The OASIS (Team 6)
Date: May 8th, 2019

Module Delivery Plans: Assessment Data Collection & Visualization
Team Report

1. Our Team:

Josh Hare (Software Lead)
Emily Lau (Project Manager)
Ahmad Bilal
Chang Luo
Hongyu Tang
Ting Zhang

2. Introduction & Background:

Module delivery is an important logistical process for an academic department to organize how assessments are administered to students. Having an efficient module delivery plan is vital to making sure students receive assessments on a well-paced schedule and have the most fulfilling learning experience possible.

The Department of Computer Science's present module delivery method requires a good deal of manual effort. Every year, Learning and Teaching Managers (LTMs) send a Google Form to module leaders, who populate this form with information on each of the assessments they plan to administer. This information is manually validated by academic tutors, who check that assessment dates are correct and that they add up to 100% within their module. They must also check that there are not too many assessments due in one particular period of time, so students do not become overwhelmed. Afterwards, tutors manually compile the information into a module booklet.

This system is user-unfriendly and inefficient, since it relies on the academics' initiative to complete the Google Form with the correct information, as well as the academic tutors' painstaking effort to validate and visualize all information.

3. Project Scope and Objectives:

The objective of our project was to create a user-friendly system that automates and streamlines the tasks described above, so that academic tutors can collect and visualize assessment information with ease. Our system needed to automatically validate the assessment information entered by academics, from the handout/handin dates to their percentages. We had to enable users to visualize the intensity and timing of assessments using heatmap and timeline features. Our system also needed to automatically generate a booklet page from each module's assessment information. Finally, the database of our system had to be stored appropriately, such that each instance of the software opened on different devices would reflect the same data.

The OASIS (Team 6)
Date: May 8th, 2019

4. Process:

4.1. Sprint 1 & 2:

Back End Code:

Josh, Emily and Ahmad were in charge of writing the back end functions. At these initial stages we decided to use the flat CSV file as the database. Then the dataset was organized into “module” and “assessment” classes, and different functions needed for each class were divided among the team. Ahmad wrote the check_date function, Emily wrote the check_percentage function and print_assessment_list function, and Josh wrote the rest. These functions were written to validate the information provided on the CSV.

Interface Design:

At the beginning of building an application, a prototype is essential for later implementation. Initially, Ting and Hongyu designed the prototype using Axure RP8 based on the application requirement which can be viewed in Figure 1 below. As it shows, a prototype is a good reminder for function added to the button in later coding.

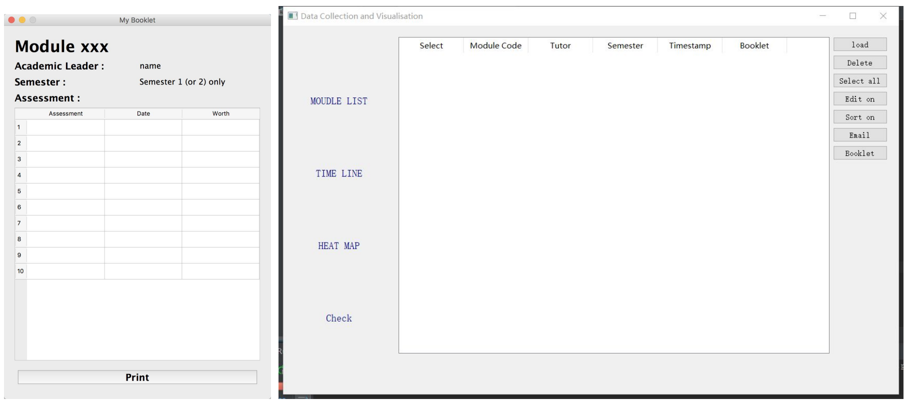


Figure1. Original interface prototypes for booklet and main page

After reading the requirement of the instruction, Python was decided as the programming language which means related GUI packages should be used in the project. Chang, Ting and Hongyu discussed with each other and agreed to use PyQt5 as our GUI library. It took some time for them to study it firstly and then completed the fundamental framework based on the designed prototype. Specific functions were added during the Sprint 3 as our fundamental framework only contains the require buttons and interface.

4.2. Sprint 3:

This sprint focused on bringing the front and back end together and make the interface more aesthetically pleasing.

