Term 3 Submission - Manuela Spies

1) AS91905 - Use complex programming techniques to develop a computer program (6 credits)

1) a) “Use flask to create a web app” □

```python
from flask import Flask, render_template, request, redirect, session
import sqlite3
from sqlite3 import Error
from import_data import *
from import_messages import *
from flask_bcrypt import Bcrypt

DATABASE_NAME = "credit.db"

app = Flask(__name__)

flask_bcrypt = Bcrypt(app)

app.secret_key = "コレは秘密ㄇ. Jingle bells Käsekuchen. 4729371927"


def is_logged_in():
    # initialise_tables(create_connection(DATABASE_NAME))
```

1) b) “My app is complex: 2 types of data-variables, a condition (if), a loop (for/while), gets data from a form/database” □

2 types of variables: entry_cred = integer, entry_desc = text, message = boolean
& Gets data from a form or database

```python
@app.route('/new-standard')
def load_add_standard(message=False,
```

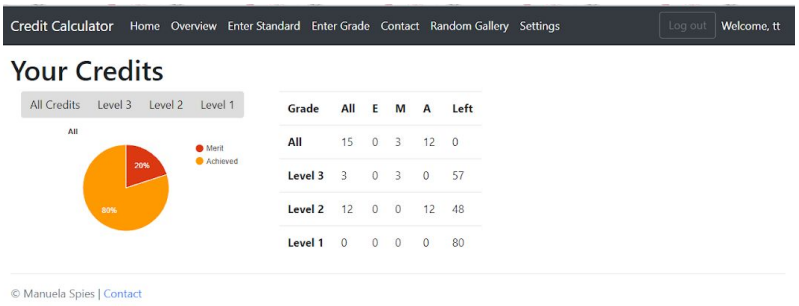
Condition (if) + loop (for)

```python
if result_standard < 1:
    print("ERROR: Some error with the standards on the tables.")
    return load_add_credits(data_used, "alert")

elif result_results >= 1:
    print("USER: Standard already entered.")
    return load_add_credits(data_used, "warning")

else:
    con = create_connection(DATABASE_NAME)
    entry_data = (entry_name, entry_grade, user_id)

    cur = con.cursor()
    cur.execute(new_credit_entry_query, entry_data)

    con.commit()
    con.close()

    return load_add_credits(success, "success")
```

```python
entry_as = request.form['standard_name']
entry_desc = request.form['description']
entry_cred = request.form['credits']
entry_lev = request.form['nzqa_level']
entry_read = request.form['reading']
entry_writ = request.form['writing']
entry_lit = request.form['literacy']
entry_num = request.form['numeracy']
entry_ue = request.form['ue_credits']
user_id = session['user_id']
```

```python
{% for row in standards %}
    <tr>
        <td>{{ row[2] }}</td>
        <td>{{ row[3] }} ({{ row[4] }} credits)</td>
        <td>{{ row[5] }}</td>
        <td>{{ row[6] }}</td>
        <td>{{ row[7] }}</td>
        <td>{{ row[8] }}</td>
        <td>{{ row[9] }}</td>
        <td>{{ row[10] }}</td>
        <td>{{ row[1] }}</td>
        <td>
            <a href="/delete-standard/{{ row[0] }}">Delete</a> -
            <a href="/edit-standard/{{ row[0] }}">Edit</a>
        </td>
    </tr>
{% endfor %}
```

1) c) “I’ve used 2 complex programming techniques (GUI, database, flask)” □

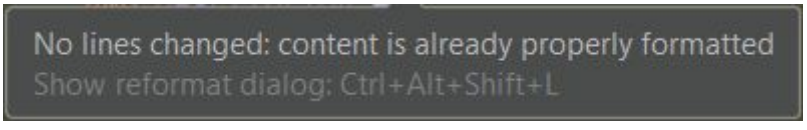



```
con.commit()
con.close()

con = create_connection(DATABASE_NAME)
cur = con.cursor()

cur.execute(count_rows_new_entry, (entry_name, user_id))
result_results = cur.fetchall()
result_results = result_results[0][0]

con.commit()
con.close()
```

```
from flask import Flask, render_template, request, redirect, session
import sqlite3
from sqlite3 import Error
from import_data import *
from import_messages import *
from flask_bcrypt import Bcrypt

DATABASE_NAME = "credit.db"

app = Flask(__name__)

flask_bcrypt = Bcrypt(app)
```

