ASSESSMENT TIMETABLE AND SCHEDULING SYSTEM

CM3203 – ONE SEMESTER INDIVIDUAL PROJECT 40 CREDITS

Nicole Kan – C1645959

May 2019

Supervisor: Helen Phillips

Moderator: Paul Rosin

Acknowledgements

I would like to thank my supervisor Helen Phillips for supporting me throughout this project as well as offering guidance and constructive feedback. It has greatly helped to tackle any issues I faced and ensure the completion of the system within this project.

I would also like to thank my friends and family who have supported me over the last 3 months and helped me keep going whenever I hit a roadblock.

Abstract

This report documents the work done to create an automated system to make assessment timetable pdfs for each year group within the School of Computer Science and Informatics at Cardiff University. The main aim of this project is to make the assessment timetabling process quicker and easier to manage which has been achieved by creating a standalone java application that makes the timetables for the user. The system also allows a couple of additional functionalities such as selecting which master's modules are to be put into the timetable and allowing the user to create an updated timetable when they wish to change an assessment's hand out/hand in dates.

Throughout this project a combination of Agile and Waterfall development methodology has been used to manage its progress. The requirements for this project were discussed with the client before being finalised and used in the short sprint like cycles of development. Upon completion of development, testing with test cases and usability testing was conducted to assess if the system was a success.

With the system passing all of its tests it has been deemed successful and although there are a couple of minor errors discovered upon usage after testing, the basic functionality of the system works whilst also allowing for potential future works.

Table of Contents

List of Figures

List of Tables

1. Introduction and Background

1.1 The problem

The School of Computer Science and Informatics at Cardiff University currently provides students and staff with a coursework assessment timetable PDF that is developed by each year group's tutor. Despite there being a workflow (found in Appendix 1) of when in the prior academic year these timetables are made, there is no specific outline for how they should be made other than them generally being an Excel spreadsheet (Phillips H, 2019a). Since the timetables are produced by a collection of people rather than just the one and are made by hand, this means they are often inconsistent in layout and contents as well as being time consuming to make. Additionally, this means altering the timetable if/when change requests come through during the academic year also has to be done manually to produce the corrected assessment timetable again.

The aim of this project is to create a dynamic system that is able to easily generate coursework timetables for all years that are consistent, understandable and easy to use. This will be achieved by creating a GUI for a Java program that automatically creates the PDF timetables. The system will also create the timetables in a faster timeframe than when it is done by hand allowing the examination officer(s) and the assessment and feedback lead longer to analyse the time frames for assessments and any assessment bunching that occurs. This will then hopefully reduce the number of change request forms that are created throughout the academic year as they will be able make any relevant changes prior to the timetables prior to being sent out to the students at the start of the academic year.

1.2 Background

1.2.1 Current timetable creation workflow

As previously mentioned, there is workflow that documents roughly how the assessment timetables are currently made and the timeframe in which they are made. It first collects the number, detail and dates of the assessment as part of the module review form in February/March of the previous academic year and it is this information that is used to create the timetables. Once the module leaders for the following academic year are approved, lecturers are asked to confirm or alter their assessment dates. The assessment and feedback lead will then examine the timetables to reduce bunching and ensure that the assessment deadlines are suitable. As the preparation for the academic year starts the timetables are then published in the exam share area allowing staff and admin to access them when needed, whether that be for moderating, checking and setting assessments. As students begin the academic year, the timetable for each year is then made available to each year's students via learning central. During the academic year, any changes that need to be made to an assessment then have to go through the change request process, requiring the assessment timetable to be re-made and shared again.

This current process of creating the timetable creates the following problems:

- Since the workflow does not address the layout of the timetable or what information the timetable must contain, they are often inconsistent across the year groups and from academic year to academic year since it is different people creating them each time.
- The dates in the timetable can be/have been wrong in the past since the timetable is copied from previous years and the assessment data is altered but other important information, such as term dates are not.
- Due to the manual method of creating the timetable if an assessment is updated, the timetable often is not. This means students and staff continue using the original assessment timetable that is no longer correct.

- Feedback periods for assessments often do not take into account times when the university is shut and stuff are on holiday, i.e. over Christmas and Good Friday and Easter Monday. Whilst the feedback timeframe is 4 weeks, it is counted as 20 working days, therefore bank holidays etc. should alter the feedback timeframe.