There are four main pages in our interface. Josh and Chang wrote the most of functions on module list page and check page. Josh is also responsible for the heatmap page, and Emily is responsible for the timeline page, and connected the data to widgets on those pages. Josh also wrote function for the load button to load the data from back end and wrote the function for the update button on the check page to show the error information on QTextEdit widget. Chang wrote functions for delete, select all, edit and sort button on the module list page to make data editable and wrote the email function on the check page. Josh and Chang also worked together to finish the print button on the booklet window. Josh converted the content of booklet to HTML file. Chang converted the HTML file to PDF.

Our later work was mainly about improving the beauty of the interface to enable the project to look more user friendly. It includes adding QSS to change font, replacing buttons with icons and layout size adaptively for the beauty of the interface.

- QSS (Qt Style Sheets) is similar to CSS. It controls the function of change the font-size, font-style and background color. Most of the style settings can be stored in QSS which ensures coding readability.
- In addition, we also used some icons in navigation bar instead of text as clickable buttons. Icons seem more lively and user friendly when they were put in the navigation bar.
- Our table was not well laid out in the beginning, as the content was displayed at the left and a large blank was present at the right of the window. In order for the content of each cell to fill the whole column, we found a solution online to make the content to adapt the width of the column.

To safeguard access to the dataset, Ahmad created the login functions and encrypted the password with salt hash lib. If a user is new then he needs an account which can be provided by the only admin. As for registering a new user, the security code is required which is only known to the admin. Moving on to the main window of the project, User can load, edit, add, delete and print the data in the form of a booklet. One of the major tasks was loading the data from a CSV file and storing them in a locally created database was done by Josh. This way, the user is making amendments in the database instead of a CSV file.

5. User Stories:

During each sprint we agreed on a set of features (in the form of user stories) that we wished to accomplish. Below are the user stories we set for each sprint.

Sprint 1:

- As a Academic Tutor, I want to check if the assessments in a module add up to 100%, so that the students grades are calculated correctly
- As a academic tutor, I want to check the assessment dates are within the semester set for the module, so that students are able to complete assessments on time
- As a academic tutor, I want to check the assessment dates are not on weekends and public holidays, so that students can maintain a good work-life balance

Sprint 2:

- As a course administrator, I want to be able to view a timeline plot and heatmap of module assessments, so that I can check if the modules having too many clashing deadlines

- As a LTM, I want to filter the visualisations by level, so that I can check if the course module assessments have too many clashing deadlines
- As a course administrator, I want to make sure exams are only set during the designated exam period

Sprint 3:

- As a user, I want a login and password protection, so that only verified colleagues can alter the data
- As a member of academic staff, I want to be able to enter information on my module's assessments, so that academic tutors will be able to account for my module
- As an LTM, I want to be able to add new login usernames and passwords, so that I can control who has access to the software
- As a administrator, I want to be able to save the database onto a remote server, so that users have an up to date copy of the data
- As a administrative assistant, I would want to automatically produce a A4 booklet of the course modules, so that I can handout a copy of the content without a member of staff having access to the software

6. Analysis & Design:

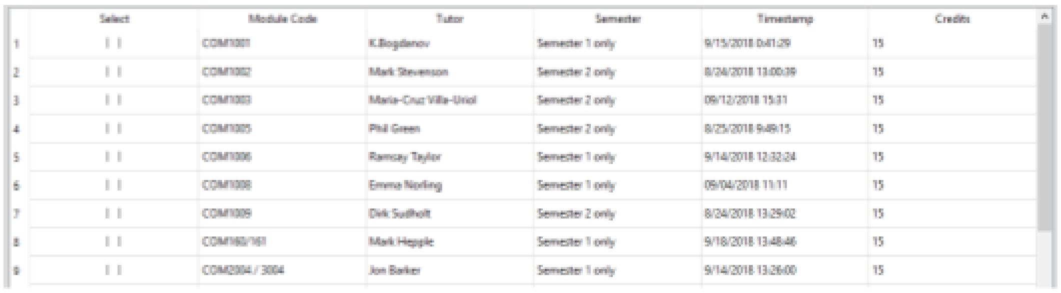
Python was used to develop this program, as it was one of the preferred languages specified by the client. It also has many workable plotting tools that the team were familiar with from previous work. This choice was preferred over the initial choice of Java, since the team decided it would be easier to learn a new graphical user interface (GUI) tool PyQt (a Python wrapper for the C++ library Qt) than to implement custom graph or use a Java graphical package with a small user base, since there would be less support for common troubleshooting queries. In addition, as previously stated, one of the most important features to the client was a good visualisation of the data via graphs and plots, and the team decided this outweighed the importance of a more stylish GUI.