1) d) “My code is set out clearly” □

Result of ctrl + alt + L / PyCharm beautify:



1) e) “I have comments throughout my program” □

```
def create_connection(db_file):
    """create a connection to the sqlite db"""
```

```python
@app.route('/overview')
def overview():
    "renders overview including data displayed"

    if is_logged_in() == False:
        return redirect('/login')

    # LIST OF ALL COMPLETED STANDARDS
    con = create_connection(DATABASE_NAME)
    cur = con.cursor()

    cur.execute(get_all_done_standards, (session['user_id'],))
    standards = cur.fetchall()

    con.commit()
    con.close()

    # LIST OF ALL CREDITS BY GRADE / LEVEL
    credits_package = credits_numbers()

    # LEVEL ENDORSEMENT ["level", To E Endorsement, To M Endorse
    endorsement_data = [["l3", 50 - credits_package[1][2], 50 -
                        ["l2", 50 - credits_package[2][2], 50 -
                        ["l1", 50 - credits_package[3][2], 50 -

    for level in endorsement_data:
        if level[1] < 0:
            level[1] = 0
        if level[2] < 0:
            level[2] = 0

    # LIT, NUM, ...
    con = create_connection(DATABASE_NAME)
    cur = con.cursor()

    cur.execute(get_all_lit_num_things, (session['user_id'],))
```

1) f) “I’ve used sensible names”

They make sense to me.

I.e. variable names of the queries:

- create_table_result
- create_table_standard
- create_table_user

There is a common pattern (create_table) that denotes what the function does (creating a table), with the addition (_result, _standard, _user) that describes which table is created. This means that there is consistency and sensible variable names.

Routes & respective functions:

- /edit-standard/<id> & edit_standard()
- /update-standard/<id> & update_standard()
- /register & register()
- /new-standard & load_add_standard()
- /add-standard & do_add_standard()

Mostly, especially in newer functions, I tried to keep function names to be the equivalent of the route, but in older functions, occasionally, the name of the function is load_[whatever it does] and the ‘post’ version is either [whatever it does] or do_[whatever it does]. In the occasions where it’s different, it’s for the sake of making the coding/user experience easier:

/login is the path for login_page(). Having /login-page would be confusing for the user, but having login() would be confusing for the coding aspect.

1) g) “My variable and functions use python naming conventions” □

Yes. See code.

1) h) “My program deals with boundary cases & invalid input” □

See 2) f).

