David Wood

ISAD157 Coursework

Requirements Report

Contents

[1 Introduction 2](#_Toc38360399)

[2 Design 2](#_Toc38360400)

[2.1 Requirements 2](#_Toc38360401)

[2.2 UML Diagrams 2](#_Toc38360402)

[2.2.1 Entity Relationship Diagrams 2](#_Toc38360403)

[2.2.2 Class Diagrams 4](#_Toc38360404)

[2.2.3 Use Cases 4](#_Toc38360405)

[2.2.4 Activity Diagrams 5](#_Toc38360406)

[2.2.5 Sequence Diagrams 5](#_Toc38360407)

[2.2.5 State Diagrams 5](#_Toc38360408)

[2.3 SQL Statements 5](#_Toc38360409)

[2.3.1 Create the Tables 5](#_Toc38360410)

[2.3.2 Retrieve Data from Tables 5](#_Toc38360411)

[2.3.3 Add Data to Tables 5](#_Toc38360412)

[2.3.4 Delete Data from Tables 5](#_Toc38360413)

[2.4 List of High Level Functions 5](#_Toc38360414)

[2.5 Interface Designs 5](#_Toc38360415)

[2.5.1 Design 1 5](#_Toc38360416)

[2.5.2 Design 2 5](#_Toc38360417)

[2.5.3 Design 3 5](#_Toc38360418)

[2.5.4 Success Criteria for final design 5](#_Toc38360419)

[2.5.5 Final Design 5](#_Toc38360420)

[2.4 Link to Git Account 5](#_Toc38360421)

[3 Implementation 5](#_Toc38360422)

[3.1 Included Functions 5](#_Toc38360423)

[3.2 Interface Design 5](#_Toc38360424)

[3.3 Program Structure 5](#_Toc38360425)

[4 Evaluation 5](#_Toc38360426)

[5 Conclusion 5](#_Toc38360427)

[6 Appendix 5](#_Toc38360428)

# 1 Introduction

The following is documentation on the coursework assignment for the ISAD157 module. This will contain the requirements, design process and a description of the final implementation. It will also contain a link to the remote git repository that houses the files for the project.

If you want to understand the final implementation and don’t want to read through the full design process then skip to the implementation section (section 3, otherwise carry on reading.

Note that I will be referring to the system user as “System User” and a user that is stored in the database as “User”.

# 2 Design

## 2.1 Requirements

Using the coursework specification document I have identified that the system is required to do the following:

* Store user information
* Store friend connections between users
* Display information of a particular user

And allow the System User to do the following:

* Read information of a particular user
* Update information of a particular user
* Delete information of a particular user
* Delete friend connections of a particular user
* Create a user
* Create User friend connections

I have used the terms Create, Read, Update and Delete to relate to CRUD so it is easier to identify which functions require which SQL statements further down the design route. Also note that information relates to any data that has an identifier relating to the user in question – this is described later in the document.