1.2.2. Target user(s) and the need for the program

The target user(s) for this program would be those who help organise each year's assessments and provide students with the assessment information at the beginning of the academic year. This includes the assessment and exam lead and the director and deputy director of teaching for the School of Computer Science and Informatics. Whilst it is the head tutor of each year group that currently makes the timetable, they would not be the target user(s) for this program as the program would allow the assessment and exam lead to automatically generate the timetable for all years in one go.

This program has been chosen to be implemented for various different reasons. The first being to address the problems discussed in section 1.2.1 Current timetable creation workflow of this report. Also, during the Periodic review for the School of Computer Science and Informatics improving the assessment scheduling was put as an action to revise the process for assessment information collecting and scheduling in the timeframe January 2019 (Phillips H, 2019b).

Additionally, Cardiff University follows a set of Academic Regulations which requires the head of the school to deliver to students in writing at the beginning of each session (i.e. the beginning of each term/year) the following information regarding coursework: the form of any coursework requirements, the deadlines and procedures for the submission of coursework, if any electronic plagiarism detection will be put in place and the timing of any viva voce that contributes to the overall assessment of a module (Cardiff University 2019, p.100). Having this information be accurate and correct and less likely to change throughout the academic year can also benefit Student Support at Cardiff University who offer additional learning support. If they were to be given an assessment timetable at the beginning of each term/academic year then they would know when the workload for students is higher and therefore more likely to require additional learning support. They would then know when more staff are likely to be required to ensure adequate support for students.

Furthermore, staff are encouraged to spread the assessment load in the Guidance on the Nature and Volume of Assessment in Modules on Taught Programmes of Study (found in Appendix 2) across modules to ensure an even spread for both students and staff. This also allows students to submit work that accurately represents their knowledge, skills and understanding of a module when the workload is spread evenly. However, making sure the assessments are spread evenly throughout a term takes time, especially when year groups start having optional modules and multiple different combination of modules need to be taken into account. If changes to assessments are then needed this is even more time consuming. With the assessment timetables and analysing being done in one time frame, as seen from the workflow diagram in Appendix 1, and the timetables being made by hand by multiple different people, the time available for the assessment and exam lead or director/deputy director of teaching to ensure the workload is spread evenly is varied and can sometimes not be adequate to make required or recommended changes before the timetable is required to be shared to students and staff. The automatic creation of these tables would then mean they are quicker to make allowing more time to alter them where necessary.

2. Approach

2.1 Choosing a development process

When approaching the development of this software I considered two development methodologies: Waterfall Methodology and Agile Methodology. Waterfall approaches a project in a sequential, linear approach. The progress of the project cascades down the stages starting with requirements gathering, to system design, then implementation, followed by testing, to delivery and finally ending with maintenance.

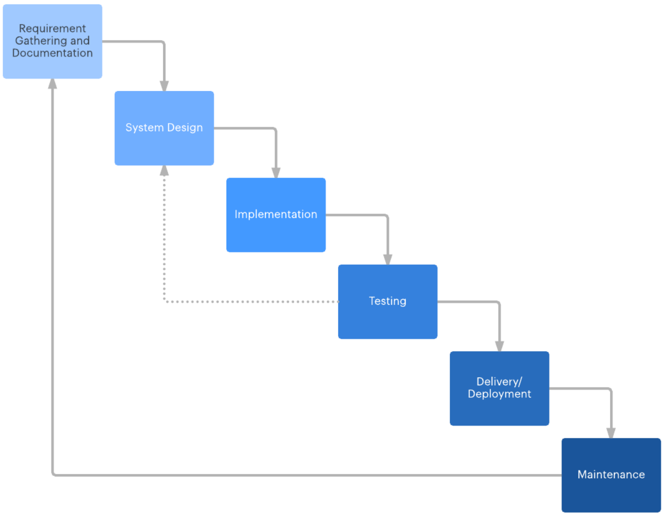


Figure 2.1 Waterfall Model
(LucidChart Content Team, 2017)

The distinct phases in Waterfall means that the next phase of the project can only begin once the previous one has been completed. This also means that once a phase is completed it is not possible to revisit it; to revisit a phase the only way is to restart from phase one. Since clear milestones are set in phase one it makes tracking progress of a project easy and is best used when requirements are not likely to change during development and when there is a set time frame for delivering the project (LucidChart Content Team, 2017).

