figures entries found.**

# List of TABLES

**No table of tables entries found.**

# Introduction

This report aims to give a bigger scope over software development and database handling; using different tools and techniques to create programs efficiently. This report will be reinforced by code artefact made during the writing of this report, with a given deadline (six weeks). Many lessons taken out of this first experience in coding, data manipulation and even consideration of ethical issues in software design and data storage.

This is all a beginner’s view of these said features of reflective writing and programming, thus should not be considered as fact or as a professional’s word but as a realisation after one’s personal experience.

# Findings and Analysis

## Software Development LifeCycle (SDLC)

Software development lifecycle is a structured process consisting of multiple phases (seven phases; feasibility, planning, requirements, design, construction, testing/installation, maintenance/review). The most well-known software development lifecycle methodologies are Traditional and Agile. Traditional consists of following a strict order, previous tasks have to be done in order to go forward whereas Agile follows a more “swift” approach allowing multiple tasks to be tackled at the same time.

For the production of the code artefact, an Agile approach was used given the fact that the result was known but not the way of reaching the said result, unfortunately though not a solid enough structure in the planning making the process of building the code quite random, though, “The principle of agile software development proposes that “at regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly” In other terms it may be said that agile methodology addresses exactly the challenges of an unpredictable, disordered business and technology environment” (Kumar & Kumar Bhatia, 2012, p. 46), thus thanks to the usage of Agile it isn’t much of a problem because it allows one to go back at any given time, it would’ve been a problem if Traditional was employed, as change in previous tasks cannot be made unless the project in its whole is reset.

As a result, it would be better to plan projects more thoroughly by following the seven phases of software development lifecycle though the Agile approach allows a bit of laxism and has proven to be pretty efficient. Especially in software development where the means of reaching a goal are not well-known; to maximise efficiency and easily meet customer needs.

## Requirement Elicitation and Engineering

For the artefact in question, the requirements are the given; implementation of Object-Oriented Programming, CRUD support, Database Management System connectivity, Entity Relational Diagrams for understanding, normalized 3NF database and a sorting/searching algorithm to filter loaded data.

FURPS describes the key features a program should fulfil (functional, usable, dependable, performance, supportability).

MoSCoW considers the musts, should, could and will not haves in a program.

The importance of these requirements is crucial as it helps establish clear goals for the program whereas the use of FURPS and/or MoSCoW can only bring benefits.

Making a FURPS/MoSCoW, thus fusing both, allows one to plan and section their goals in an organisational manner, also allowing reviewers to easily understand what is being done (Dyson, 2019). Making minor changes as the project is being built as it allows one to see what has been achieved and what there is yet to achieve, with extra aspects that could be approached for further development.

For future reference, planning a FURPS/MoSCoW should be a priority before taking on the project’s construction, it should also be customisable if the means of reaching a goal are not clearly defined (see Appendix 1).

## Object Oriented Programming (OOP) Principles

Object Oriented Programming is a programming paradigm that organizes data and with objects. These objects contain the data or the code. Object Oriented Programming uses these objects and data rather than logic-based actions or traditional functions.

For the realization of the code artefact, the use of Object-Oriented Programming did not seem necessary, in fact, even when purposefully looking for ways to implement it, it felt like over complicating the program or less straightforward. With a bit of research, it can be seen that amongst programmers Object Oriented Programming is a fair bit disliked (see Appendix 2 for extra information). In the case of the artefact, SQL query was favoured and seemed easier and more straightforward to manage whereas classes in Object Oriented Programming were mainly used to sort functions.

Unless further use is found or Object-Oriented Programming seems more fitting than any traditional programming functions, it would be pointless to use Object Oriented Programming over regular code.

## Programming Constructs

For the creation of any kind of program, it is necessary to understand what most programming constructs do. Programming constructs such as conditional statements, iterations and functions are the heart, the muscles and the brain of a program; just like how one’s heart pumps, one uses their brain or trains their muscles, more practice, thus, better understanding of these constructs leads to better and more efficient programs.

My proudest, yet simple piece of code, is this small piece of code (see Appendix 3) that allows me to tell if a string contains a specific character (uppercase, lowercase, space,… etc.). My first actual, working, useful function which will not be exclusive to this artefact and most probably be especially useful for future projects.

Let it be success or failure one can always take a lesson out of using programming constructs and their endless possibilities. (myself, 2024)

## Database Design

### Database Design (ERD AND NORMALISATION)

For database design, the easier, the better making SQLite a great option. There is much to learn from using SQLite, such as how SQL works and it is easy to implement in python, not needing any host, making SQLite beginner friendly. Though it would be better to first build an ERD, before starting a database; doing so guarantees high quality database design in various aspects such as management, creation and maintenance (VisualParadigm, n.d.)