Rather than use one table to visualise the data, a dynamic approach was used, such that a single table contained the core module information, such as the academic teacher, with a feature to view a separate table for each of the modules which shows the assessment data. as shown in Figure.1

To this end, an OOP approach was taken to model the dynamic nature of the number of assessments in a module. In this approach two separate classes were used: 1) A module class which contained module name, teacher, number of credits, semester, a timestamp and a list of class assessment and 2) assessment class which contained a handout date, handin date, assessment type, name and percentage.
The module class contained methods which validate the data inputted into the class.

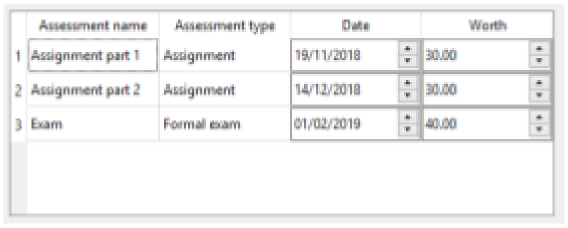
- A method to check that the sum of assessment percentages was equal to 100%.
- A method to check whether the validity of the inputted handin date which comprised of
    - A method to check whether the handin date was on a weekend or public holiday
    - A method to check whether the handin date was after the handout date
    - A method to check whether the handin date was within the academic year

Figure.1



	Select	Module Code	Tutor	Semester	Timestamp	Credits
1	☐	COM1001	K.Bogdanov	Semester 1 only	9/15/2018 0:41:29	15
2	☐	COM1002	Mark Stevenson	Semester 2 only	8/24/2018 13:00:39	15
3	☐	COM1003	Maria-Cruz Villa-Uriol	Semester 2 only	09/12/2018 15:31	15
4	☐	COM1005	Phil Green	Semester 2 only	8/25/2018 9:49:15	15
5	☐	COM1006	Ramsay Taylor	Semester 1 only	9/14/2018 12:32:24	15
6	☐	COM1008	Emma Norling	Semester 1 only	09/04/2018 11:11	15
7	☐	COM1009	Dirk Sudholt	Semester 2 only	8/24/2018 13:29:02	15
8	☐	COM160/161	Mark Hepple	Semester 1 only	9/18/2018 13:48:46	15
9	☐	COM2004 / 3004	Jon Barker	Semester 1 only	9/14/2018 13:26:00	15

(a) A table showing the module information



	Assessment name	Assessment type	Date	Worth
1	Assignment part 1	Assignment	19/11/2018	30.00
2	Assignment part 2	Assignment	14/12/2018	30.00
3	Exam	Formal exam	01/02/2019	40.00

(b) A table showing assessment information for a specific module

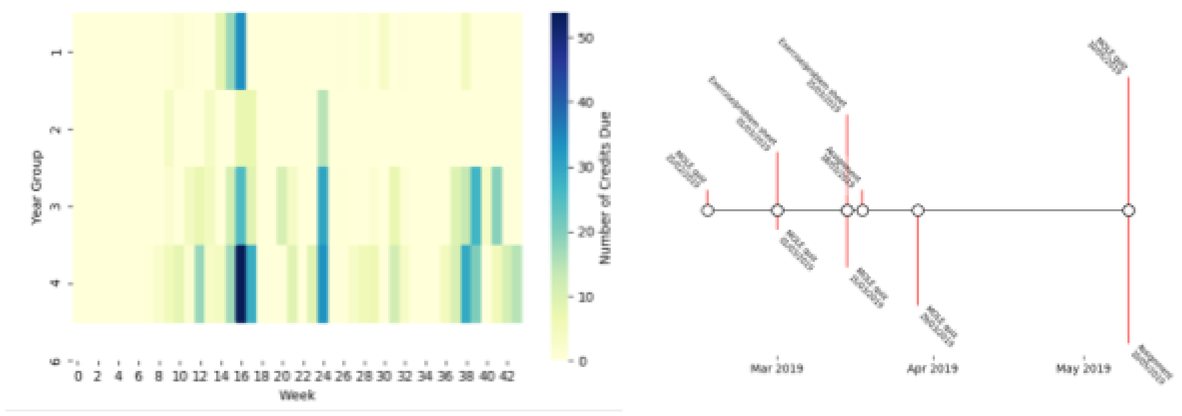
Initially an algorithm extracted the class attributes from a singular CSV file containing the data information. However, rewriting the data back to the CSV felt too inconvenient and would be prone to too many errors, the team decided to save and store the data in a custom database.