Agile on the other hand is common where flexibility and high collaboration is required during the project. It also encourages frequent communication between the developers of the project and those who will ultimately be using the deliverable. This is because agile implements the deliverable in iterations allowing it to be incrementally accepted, making it easier to make any changes from client feedback at the end of each iteration and change the requirements throughout the project (Lonergan, K. 2016).

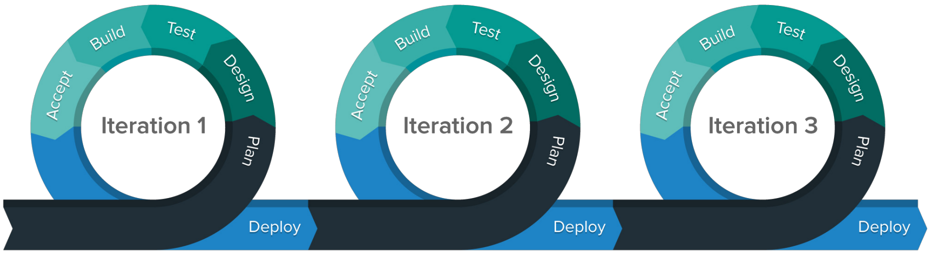


Figure 2.2 Agile Model
(Pivotal Tracker, 2017)

Having used both of these methodologies before I decided to combine them for this project and use an iterative and incremental development methodology (Inflectra, 2015). Using this

methodology, it allows me to have an incremental progression in my project whilst also allowing an iterative process for the implementation of the software.
This model follows the waterfall model for the creation of the initial requirements and design of the system but allows the flexibility of agile for implementing and developing the software. Each of the cycles will be 1-2 weeks with testing and feedback from the user given at the end of each cycle. Feedback from the previous cycle will then implemented in the next cycle so that the software is constantly being improved as it is being developed. Once all the requirements have been met, final user testing will be carried out and the deliverable will be released.

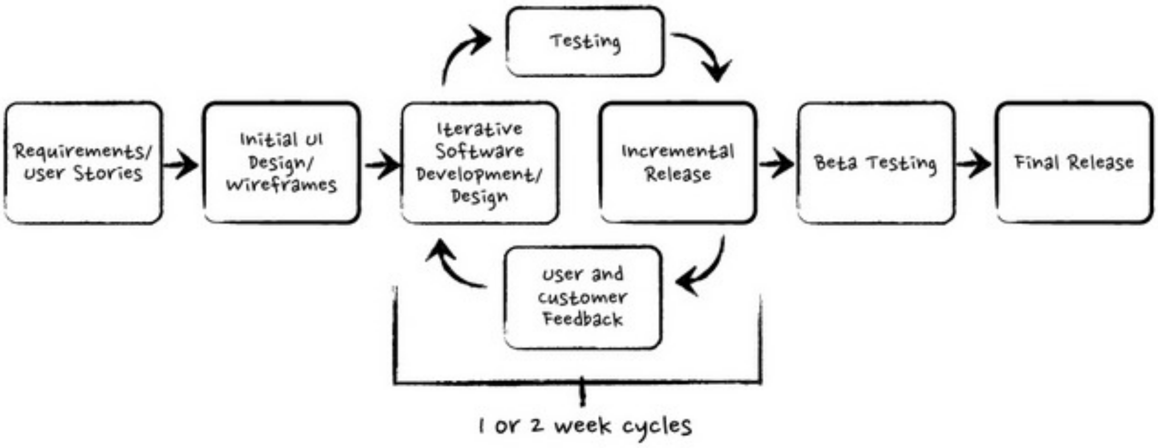

Testing
Requirements/ user Stories
Initial UI Design/ Wireframes
Iterative Software Development/ Design
Incremental Release
Beta Testing
Final Release
User and customer Feedback
1 or 2 week cycles

Figure 2.3 Hybrid Model
(Darmawan, NB. 2014)

In order to keep track of the iterations for this project and ensure code integrity is maintained I will be using GitHub and uploading the source code to a repository at the end the software development part of each short 1-2-week cycle.