For the ERD provided (see Appendix 4), it can be seen that there are two different entities for student/teacher separating user details from login details (username, password,… etc.), this choice was made in respect of 3NF which states that, “A relation is in the third normal form, if there is no transitive dependency for non-prime attributes as well as it is in the second normal form” (GeekforGeeks, 2023). There are also many primary-foreign key links to respect 3NF whilst maintaining an explicit link with every entity, mostly surrounding the two main tables (Student\_Users, Teacher\_Users). The making of this diagram has also helped better understand, how to establish and understand how relationships between tables work, resource which will always be useful for the realization of software development related projects, let it be this one or prospective programs (see Appendix 4 for more info).

”

### Database Design (SQL)

SQL queries are the core of the artefact, with its many queries (SELECT, INSERT INTO,… etc.) they provide versatility allowing data retrieval with SELECT and WHERE EXISTS or even data deletion with DELETE.

The importance of SQL in database design has helped build a foundation on how code works fundamentally serving well as an introduction to the greater scope what programming is. It has helped, on a smaller scheme of things to build the artefact and has helped to grow closer to understanding the key values of software development and gives great scope on what can be achieved for potential future projects and databases; may it be in SQLite or in python SQL has proven to be very useful and will most likely be for even the most complex programs. (see Appendix 6 for demonstrations)

## TESTING

This is pretty self-explanatory but testing allows one to verify if their program functions as intended.  
 In this case, it was heavily relied on to confirm that each command and role works properly, it also helped with error handling as it shows where the issue in the code is and the cause of the said issue.  
 Writing more dependable, efficient, and maintainable code is encouraged by testing. Developers can make sure that code functions as intended and satisfies project quality standards by regularly testing it. (Samuel, 2023).

## Consideration of the ethical issues in software design and data storage

“The General Data Protection Regulation covers personal data, that is, any data that uniquely identifies a living person. It also covers data that is listed as “special categories” or that which may be regarded as” sensitive.” For example: Ethnic origin, trades union membership or sexual orientation” (Denley & Foulsham & Hitchen, 2019, p. 7).

In respect of GDPR, every account has a unique username and personal password to ensure security as well, making each account accessible by the owner of the account, and for accounts such as teacher accounts or admins, who are able to view student accounts, limit the access by hiding certain parts of data (password).

So, to respect GDPR and be exempt of its penalties, give limited access to ‘foreign’ accounts when viewing other accounts by hiding personal info such as passwords and have the restricted data locked in the database to which a limited amount of people should have access to.

# Conclusion and Areas of Further improvement

Through thorough planning, by using an Agile approach which allowed every aspect to be reviewed, software development lifecycle to “pave to way” towards the construction of this artefact whilst following the given requirements, not much Object-Oriented Programming to show off for however great use of SQL and databases, in respect of normalization and by following a detailed diagram of the framework (ERD). All this whilst respecting General Data Protection Regulations.

Many resources, which will serve me well along this course and hopefully career in software development; the result of these six last weeks can been seen through the code artefact and this report.

Though if more time were to be given future development of the program such as adding a GUI or a mobile application would be nice and in the future possibly find ways to insert Object-Oriented programming to gain more knowledge on the subject, less represented than any other aspect in the code artefact.

# List of References

Denley, A. & Foulsham, M. & Hitchen, B. (2019). *GDPR: How To Achieve and Maintain Compliance.*

<https://books.google.co.uk/books?id=Vzr3DwAAQBAJ&dq=what+is+gdpr+software+compliance&lr=&hl=fr&source=gbs_navlinks_s>

Dyson, J. (2019). *Conjoining FURPS and MoSCoW to analyse and Prioritise Requirements.* [*https://www.linkedin.com/pulse/conjoining-furps-moscow-analyse-prioritise-jonathan-dyson*](https://www.linkedin.com/pulse/conjoining-furps-moscow-analyse-prioritise-jonathan-dyson)

GeekforGeeks. (2023). *Third Normal Form (3NF).* <https://www.geeksforgeeks.org/third-normal-form-3nf/>

Gillis, S. A. (2016). *object-oriented programming.* <https://www.techtarget.com/searchapparchitecture/definition/object-oriented-programming-OOP>

Kumar, G., & Kumar Bhatia, P. (2012). *Impact of Agile Methodology on Software Development Process*. <https://www.researchgate.net/profile/Gaurav-Kumar-175/publication/255707851_Impact_of_Agile_Methodology_on_Software_Development_Process/links/00b49520489442e12d000000/Impact-of-Agile-Methodology-on-Software-Development-Process.pdf>

Samuel, A. (2023). *5 Reasons Why Testing Your Code is Essential for Software Development.* <https://www.linkedin.com/pulse/5-reasons-why-testing-your-code-essential-software-adedayo-samuel#:~:text=Improve%20code%20quality%3A%20Testing%20encourages,standards%20set%20for%20the%20project>.