1) i) “My program uses constants where appropriate” □

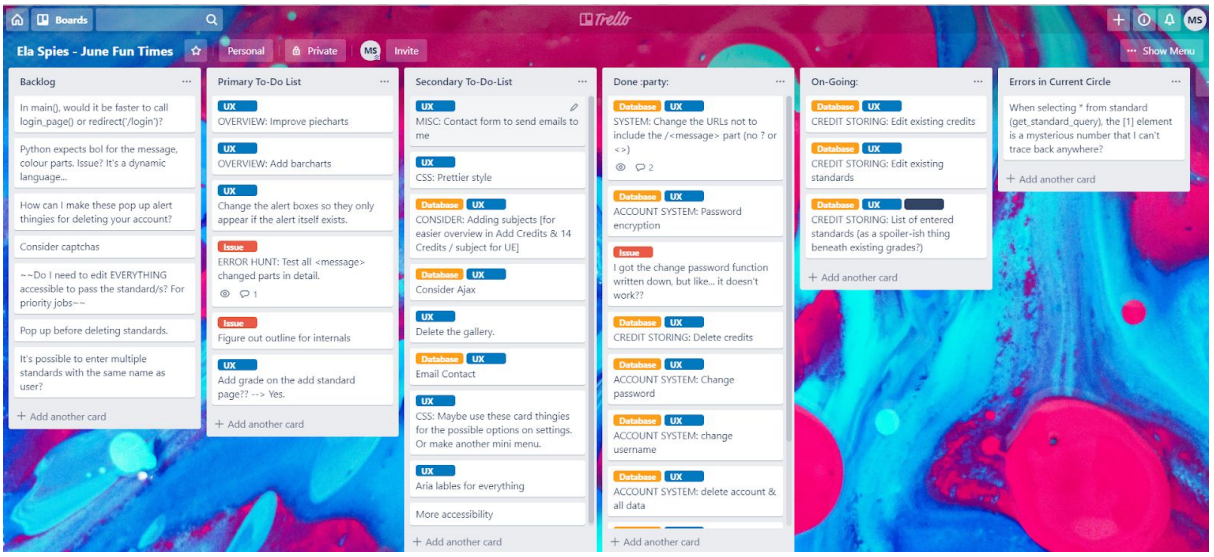
```
DATABASE_NAME = "credit.db"
```

2) AS91907 - Use complex processes to develop a digital technologies outcome (6 credits)

2) a) “Used agile methodology (scrum) to plan” □ish

https://docs.google.com/presentation/d/1GXNWdOPAUzhwVoWJeRDjaq17r5BCB5EISyCbKy9sv0g/edit?usp=sharing

2) b) “Used a kanban or scrum board (to do/doing/done) to plan □ish



2) c) “In my sprint documentation, I have evidence of trialling components” □ish

- Sprint 4
- Sprint 5
- Sprint 6

See link at 2) a)

2) d) “I have tested my website does what it is meant to do” □

See programming boundary documentation. Every possible input has been tested. Additionally, I added an error 404 function that prevents people from using non-existent URLs.

2) e) “Written an evaluation of how I’ve addressed some relevant implications” □

Accessibility

A relevant implication is accessibility. Accessibility is a term used to define how easily a place, event, thing or, in this case, website, is usable for anyone. It particularly focuses on

people with disabilities, such as i.e. blind people. This is very important in web design, as people who are not able to see struggle to use many websites that do not define things clearly for their usage / make their websites difficult to use for the screen readers these people use. As my project does not use sound, this is not relevant to my outcome, but if I used i.e. video or sound clips, providing subtitles would be essential.

For blind users, I have alt-tags on images describing them. This is important because people with visual disability, i.e. blind people use screen readers. The text within the alt tag describes the photos. Similarly, I used short but descriptive page titles as screen readers use them too, and they are also used for web searches and bookmarks. The text is highly readable.

For people struggling to differentiate colours, I tried to keep most things on a black-white scale, while adding a little blue for the sake of appearance. This, alongside the lack of fancy colours, should allow the users to view the website easily.

For people struggling to use website, I decided not to use a drop down menu for the selection of NCEA level / grade, but one that shows all available answers, since people might not realise that you can use drop down menus. This is for i.e. elderly people who don't know how to use the internet. Since the outcome could be used by parents to track their children's results, this is relevant. Otherwise, I also tried to model the design similar to parent portal, but in a more effective way (i.e. pie charts) to allow the transfer of pre-existing skills for such people.

By making sure that the outcome is usable on different devices (by using i.e. difference lengths through Chrome's inspect tool), I made sure that people who might not have a laptop available can access the website. This was a particular issue when my table was too large for a small (i.e. phone) screen, or the menu did not work perfectly fine. I also used an accessibility tester to check the website (WAVE).

Copyright

Another relevant implication is copyright, which is a legal implication. Copyright is the right that are given to the creator and/or owner of an original content upon creation. It gives right of ownership to things you created, such as books, or programs which applies to this case. Copyright by default is very "closed", i.e. the creator has many rights and nobody else has any rights, and if anyone owns, sells, modifies, or uses it can be punished through the law. However, the creator of the property can grant people license that allow them to use the property under given restrictions, i.e. time period, whenever they can edit it, whenever they can sell it or publish it, and costs for licenses. There are various types of pre-written licenses, and individual ones, under which these things are made available.

This is relevant to me. Thus, through licenses, I can use various written codes and similar things in my project. I am using bootstrap (licensed under the MIT License), Google Charts (under Apache license) and other things that don't need to be noted going by Mr Harford. The outcome only used open licenses, meaning licenses for which you do not need to pay and where the restrictions aren't strict. The licenses allow me to use the code on my project, due to the circumstances and limits/lack of limits.

Sqlite is public domain and thus usable, as nobody owns the copyright. Because of copyright implications, I used my own photos and obtained verbal consent of the models.