2.2 Project Scope and Outcomes

The scope of this project is limited to The School of Computer Science and Informatics at Cardiff University and the current way that information regarding the assessments is collected and stored. The scope of this project does not allow for altering the amount of information contained in the PDF timetable outputs but can be included for future works.

The main outcomes of this project are the following.

1. An application that can be used to automatically create assessment timetables in the form of pdfs for all year groups in the school when given a csv file containing the assessment data. This application should be able to run on the main three computer operating systems: Windows, Macintosh and Unix and be self-explanatory to use.
2. Documentation of the overall project, including requirements, system design, the implementation process, user testing, a full code base and other relevant or essential information.

2.3 Assumptions

During this project the following assumptions are made:

- All hand in and hand out dates for coursework are during term weeks of a semester, i.e. not in exam or recess weeks.
- Both semesters, Autumn and Spring are 12 weeks long. This excludes recess weeks and exam weeks.
- The exam period in a term starts the week following week 12 in the relevant term.
- The exam period for Autumn exams is always 2 weeks.

- The recess weeks of a term are between week 11 and week 12.
- The start dates of a term is always a Monday.
- The user has access to all of the source code in order to download it to use the program.
- The user has Java installed on their machine and are able to run Java programs from their command line.
- The csv file is assumed to have at least the following information for each assessment in the order listed: module code for assessment, module title for assessment, type of assessment, contribution, assessment title, hand out week and hand in week. A further two columns of information can be added into the csv file if necessary but will not be required for making the pdfs.
- Each year group has a set year code that is used at the beginning of each module code to distinguish which year group the module is studied in.

3. Specification and Design

3.1. Functional and Non-Functional Requirements

Functional requirements specify what the system what the system does, and non-functional requirements specify how the system works (Eriksson, U. 2012). For this project I will be splitting the requirements into 3 different tiers: must have, should have and could have. Must have includes requirements that without them there is no point in delivering the product and a viable solution cannot be delivered without them. Should have requirements are ones that are important but not essential, they may require some kind of temporary workaround for the given delivery date that at a later point can be fixed with a better solution. Could haves will contain requirements that are desirable but less important, i.e. they will have less of an impact if left out compared to should have requirements (Agile Business. 2014).

The following requirements have been decided on after discussion with the client on what they wish the main functionalities of the system to be along with additional desirable ones that if not completed in this iteration of the project could be implemented in future work.

3.1.1. Functional Requirements:

Must have requirements:

Requirement	Acceptance Criteria
The system must be able to take csv data as the input for creating the timetable pdfs.	The user is able to enter the name of a csv file containing all the assessment data that is then used to create the timetables into the system.
The system must create pdfs containing each year groups assessment timetable as the output.	The user will be able to save the timetable pdfs to a directory on their machine.
The system must be able to let the user alter an assessment's hand in/hand out dates.	The system allows the user to enter all the relevant information for updating the dates of an assessment and create a new updated timetable for the year group containing this assessment.
The system must let the user know when information used in creating the table for an assessment is missing.	The pdf timetables will highlight any missing information that is meant to be present.
The system must have a graphical user interface (GUI).	When the system is run a GUI is launched that allows the user to enter all the required information to create the assessment timetable pdfs.

Table 3.1

Should have requirements:

Requirement	Acceptance Criteria
The system should take into account days the university is closed when showing feedback dates on the timetable.	If an assessments period falls over days the university is closed or staff are on holiday, i.e. Christmas and Easter, the typical 4 working weeks feedback time will be extended (2 additional weeks for Christmas and 1 additional week for Easter).
The system should let the user generate and choose which master's modules they want included in the master's assessment timetable.	The user is able to click a button that shows them all the master's modules and select the ones they want to be in the timetable. If they choose not to select them, all modules are automatically included.

The system should remind the user to alert staff members and students of an updated table when they update an assessment and create a new table.	Once the user has updated an assessment the system will remind the user to upload the updated timetable to the relevant places for both staff and students and send an email out alerting people to this change.