In order to effectively store and navigate the saved data, each module was saved in a separate file. In Python, this was a simple task of "pickling" the module class object, which "dumps" the class object into a binary file. This allowed for an across platform method of saving information efficiently, as binary files are more memory efficient and often faster to load than text files. Using multiple saved files rather than a singular file would also reduce the damage done if a saved file is corrupted, as most of the information is still accessible, so for a LTM to get academic staff to re-enter a single module's information is not a large task compared to having to get academic staff to re-enter the entire database.

For the user interface, a simple GUI was designed with PyQt5, on the left hand is a side-panel which navigates the user between separate panels within the main window. These include the main panel which shows the module tabular data as seen in Figure 1(a), a separate panel for each of the data visualisation tools requested by the client (Timeline and Heatmap to visualise to quantity of work due over a given time period), and a error checking panel where the user can check for data entry errors for the modules and email this file to a user defined email address.

For the heatmap and timeline, the user can specify which or all of the year groups along with which semester (1, 2, or both) should be visualised. This can be seen in Figure 2.

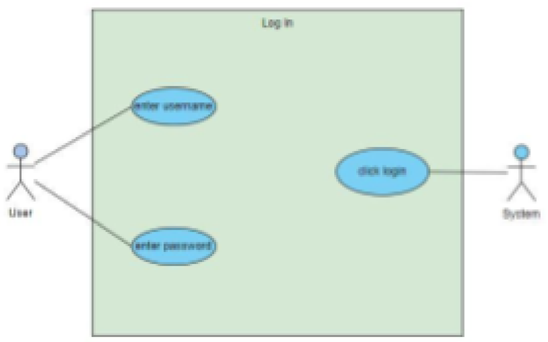
Figure.2



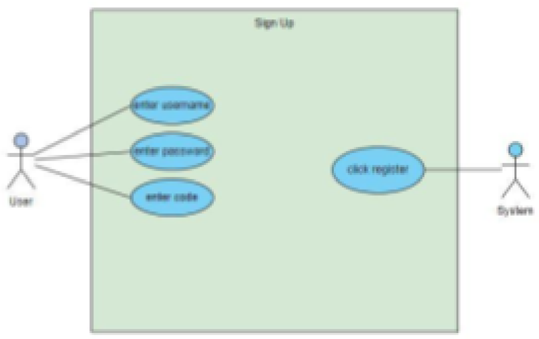
(a) Heatmap visualisation of assessments due (b) Timeline visualisation of assessments due

7. Test Plan:

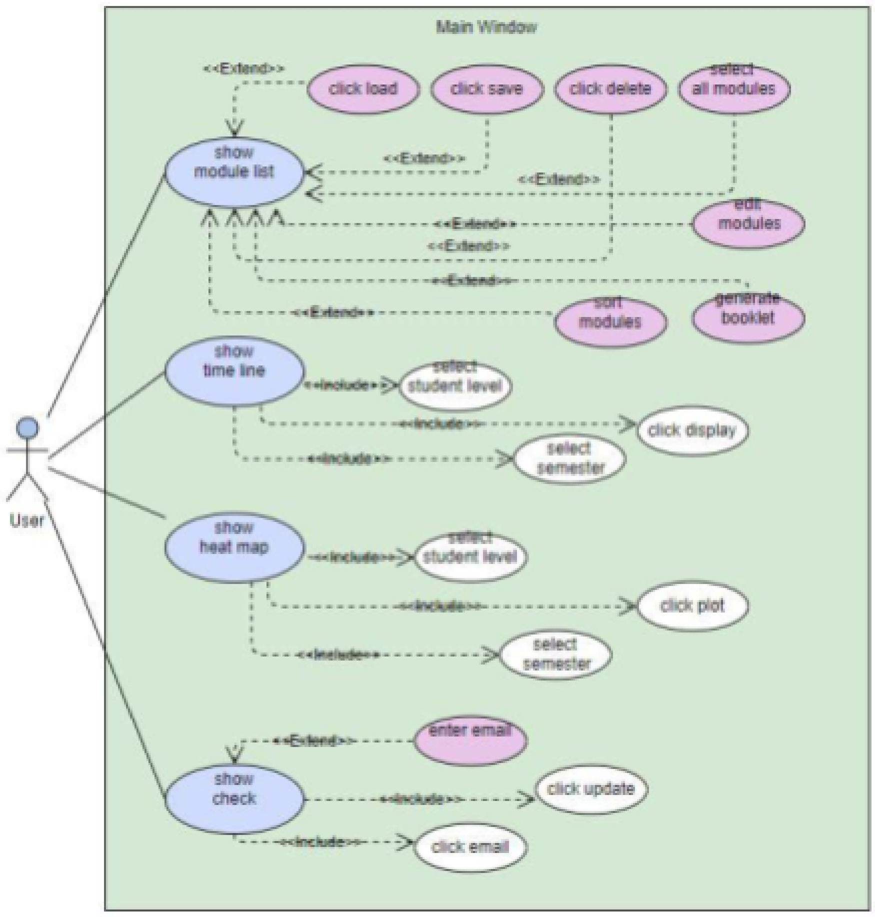
Below are the test plans for the login/registration, main window, and booklet pages. Following the actions on the "user" side of the diagram should result in the "system" actions on the other side.