## 2.2 UML Diagrams

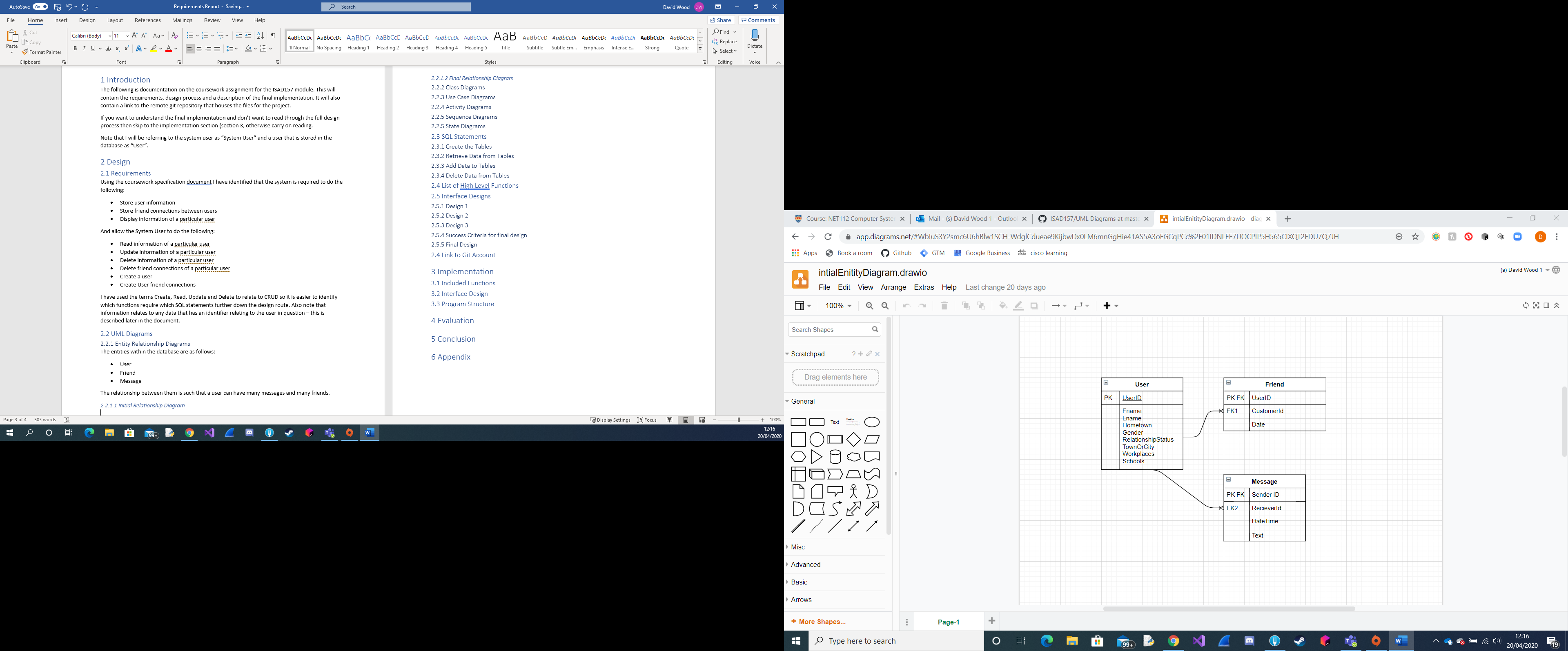
### 2.2.1 Entity Relationship Diagrams

The entities within the database are as follows:

* User
* Friend
* Message

#### 2.2.1.1 Initial Relationship Diagram

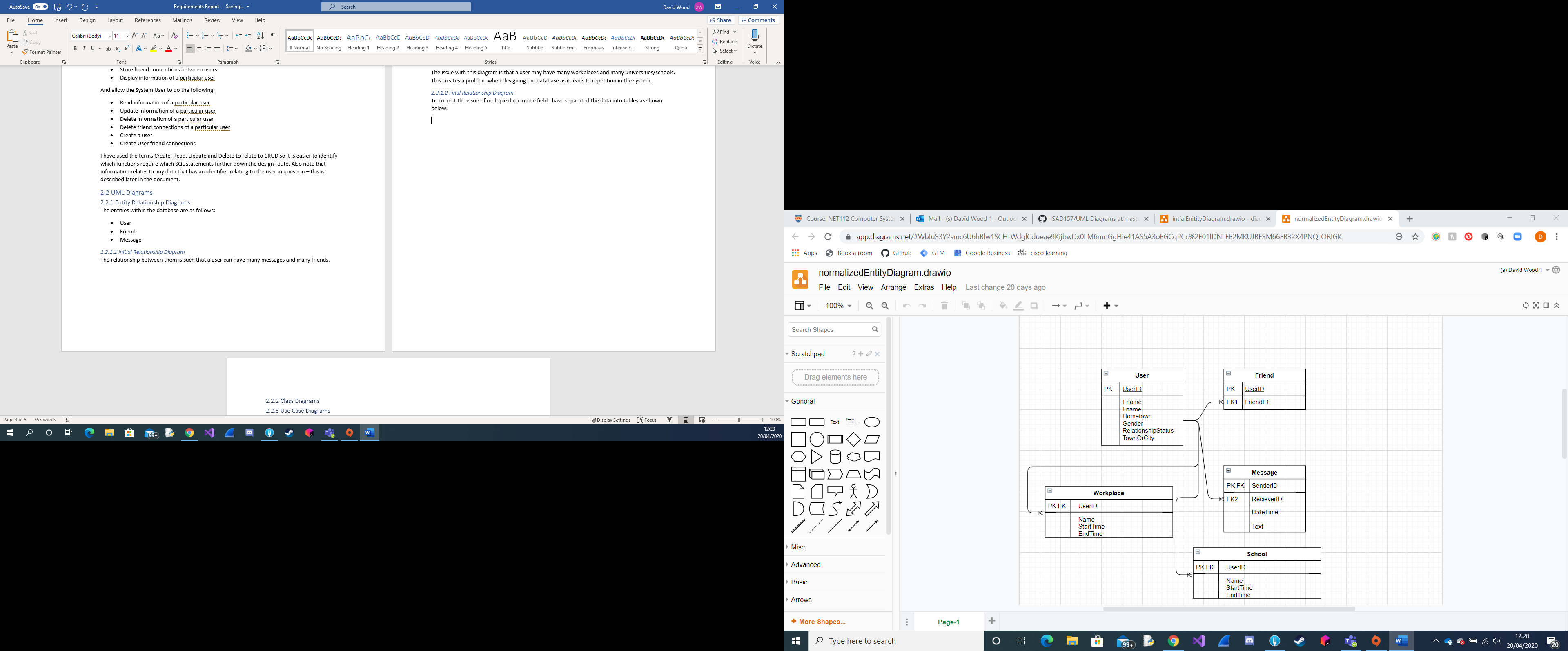
The relationship between them is such that a user can have many messages and many friends.