VisualParadigm, (n.d.) *What is Entity Relationship Diagram (ERD)?.*

<https://www.visual-paradigm.com/guide/data-modeling/what-is-entity-relationship-diagram/>

# Appendix 1 – Requirement Catalogue USING FURPS & MOSCOW

A table of information

Description automatically generated

# Appendix 2 - PYTHON Code Samples Screenshots of OOP

Object Oriented Programming used to organize functions:

A screenshot of a computer program

Description automatically generated

Extra information from the programming community:

<https://stackoverflow.blog/2020/09/02/if-everyone-hates-it-why-is-oop-still-so-widely-spread/>

<https://www.reddit.com/r/learnpython/comments/1318dwg/when_is_oop_necessary/>

Even though these are not expert review, the shear amount of attention this has amassed amongst the community and the recurrency of similar statements can serves as a valid argument.

# Appendix 3 - PYTHON Code Samples Screenshots of PROGRAMMING CONSTRUCTs

First function I made; with an if/else statement and a “for” iteration, all that in a defined function.

A screenshot of a computer program

Description automatically generated

# Appendix 4 - Entity Relationship Diagrams (Conceptual, logical, and physical)

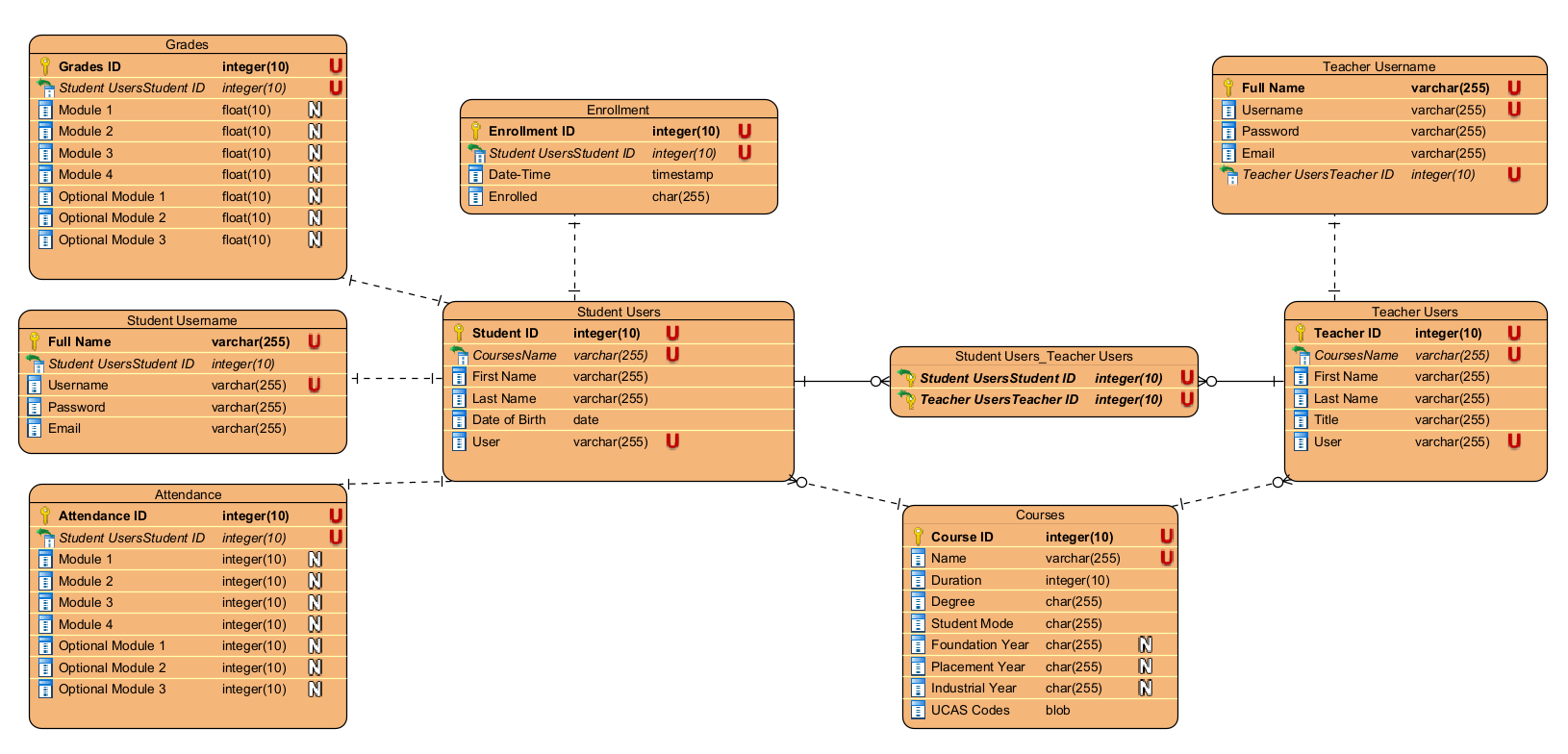
U = Unique

N = Can be null

Key = Primary Key

Green Arrow = Foreign Key