Privacy

Privacy is an important thing to be considered. I am storing data that some students may not want others to know about their grades. This is why you are getting logged out immediately if you put in the wrong password--even if it's just a typo. As well, the password is stored in a bcrypt encrypted form, meaning you can't just hack the database. I also tried to find any case where you could 'cheat' the application by i.e. putting a different ID into the url to access another standard.

2) f) "Written an evaluation discussing how this information led to the development of a high-quality digital technologies outcome"

How did you test? What methods did you use?

First of all, I listed all possible inputs the user would have. For each HTML document, I ran through the possible inputs (what I expected, boundary cases, no input) and tested whenever this would break the app and if the error appearing would be appropriate and if there was any issue with the app breaking, I would solve that problem.

I ran all of my web pages through a HTML validator (https://validator.w3.org/nu/#textarea).

Using the Wave editor to test accessibility. First, I ran through all pages to test whenever there were any red errors, and then I considered the yellow ones. On almost every page, I was alerted that I had two links next to another (to be exact; the Credit Calculator header & Home, as well as Username & Settings lead to the same URL), but I decided to maintain this, as being able to click on the title of a page and leading to the home page is a common occurrence online, and so is the name thing, but I wanted to maintain the 'home' and 'settings' part in the navigation for technologically illiterate people.

I had friends use the web page and give me feedback, which I listed in my backlog. I also did user testing, with verbal feedback.

What inputs did you have to test? What types of values did you enter?

Input	**Outcome / Error**	**Notes**
/new-standard		
Expected - 9991 - Standard Test 9991 - 1 - Level 1 - Yes - Yes - No	- "Your action was completed." - message. - Data correctly added (without grade) - Standard's grade isn't added.	- Fixed the standard-grade-issue.

- Yes - No - Not yet done		
Invalid - No input at all - Text input for AS Number and/or credits	- The app proceeded with action when everything BUT the grade was filled in. - There was an error despite the try-catch.	- I fixed the grade issue. - I noticed that the standard was entered either way, and by accident, that the app allows one user to have multiple standards with the same name and description. This isn't a fatal error but an issue, and I fixed it. - Try-Catch: - Solved. It was a syntax thing.
Boundary - AS Number: negative, above max int - Credits: negative, above max int	- User Error Message (ffor AS number & credits) as expected. -	Nothing to fix.
/new-achievement		
Expected - Selected an option in all fields	- Success message.	All fine.
Invalid - Nothing selected	- You can't go on without selecting a grade; automatically selects a standard due to drop down selection	All fine.
Boundary - Standard already added	- Notice to user that they already added the standard	All fine.
/settings -- change your password		
Expected - Correct password	- Success message & changed	All fine

& matching new passwords	passwords	
Invalid - No input	- Browser forces user to put in something.	All fine.
Boundary - Wrong password - Not fitting passwords	- User is notified and logged out. They can always log in again if it's actually their account. - Error message.	All fine
/settings -- change your username		
Boundary - Pre-existing username (someone else uses it) - Wrong password	- Warning & aborted action - User is logged out without change. They can log in again if it's their account.	All fine.
Expected - Correct password & new name that isn't already used	- Success message & changed username	All fine.
Invalid - No input	- Browser forces user to put something into the fields	All fine.
/settings -- delete all data		
Expected - Correct password & yes	- All data is deleted + success message.	All fine.
Invalid - No input	- Browser forces user to put something into the fields.	All fine.
Boundary - "No" - Wrong password	- Warning message without any action taken. - User is logged out. If it's their account, they can log in again.	All fine.

/settings -- delete all data		
Expected - Correct passwords, username & yes	- All data is deleted + logged out	All fine.
Invalid - No input	- Browser forces user to put something into the fields.	All fine.
Boundary - “No” - Wrong password - Wrong username	- Warning message without any action taken. - Message is given & user is prompted to try again. - User is logged out. If it’s their account, they can log in again.	All fine.
/login		
Expected - Existing user and correct password	- User is logged in and redirected to the next	All fine.
Invalid - No input	- Browser requires input.	All fine.
Boundary - Non-existent username - Password doesn’t match with the username	- In both cases, the same warning message (so it doesn’t indicate what was wrong, in case the person isn’t the user).	All fine
/register		
Valid - Non-existent username - Matching passwords	- Account is created & user logged in	All fine
Boundary - Passwords don’t align - Username already	- User is given an error message in both cases and the register page	All fine.

used	appears again.	
Invalid - No input	- The browser forces the user to enter something.	All fine.