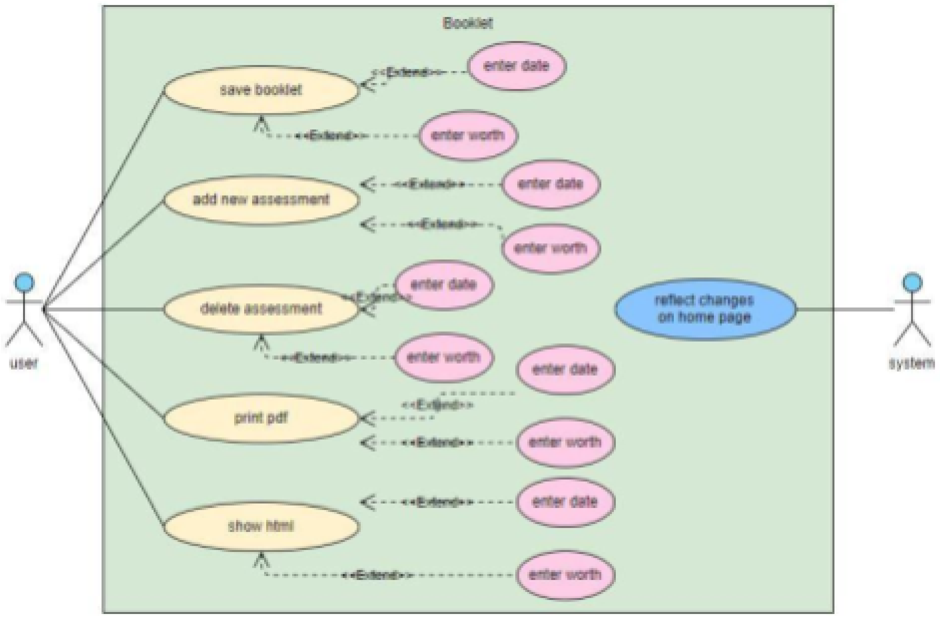
(a) use case for log In



(b) use case for sign up



(c) use case for Main Window



(d) use case for Booklet

8. Team Management & Communication:

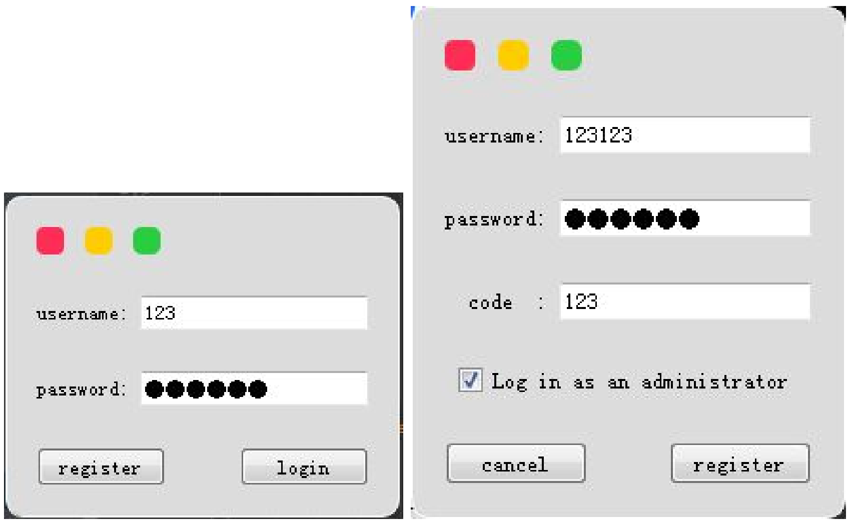
Our team communicated through a Facebook group chat and met every Monday from 2-5 pm, which allowed us to communicate face-to-face during the work process. Emily took meeting notes to document every team member's work. There was a Trello made for the group in order to list and delegate tasks.