The issue with this diagram is that a user may have many workplaces and many universities/schools. This creates a problem when designing the database as it leads to repetition in the system.

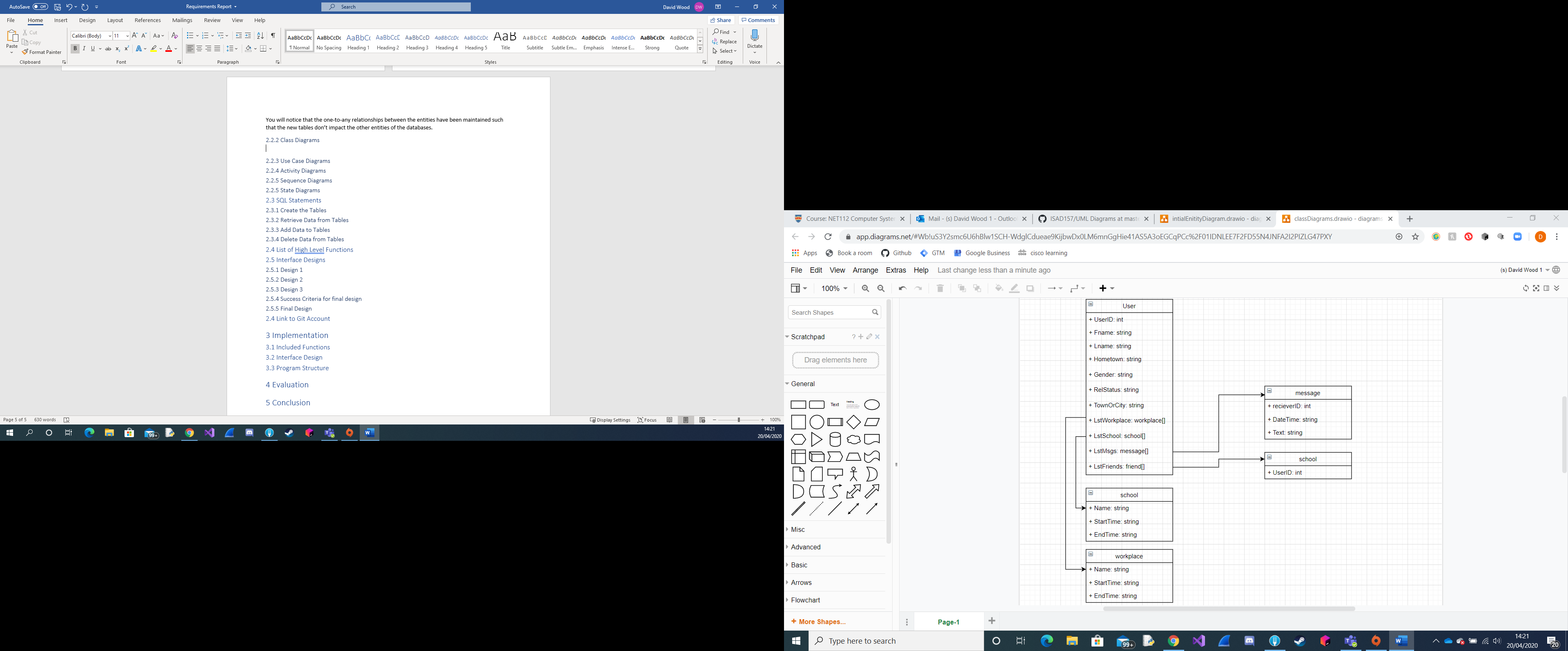
#### 2.2.1.2 Final Relationship Diagram

To correct the issue of multiple data in one field I have separated the data into tables as shown below.



This resolves the issue of having multiple data in one field and abstracts that to separate tables with the userID being used as the Primary Key as Foreign Key. This will help to find the data in the prototype implementation a lot easier than having to loop through and segment data in one field. You will notice that the one-to-any relationships between the entities have been maintained such that the new tables don’t impact the other entities of the databases.