What challenges did you have - give some examples of some annoying bugs you fixed?

I had a bug that affected how the passwords were entered. During the entering process, for some reason, I had the password stripped of all spaces and put into title format, but not during the login process. This made passwords that did not consist of numbers un-usable, and because I never noticed the line, it took quite a while to overcome.

For a while, I had trouble with the 'adding a grade or not' in the 'add-standard' form. This ultimatively boiled down to the fact that I build my database in a wrong way and required me to rewrite it. Fortunately, due to my sepperate SQL queries document, this didn't took too long.

Anothe error was the 'out of boundaries AS or credit integer'. I expected 'Except ValueError or TypeError:' to work, but it didn't and for some reason, putting them on different lines didn't occour to me for a while.

Another issue occured when I tried to prevent a user from accessing someone else's data through the /edit-standard/<id> url. The error message for the user did not pass on correctly, and this came down to me having used the wrong variable in the HTML file.

A lot of nasty problems boiled down to minor syntax errors or choosing the wrong item in a list. That being said, a 'Bad Request Error' became a little bit of my nemesis, as I constantly encountered it but also always forgot how to fix them and what caused them. From my memory, all were a little different, but still troublesome.

How did your testing improve the quality of the outcome?

Testing made sure that the user is not able to accidentally (or on purpose) active an error and crash the application. The accessibility testing allowed me, a person without disabilities, to check whenever the site could be used by people with disabilities.

Did you learn anything interesting about your app or web apps in general from your testing?

The testing helped me to make sure I wouldn't run into any problems. I learned a bit more about SQLite3 (for example, you use commas, not and when listing what you're updating) and discovered bcrypt.

Testing also made me re-evaluate the way my database built (originally, the results relied on the AS id, not the database ID of the standards which now sounds like a ridiculous idea) and helped me in understanding it better.

How did information from planning, testing and trialling of components assist in the

development of a high quality outcome?

For planning and testing, among other things I used personas, sketches and mock ups, as well as user testing and validators.

The results and comments from the user testing helped me to improve the website and make it easier to use for the end user. It made me realise that a design similar to Parent Portal (in visual terms, no bad pie charts, …) would actually be helpful as people would be able to apply previous habits. Similarly, using the competitor research, I was able to find things that I could do better than said competitors, such as multiple pie charts.

The process of using the personas made me realise that my third persona does not fit my product. However, the other two helped me to decide whenever I wanted to use a full list of all standards or let the users enter their own (I chose enter-your-own, based on the persona's desire to be able to use and decide about what is entered). I created these personas based on people I know and who are within my target group, which helped me to determine things as well.

The validator testing helped me in finding coding errors that could affect the usage of the product in older browsers and how easily the overview can be understood. Other than these two cases, it assured me that my code is good. Through validator testing, I was able to find an error on the /new-standard/enter page, and able to confirm that the coding of each of the other pages was correct. This allowed users of the website to have an easier overview of the possible choices for which NCEA level they can choose.

Making sketches and mock up versions on paper and online helped me with visualising how the website would end up looking like. The current version looks a bit different from my sketches, primarily because I've been focusing on the database aspects of the project and the coding, rather than the overview of grades. While I do aim to include more graphics, I placed a functioning website as more important. I've also added another page (that allows adding standards) between making the sketches/mockups and now.

I did not trial as much, this was because I had been thinking about the project for a while and had a relatively clear vision. I put the functionality of the website above appearance, which would have entailed most trialing. That being said, for the visual parts that I needed to decide (choosing buttons, …) using the process was helpful.

The testing and trialling aided me in considering the accessibility of the web app. By making sure that the outcome is usable on different devices (by using i.e. difference lengths through Chrome's inspect tool), I made sure that people who might not have a laptop available can access the website. This was a particular issue when my table was too large for a small (i.e. phone) screen, or the menu did not work perfectly fine.