Josh was named Software Team Lead and was responsible for much of the architecture of the project. Emily was Project Manager and kept track of tasks to be done, walk through ideas for features with the team, and organized the documentation at the end of each sprint.

Initially, Chang, Hongyu, and Ting were primarily concerned with the front end, while Ahmad, Josh, and Emily were primarily concerned with the back end. When joining front end and back end, we assessed the tasks left and committed to the ones we wanted to complete. Strictly defining the scope of each task during this part of the process was very important, as tasks were sometimes similar overlapped, and team members occasionally completed other team members' tasks by accident.

9. Completed Features:

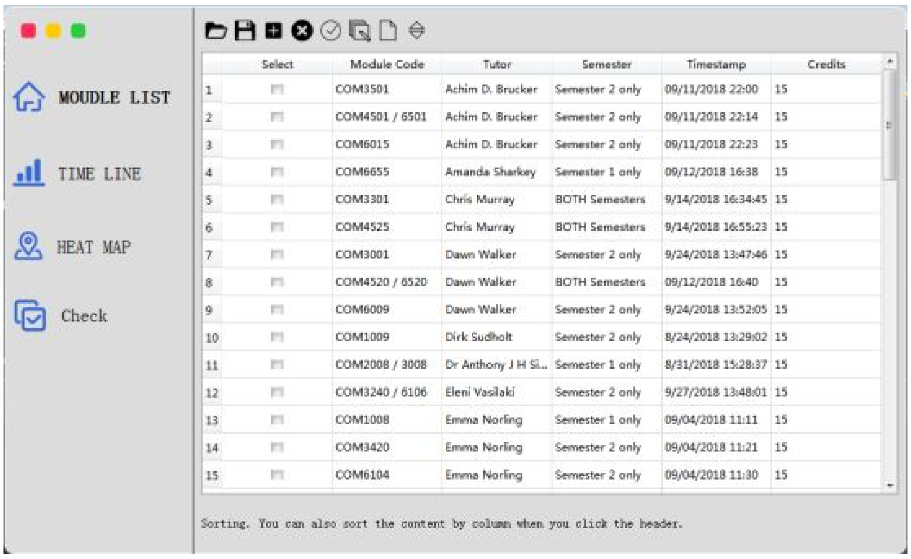
- **Login & Registration**: Allows users to login, or register if they are new. New users can register with a code.


username: 123
password: ●●●●●●

username: 123123
password: ●●●●●●
code : 123
[✓] Log in as an administrator

Figure 1. Login and Register

- **Editable Module List**: User can modify subject information and save it to documents. Users can select multiple modules to view their booklets.