### 2.2.2 Class Diagrams



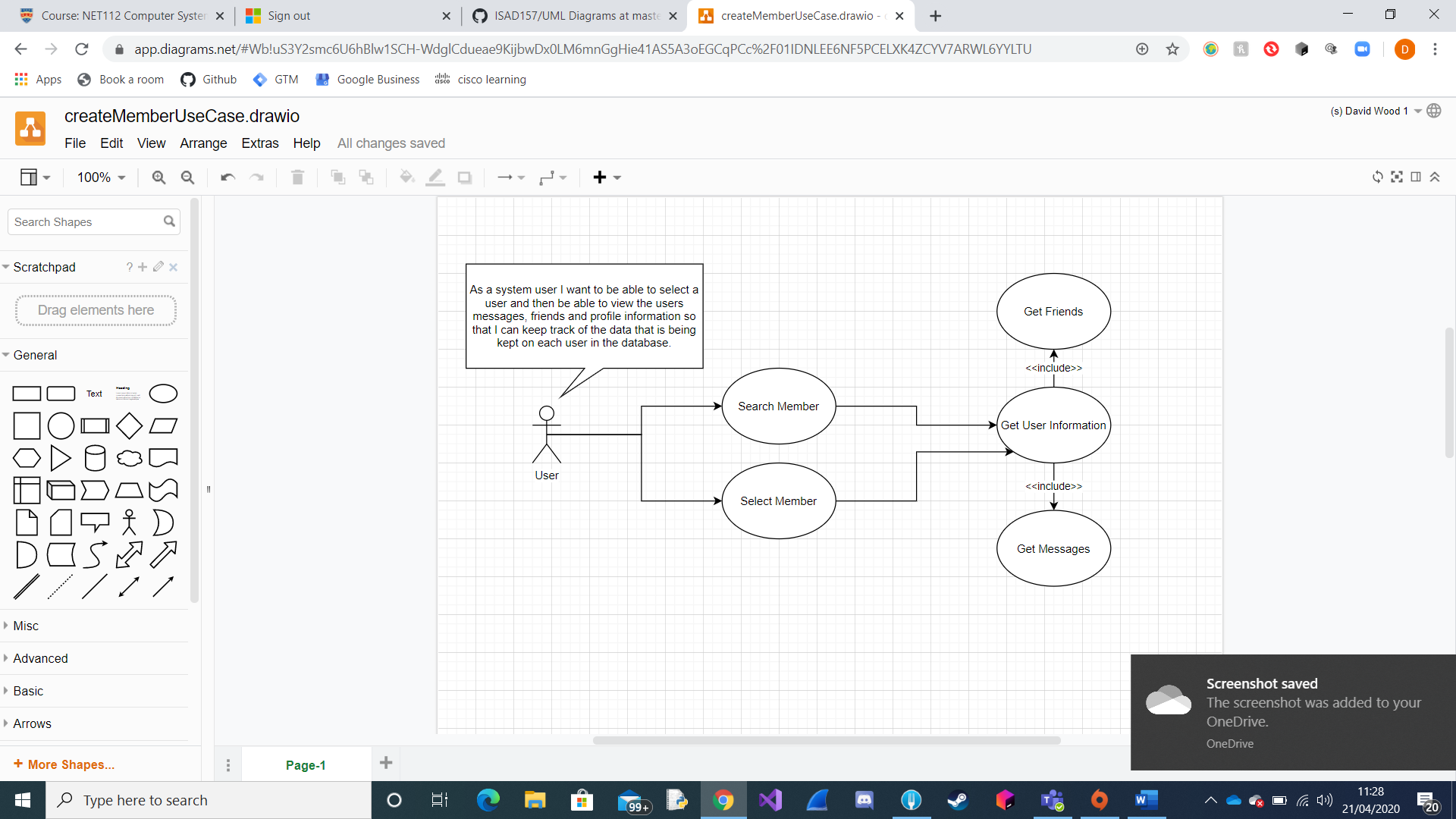
Above is an image displaying the class diagrams with the arrows dictating that the main class “User” is using the other classes as datatypes for certain attributes of “User” class. For example, the lstWorkplace attribute of User class is an array of object “workplace”. The arrow dictation along with the prefixes on the attribute names will allow me to build the class pseudocode more accurately than I would be able to if neither was included.

### 2.2.3 Use Cases

#### 2.2.3.1 Use Case Descriptions

As a system user I want to be able to select a user and then be able to view the users messages, friends and profile information so that I can keep track of the data that is being kept on each user in the database.

#### 2.2.3.2 Use Case Diagram



### 2.2.4 Activity Diagrams

### 2.2.5 Sequence Diagrams

### 2.2.5 State Diagrams

## 2.3 SQL Statements

### 2.3.1 Create the Tables

### 2.3.2 Retrieve Data from Tables

### 2.3.3 Add Data to Tables

### 2.3.4 Delete Data from Tables

## 2.4 List of High Level Functions

## 2.5 Interface Designs

### 2.5.1 Design 1

### 2.5.2 Design 2

### 2.5.3 Design 3

### 2.5.4 Success Criteria for final design

### 2.5.5 Final Design

## 2.4 Link to Git Account

# 3 Implementation

## 3.1 Included Functions

## 3.2 Interface Design

## 3.3 Program Structure

# 4 Evaluation

# 5 Conclusion

# 6 Appendix