Testing is very effective, as stated above, and also helps me with being confident in my code and app. Whenever writing a new piece of code, I was likely to just refresh the web app and check if it'd work; this was an automatic process. I generally tested the new additions to the app after I wrote the code. After I believed that a feature was fully working, I asked friends for their opinions and feedback, implemented that and went on. Doing this regularly was useful, as I didn't need to ask a friend to spend twenty minutes on testing a whole website, but also meant that I could think of a feature and/or component as 'done' and focus on the

next task. Doing a big test focusing on all the websites in the end was important though, as there was the possibility that one feature would oppose another.

How did this lead to a high quality outcome?

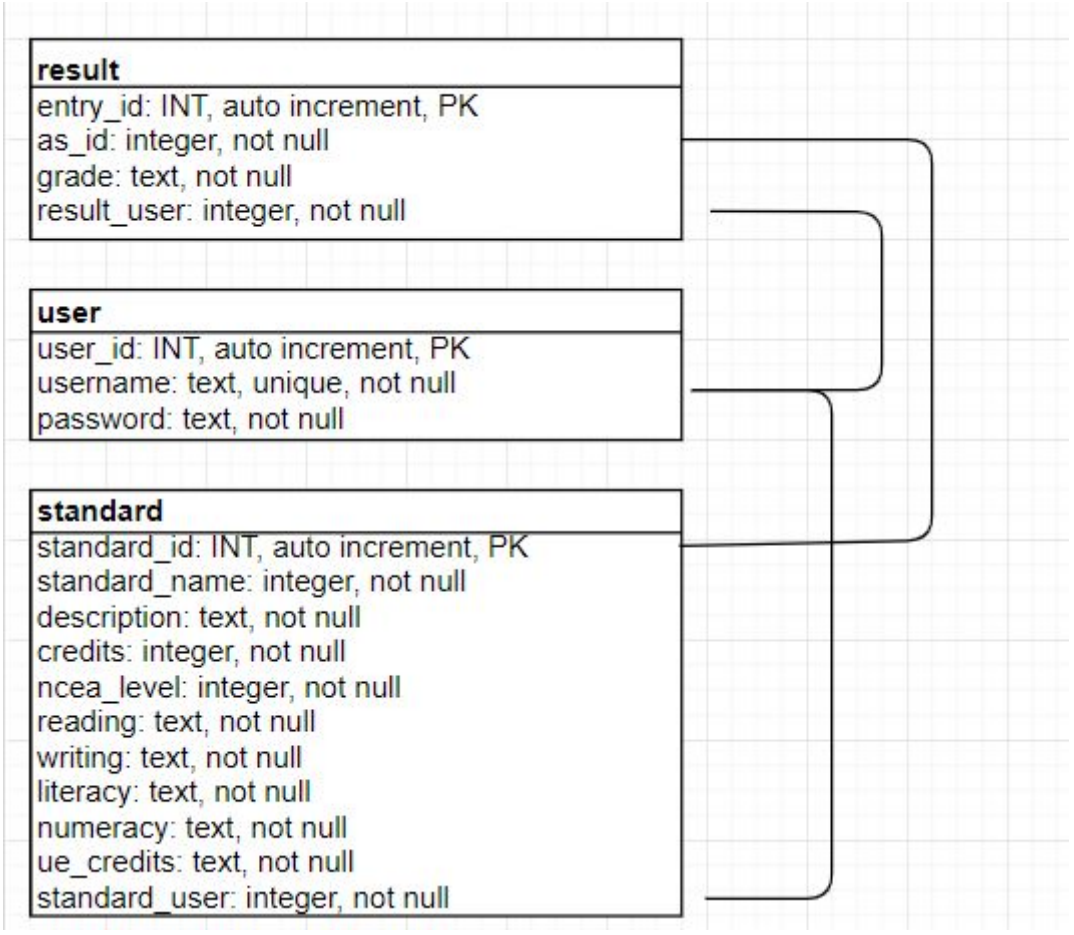
Having an outline, plan and sketches done prior to starting my code meant that I was able to produce code that was clear and organised. I've seen and heard of products having very messy, chaotic code, because features were added later on that were never intended and editing one affected the performance of a totally unrelated other due to that mess. My planning meant that I had a list of features I wanted to add and this mess was avoided.