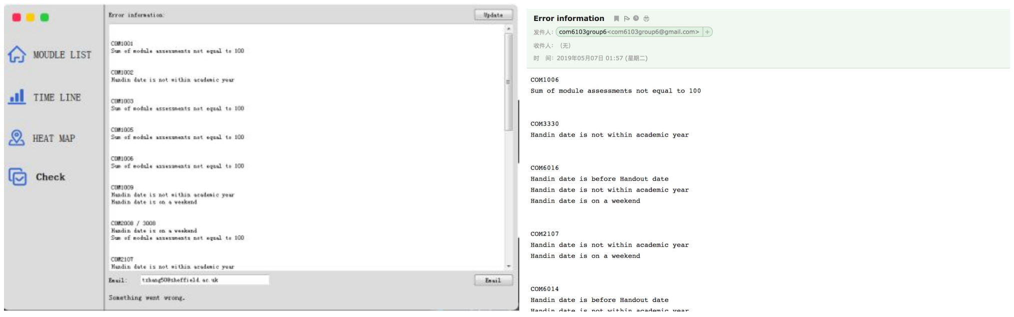


	Select	Module Code	Tutor	Semester	Timestamp	Credits
1		COM3501	Achim D. Brucker	Semester 2 only	09/11/2018 22:00	15
2		COM4501 / 6501	Achim D. Brucker	Semester 2 only	09/11/2018 22:14	15
3		COM6015	Achim D. Brucker	Semester 2 only	09/11/2018 22:23	15
4		COM6655	Amanda Sharkey	Semester 1 only	09/12/2018 16:38	15
5		COM3301	Chris Murray	BOTH Semesters	9/14/2018 16:34:45	15
6		COM4525	Chris Murray	BOTH Semesters	9/14/2018 16:55:23	15
7		COM3001	Dawn Walker	Semester 2 only	9/24/2018 13:47:46	15
8		COM4520 / 6520	Dawn Walker	BOTH Semesters	09/12/2018 16:40	15
9		COM6009	Dawn Walker	Semester 2 only	9/24/2018 13:52:05	15
10		COM1009	Dirk Sudholt	Semester 2 only	8/24/2018 13:29:02	15
11		COM2008 / 3008	Dr Anthony J H Si...	Semester 1 only	8/31/2018 15:28:37	15
12		COM3240 / 6106	Eleni Vasilaki	Semester 2 only	9/27/2018 13:48:01	15
13		COM1008	Emma Norling	Semester 1 only	09/04/2018 11:11	15
14		COM3420	Emma Norling	Semester 2 only	09/04/2018 11:21	15
15		COM6104	Emma Norling	Semester 2 only	09/04/2018 11:30	15

Sorting. You can also sort the content by column when you click the header.

Figure 2. Load Module List

- **Timeline**: Generates a timeline based on the data. Software can filter the data by year group and semester and display the timeline accordingly. Users can view the time distribution of assessments. There are buttons to adjust the size of the plot, enlarge and view the parts, or save them.
- **Heat Map**: Generate the heat map based on the data. Software can filter the data by year group and semester and display the heat map accordingly. Users can visually observe the density distribution of assessments. There are buttons to adjust the size, enlarge and view the parts of the plot, or save them.
- **Check Error**: Automatically check whether the sum of all assessments for each module is 100% and whether handin date is correct. Shows user the modules with erroneous information
- **Emailing Users** : Error messages can be sent to user-specified mailboxes.



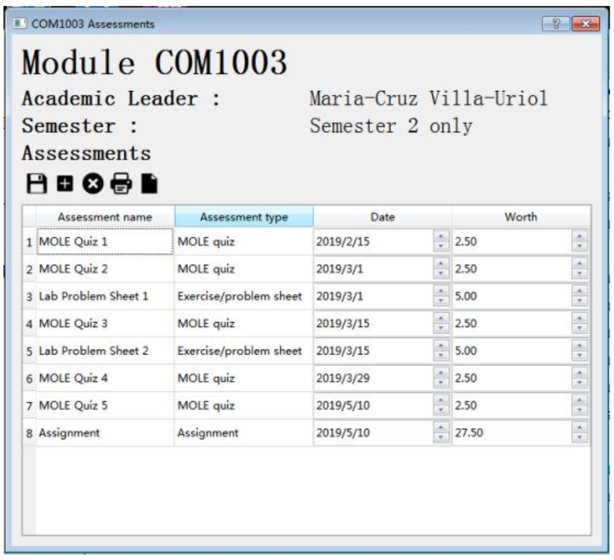
```
Error information:

COM1001
Sum of module assessments not equal to 100

COM1002
Handin date is not within academic year

COM1003
Sum of module assessments not equal to 100

COM1005
Sum of module assessments not equal to 100

COM1006
Sum of module assessments not equal to 100

COM1009
Handin date is not within academic year
Handin date is on a weekend

COM2008 / 3008
Handin date is on a weekend
Sum of module assessments not equal to 100

COM2107
Handin date is not within academic year

Email:  tzhang50@sheffield.ac.uk

Something went wrong.
```

Error information
发件人: com6103group6<com6103group6@gmail.com>
收件人: (无)
时　间: 2019年05月07日 01:57 (星期二)

```
COM1006
Sum of module assessments not equal to 100

COM3330
Handin date is not within academic year

COM6016
Handin date is before Handout date
Handin date is not within academic year
Handin date is on a weekend

COM2107
Handin date is not within academic year
Handin date is on a weekend

COM6014
Handin date is before Handout date
Handin date is not within academic year
```

Figure 3. Check Error and Email received

- **Booklet** : Details of relevant modules can be displayed, and relevant information can be modified. The content can be shown as a PDF file or displayed as an HTML file.