Table 3.2

Could have requirements:

Requirement	Acceptance Criteria
The system could be able to provide multiple different formats of an assessment timetable with different amounts of information in each format.	The user is able to choose between two versions of the timetable. One contains the essential information and the other contains essential information as well as additional information that could be considered useful.
The system could be linked to the change request forms.	When staff require a change request form during the academic year, the user can send it to that member of staff from the system. Once it has been filled in by the staff member and approved by the user, the system will then automatically create a new updated timetable.

Table 3.3

3.1.2. Non-Functional Requirements:

Must have requirements:

Requirement	Acceptance Criteria
The system must create an output of timetables that are consistent in design and easy to use and understand data from.	The layout and colour scheme for each year group's timetable is the same and they all contain the same level of information. In user testing at least 70% of users agree the timetable is easy to use and understand.
The system must have a GUI that is easy to use, learnable and overall user-friendly.	The GUI launched by the system adheres to Jakob Nielsen's user interface heuristics (Nielsen, J. 1994) when being designed and scores at least 70% on user testing.
The system must be reliable.	The system will not fail at random intervals and will always produce pdf timetables when given csv data.
The system must be able to run on multiple different operating systems.	The system can run on the 3 main operating systems: Windows, Macintosh and Linux without error.

Table 3.4

Should have requirements:

Requirement	Acceptance Criteria
The system should have a suitable response time for creating the timetable pdfs.	The system is faster than creating the timetables by hand. When the user asks the system to create the tables there should not be a significant delay (no more

	than 10 seconds) in letting them know they have been created.
The system code should be easily maintained and updated in the future.	The system is comprised of code in classes and each file does not exceed 1000 lines of code.

Table 3.5

Could have requirements:

Requirement	Acceptance Criteria
The system could have a feedback button to make maintenance of the program easier.	The user is able to click a button that links them to a feedback form regarding the system.

Table 3.6

3.2. Testing

In order to see if the system requirements are met, I will be using a combination of test cases and user testing. The requirements that I will be testing to ensure they meet the acceptance criteria will be the 'must have' and 'should have' requirements. Since the could have requirements are of the lowest priority for this project and are not deemed essential or of high importance for the timeframe of this project, I will not be testing them in this report. The requirements that require a step by step procedure to see if they meet their acceptance criteria will be tested using test cases and those that can't, i.e. seeing if users agree the GUI is easy to use, will be tested through user testing.

3.2.1. Test Cases

Since there is large number of test cases, only the ones that primarily focus on the functionality of the system achieving the aims set out earlier on in this report are shown in this section. For the rest of the test cases please refer to appendix 3.

Test Case 02			
Requirement:	The system must create pdfs containing each year groups assessment timetable as the output.		
Pre-condition(s):	The user has the system downloaded on the machine they are using. The user has java downloaded on the machine they are using the system on. The user has the system open. The csv file with the assessment data in is in the same directory as the code for the system.		
Test Case Steps			
Steps	Expected Outcome	Actual Outcome	Pass/Fail
1. The user inputs all the required data into the GUI text fields. 2. The user clicks the make timetables button. 3. User opens the created timetable(s).	The system lets the user know the timetables have been made and user is able to open the pdf timetables with a pdf reader.		
Checker & Date of Check:		**Author:** Nicole Kan	

Table 3.7

Test Case 03	
Requirement:	The system must be able to let the user alter an assessment's hand in/hand out dates.
Pre-condition(s):	The user has the system downloaded on the machine they are using.

	The user has java downloaded on the machine they are using the system on. The user has the system open. The csv file with the assessment data in is in the same directory as the code for the system.		
Test Case Steps			
Steps	Expected Outcome	Actual Outcome	Pass/Fail
1. The user fills in the required text fields in the GUI. 2. The user enters the details of the assessment they wish to change in the change module section of the GUI. 3. The user clicks the new timetable button.	The system lets the user know a new timetable and user is able to open the updated timetable pdf.		
Checker & Date of Check:		**Author:** Nicole Kan	

Table 3.8

Test Case 05			
Requirement:	The system must have a graphical user interface (GUI).		
Pre-condition(s):	The user has the system downloaded on the machine they are using. The user has java downloaded on the machine they are using the system on.		
Test Case Steps			
Steps	Expected Outcome	Actual Outcome	Pass/Fail
1. User opens the terminal/command line. 2. User changes to the directory containing the source code for the system. 3. User enters the compile command as found in the ReadMe.txt. 4. User enters the run command as found in the ReadMe.txt.	The system will open a GUI on the user's machine allowing them to create the timetables.		
Checker & Date of Check:		**Author:** Nicole Kan	