The regular testing meant that I maintained feedback and when I did my big tests that I considered as 'milestones' which in return assured me that 'everything works right now'. Testing in itself is a must, because to ensure a good outcome, I would need to be able to guarantee that it worked. Testing was vital for that, and regular testing meant that I fixed various minor issues already.

3) AS91902 - Use complex techniques to develop a database (4 credits)

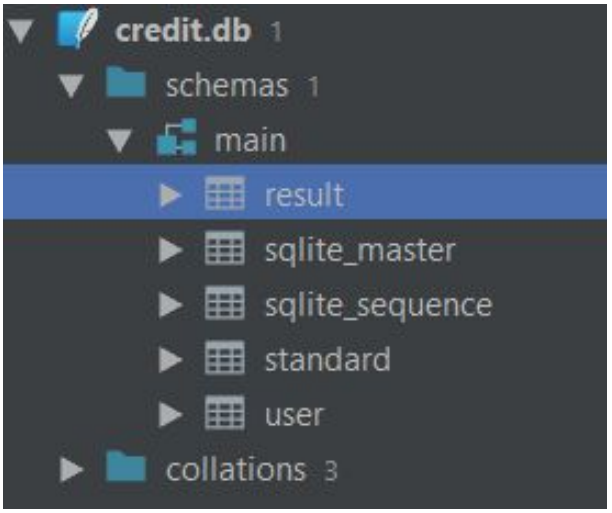
3) a) “A range of spreadsheet planning for my database; and/or A complete entity-relationship diagram (ERD)” □

result	entry_id	as_id	grade	result_user							
	integer	integer	text	integer							
result user references user.user_id	auto increment	not null	not null	not null							
as_id references standard.standard_	primary key										
standard	standard_id	standard_name	description	credits	ncea_level	reading	writing	literacy	numeracy	ue_credits	standard_user
	integer	integer	text	integer	integer	text	text	text	text	text	integer
standard_user references user_id	primary key	not null	not null	not null	not null	not null	not null	not null	not null	not null	not null
	auto increment										
user	user_id	username	password								
	integer	text	text								
	primary key	unique	not null								
	autoincrement	not null									



3) b) “Multiple tables in my database”

```
create_table_standard = """CREATE TABLE IF NOT EXISTS standard(
                        standard_id INTEGER PRIMARY KEY AUTOINCREMENT,
                        standard_name INTEGER NOT NULL,
                        description TEXT NOT NULL,
                        credits INTEGER NOT NULL,
                        ncea_level INTEGER NOT NULL,
                        reading TEXT NOT NULL,
                        writing TEXT NOT NULL,
                        literacy TEXT NOT NULL,
                        numeracy TEXT NOT NULL,
                        ue_credits TEXT NOT NULL,
                        standard_user INTEGER NOT NULL,
                        FOREIGN KEY(standard_user) REFERENCES user(user_id)
                        );
                        """

create_table_result = """CREATE TABLE IF NOT EXISTS result(
                        entry_id integer PRIMARY KEY AUTOINCREMENT,
                        as_id integer NOT NULL,
                        grade text NOT NULL,
                        result_user INTEGER NOT NULL,
                        FOREIGN KEY(result_user) REFERENCES user(user_id)
                        FOREIGN KEY(as_id) REFERENCES standard(standard_id)
                        );
                        """

create_table_user = """CREATE TABLE IF NOT EXISTS user(
                       user_id integer PRIMARY KEY AUTOINCREMENT,
                       username text UNIQUE NOT NULL,
                       password text NOT NULL
                       );"""
```



3) c) “Used a query to select data from the database”

```
# Get Credits queries
get_credits_all_query = """SELECT credits, grade
            FROM result JOIN standard
            ON as_id = standard_id
            WHERE result_user = ?;"""
get_credits_l3_query = """SELECT credits, grade
            FROM result JOIN standard
            ON as_id = standard_id
            AND ncea_level = 3
            AND result_user = ?;"""
get_credits_l2_query = """SELECT credits, grade
            FROM result JOIN standard
            ON as_id = standard_id
            AND ncea_level = 2
            AND result_user = ?;"""
get_credits_l1_query = """SELECT credits, grade
            FROM result JOIN standard
            ON as_id = standard_id
            AND ncea_level = 1
            AND result_user = ?;"""
```