COM1003 Assessments

Module COM1003

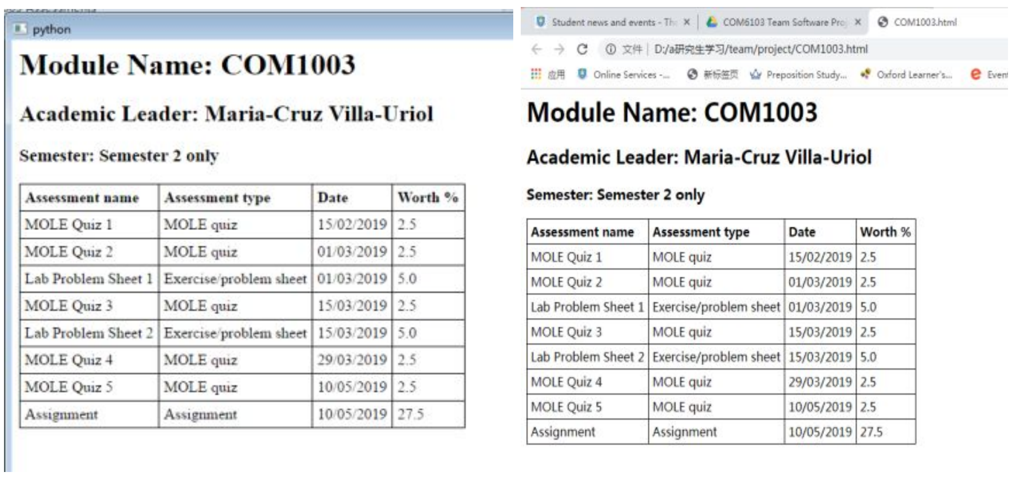
Academic Leader : Maria-Cruz Villa-Uriol

Semester : Semester 2 only

Assessments

	Assessment name	Assessment type	Date	Worth
1	MOLE Quiz 1	MOLE quiz	2019/2/15	2.50
2	MOLE Quiz 2	MOLE quiz	2019/3/1	2.50
3	Lab Problem Sheet 1	Exercise/problem sheet	2019/3/1	5.00
4	MOLE Quiz 3	MOLE quiz	2019/3/15	2.50
5	Lab Problem Sheet 2	Exercise/problem sheet	2019/3/15	5.00
6	MOLE Quiz 4	MOLE quiz	2019/3/29	2.50
7	MOLE Quiz 5	MOLE quiz	2019/5/10	2.50
8	Assignment	Assignment	2019/5/10	27.50

Figure 4. Booklet



python

Module Name: COM1003

Academic Leader: Maria-Cruz Villa-Uriol

Semester: Semester 2 only

Assessment name	Assessment type	Date	Worth %
MOLE Quiz 1	MOLE quiz	15/02/2019	2.5
MOLE Quiz 2	MOLE quiz	01/03/2019	2.5
Lab Problem Sheet 1	Exercise/problem sheet	01/03/2019	5.0
MOLE Quiz 3	MOLE quiz	15/03/2019	2.5
Lab Problem Sheet 2	Exercise/problem sheet	15/03/2019	5.0
MOLE Quiz 4	MOLE quiz	29/03/2019	2.5
MOLE Quiz 5	MOLE quiz	10/05/2019	2.5
Assignment	Assignment	10/05/2019	27.5

D:/a研究生学习/team/project/COM1003.html

Module Name: COM1003

Academic Leader: Maria-Cruz Villa-Uriol

Semester: Semester 2 only

Assessment name	Assessment type	Date	Worth %
MOLE Quiz 1	MOLE quiz	15/02/2019	2.5
MOLE Quiz 2	MOLE quiz	01/03/2019	2.5
Lab Problem Sheet 1	Exercise/problem sheet	01/03/2019	5.0
MOLE Quiz 3	MOLE quiz	15/03/2019	2.5
Lab Problem Sheet 2	Exercise/problem sheet	15/03/2019	5.0
MOLE Quiz 4	MOLE quiz	29/03/2019	2.5
MOLE Quiz 5	MOLE quiz	10/05/2019	2.5
Assignment	Assignment	10/05/2019	27.5

Figure 5. Display as PDF and HTML, respectively

10. Uncompleted Features:

On last check, all required features were completed.

11. Conclusion:

Our team was able to deliver a complete module delivery system that we believe will allow academic tutors and LTMs able to perform their yearly tasks with ease. Our system enables academics to enter their own data and constrains them to enter correct data. The automatic validation of data in our system eliminates the need for manual checking of the data. Academic tutors can automatically visualize assessments and generate module booklets.

The OASIS (Team 6)
Date: May 8th, 2019

In addition to delivering complete software, we learned a great deal about delegating and completing tasks as a team, as well as many new skills such as working with GUIs in Python.