Table 3.9

Test Case 11			
Requirement:	The system should have a suitable response time for creating the timetable pdfs.		
Pre-condition(s):	The user has the system downloaded on the machine they are using. The user has java downloaded on the machine they are using the system on. The user has the system open. The csv file with the assessment data in is in the same directory as the code for the system.		
Test Case Steps			
Steps	Expected Outcome	Actual Outcome	Pass/Fail
1. User enters all the required information into the text fields in the GUI. 2. The user clicks the make timetables button.	The user is waiting less than 10 seconds between telling the system to make the timetables and being told they have been made.	Less than 10 seconds wait between choosing the directory to save the timetables in and being	PASS

3. User opens the timetable(s) created.		told by the system they have been made.	
Checker & Date of Check: Nicole Kan 16/04/2019		**Author:** Nicole Kan	

Table 3.10

3.2.2 User Testing and Ethics Approval

For the remaining requirements I will create a short questionnaire that asks the users for their opinions on the given requirements. This questionnaire can be found in appendix 4 and the results of it will be evaluated against the acceptance criteria for each requirement in the evaluation section of this report.

Since I will be using human participation in this part project, I require formal ethical approval from the university. Please refer to appendix 5 for the consent forms and information sheets used in the testing part of this project.

3.3. Use Cases

The use cases below show and describe possible interactions between a user and the system in order to achieve certain goals. These help to clarify and show how the requirements will be used by a user. For all of the use cases there is one user who is described as an Assessment System user, this refers to any of the target users mentioned earlier in this report.

Use Case 1

Use Case Name:	Create assessment timetables.
User Type:	Assessment System User
Description:	The user will be able to input a csv data file into the system and have it automatically create the assessment timetables for them.
Pre-conditions:	User has the system downloaded on their machine. User has Java downloaded on their machine. User has the csv data file containing the assessments they wish to make tables from. The csv data file is in the same directory as the system.
Post-conditions:	The user now has assessment timetables for all year groups that were in the csv data file saved in a directory of their choosing.
Basic Flow	1. User opens the system using the command line/terminal commands. 2. User enters all the required information into the field text boxes. 3. User clicks the make timetables button.
Alternative Flow:	1. User opens the system using the command line/terminal commands. 2. User enters the csv file name into the corresponding text field box. 3. User clicks the Generate Master's modules button. 4. User selects which modules from the master's year they wish to be in the master's timetable. 5. User enters all the other remaining necessary information. 6. User clicks the make timetables button.

Table 3.11

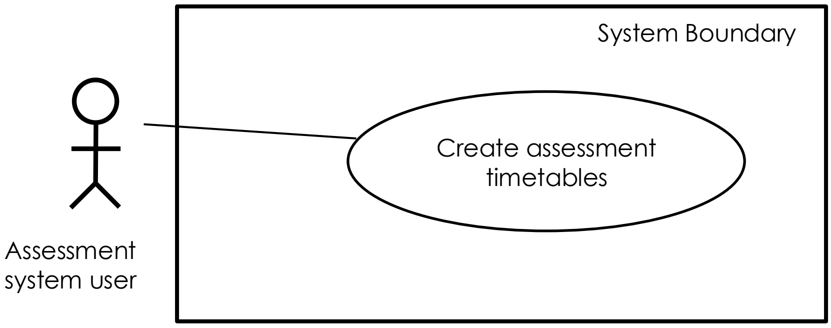


Figure 3.1 Use Case 1

Use Case 2

Use Case Name:	Changing an assessment's dates.
User Type:	Assessment System User
Description:	The user is able to remake the timetable for a year where one of the assessments hand in or hand out date needs to be changed.
Pre-conditions:	User has the system downloaded on their machine. User has Java downloaded on their machine. User has the csv data file containing the assessments they wish to make tables from. The csv data file is in the same directory as the system.
Post-conditions:	The user has an updated timetable for the year containing the changed assessment.
Basic Flow	1. User opens the system using the command line/terminal commands. 2. User enters the required information for making the timetables, i.e. dates, csv file name etc. 3. User enters the information for the assessment they wish to alter. 4. User clicks the new timetable button.
Alternative Flow:	1. User opens the system using the command line/terminal commands. 2. User enters the csv file name into the corresponding text field box. 3. User clicks the Generate Master's modules button. 4. User selects which modules from the master's year they wish to be in the master's timetable. 5. User enters all the information required when making the pdfs usually. 6. User enters the information needed for changing an assessment. 7. User clicks the new timetable button.