Physical ERD

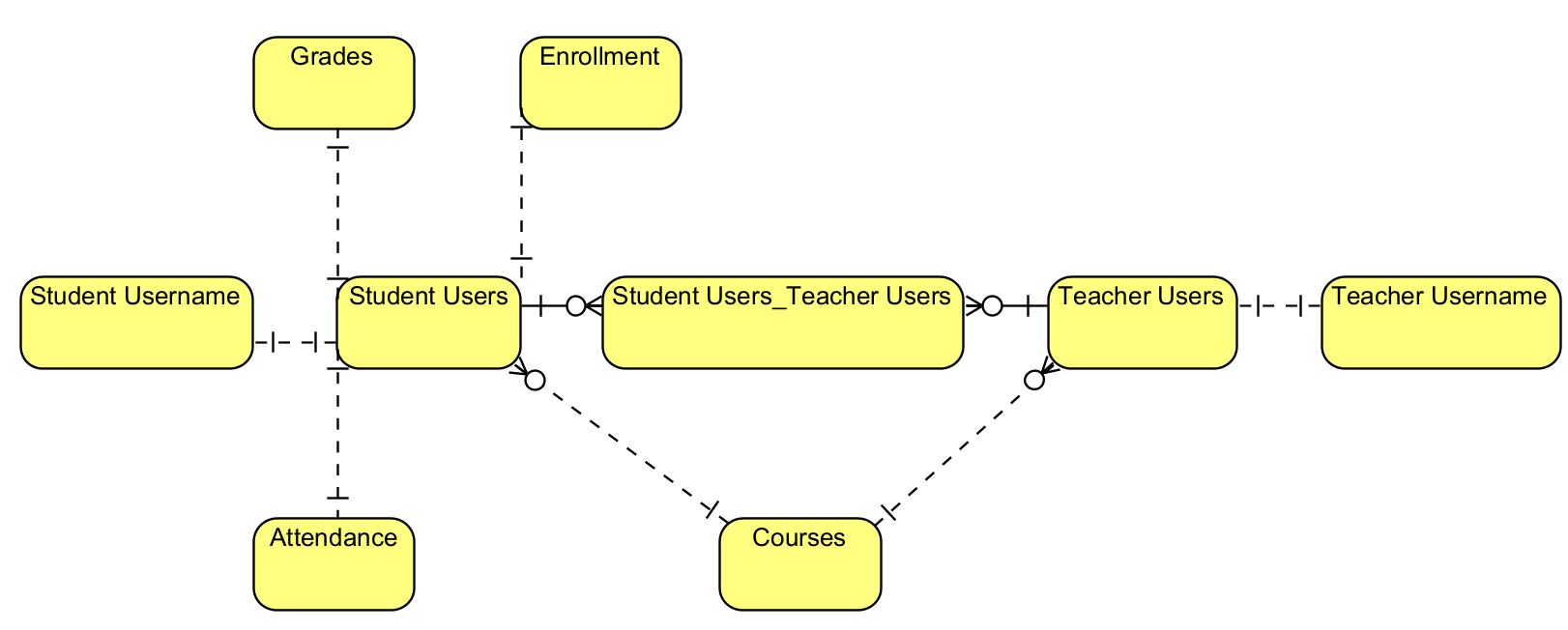


Logical ERD

A diagram of a computer program

Description automatically generated with medium confidence

Conceptual ERD



Second Normal Form: eliminates redundant data by requiring that each non-key attribute be dependant on the primary key. Each column must be related to the primary key.

# Appendix 5 – SAMPLES OF DDL/DML SQL STATEMENTS USED

SQL used to retrieve data from one table and then insert it into another as well as extra values taken from previous inputs:

A screen shot of a computer program

Description automatically generated

Succession of INSERT INTO commands in different tables:

A screen shot of a computer program

Description automatically generated

UPDATE statements:

A screen shot of a computer program

Description automatically generated

# Appendix 6 – Test logs

Test log in which we can clearly see an error, the error type and where it’s located:

A screen shot of a computer

Description automatically generated

# Appendix 7 – OUTPUT Screenshots

Main menu:

A screenshot of a computer program

Description automatically generated

Admin panel (with working authentication system):

A screen shot of a computer

Description automatically generated

Program exit:

A black screen with white text

Description automatically generated