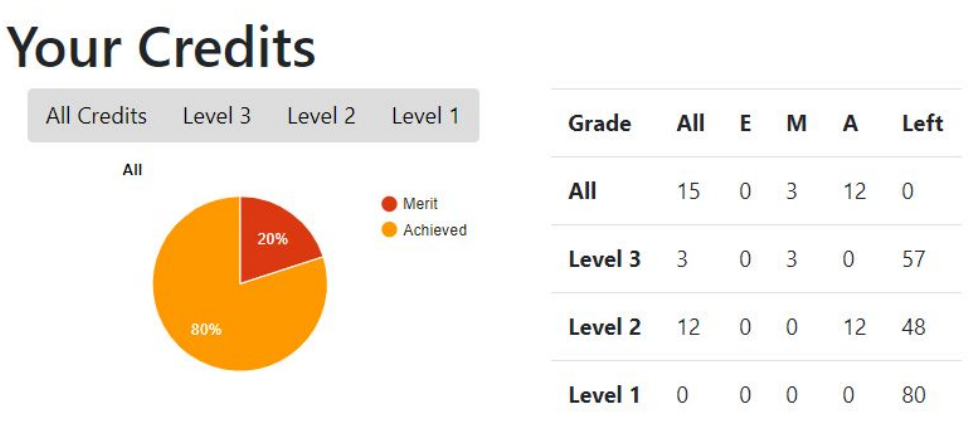
```
def get_credits(name, query):
    """counts and sorts all existing credits by grade."""
    con = create_connection(DATABASE_NAME)
    cur = con.cursor()

    cur.execute(query, (session['user_id'],))
    entries = cur.fetchall()

    con.commit()
    con.close()
```

```
all_credits = get_credits("All", get_credits_all_query)
l3_credits = get_credits("Level 3", get_credits_l3_query)
l2_credits = get_credits("Level 2", get_credits_l2_query)
l1_credits = get_credits("Level 1", get_credits_l1_query)
```

3) d) “Used a query to modify the database (insert/update/delete)” □

```
update_standard_query = """UPDATE standard
                    SET standard_name = ?
                    AND description = ?
                    AND credits = ?
                    AND ncea_level = ?
                    AND reading = ?
                    AND writing = ?
                    AND literacy = ?
                    AND numeracy = ?
                    AND ue_credits = ?
                    WHERE standard_id = ?
                    AND standard_user = ?;"""
update_grade_query = """UPDATE result
                    SET grade = ?
                    WHERE as_id = ?
                    AND result_user = ?;"""

# User Related Queries
create_user = """INSERT INTO user(user_id, username, password)
                VALUES (NULL,?,?);"""
find_user = """SELECT user_id, username, password
                FROM user
                WHERE username = ?;"""
select_all = """SELECT standard_name, description, credits, ncea_level FROM standard;"""

setting_change_password = """UPDATE user SET password = ? WHERE user_id = ?;"""
setting_change_username = """UPDATE user SET username = ? WHERE user_id = ?;"""

delete_account_result = """DELETE FROM result WHERE result_user = ?;"""
delete_account_standard = """DELETE FROM standard WHERE standard_user = ?;"""
delete_account_user = """DELETE FROM user WHERE user_id = ?;"""
```

3) e) “Presented some data from my database on a webpage” □



Grade	All	E	M	A	Left
All	15	0	3	12	0
Level 3	3	0	3	0	57
Level 2	12	0	0	12	48
Level 1	0	0	0	0	80

3) f) “Written an evaluation of how I’ve addressed some relevant implications”□

See 2) e)

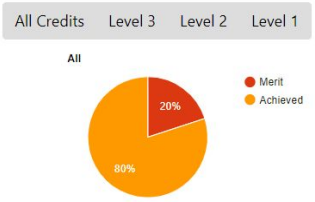
3) g) “Refactored my database to normal form” □

Mr Harford said it’s fine.

3) h) “Completed the sprint documentation (with at least 3 sprints). This should show some planning, trialling and testing in each sprint.” □

- Sprint 4
- Sprint 6
- Sprint 7

3) i) “Presented the data effectively for the purpose and end-users of my site” □



Grade	All	E	M	A	Left
All	15	0	3	12	0
Level 3	3	0	3	0	57
Level 2	12	0	0	12	48
Level 1	0	0	0	0	80