Table 3.12

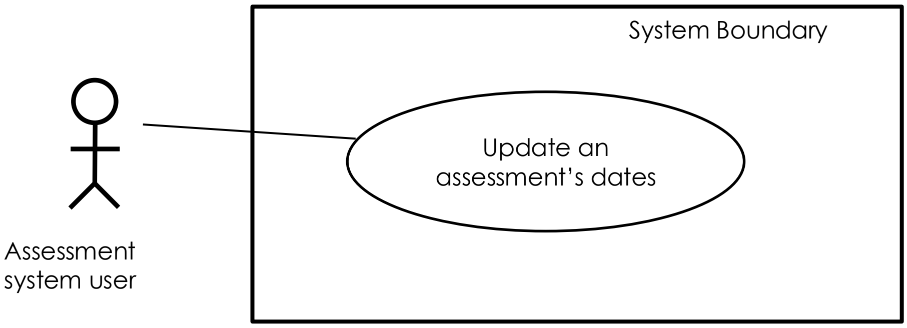


Figure 3.2 Use Case 2

3.4. Design

The design of this system has been decided by looking at the project brief and through discussions with the client. The functionalities of the system have been clarified and prioritised and can be seen in the requirements section.

To begin with I have created a draft of what the output of the timetable should look like. In order to make this draft I have looked at previous year's timetables. I have also looked at a timetable that was the output of a Python script that was created by another member of staff as a temporary solution to the timetable assessment making problem.

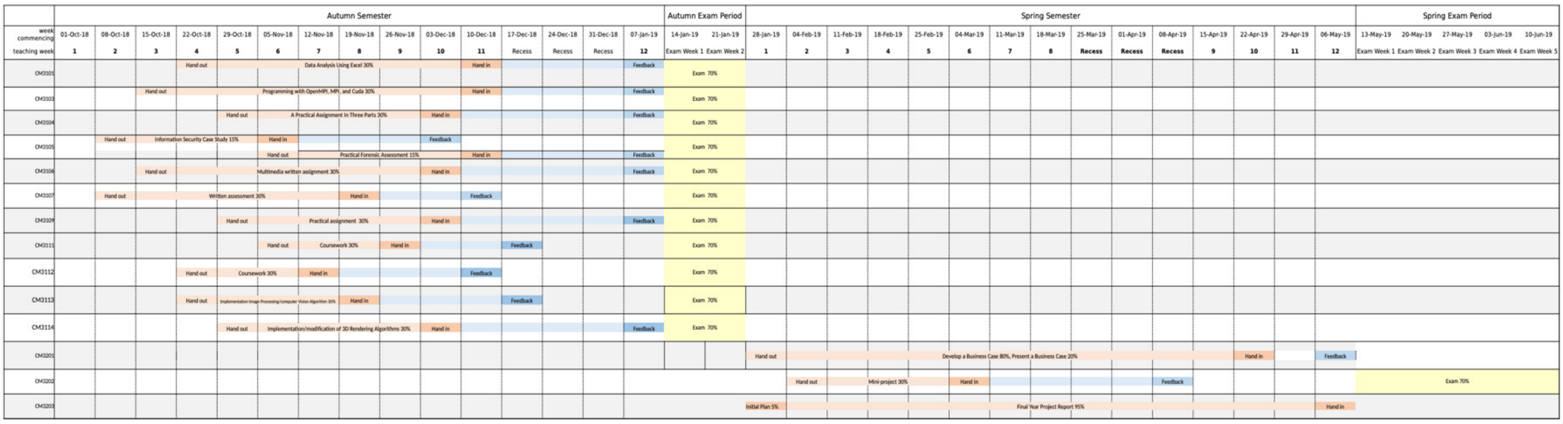


Figure 3.3 2018-2019 ComSc Year 3 assessment timetable

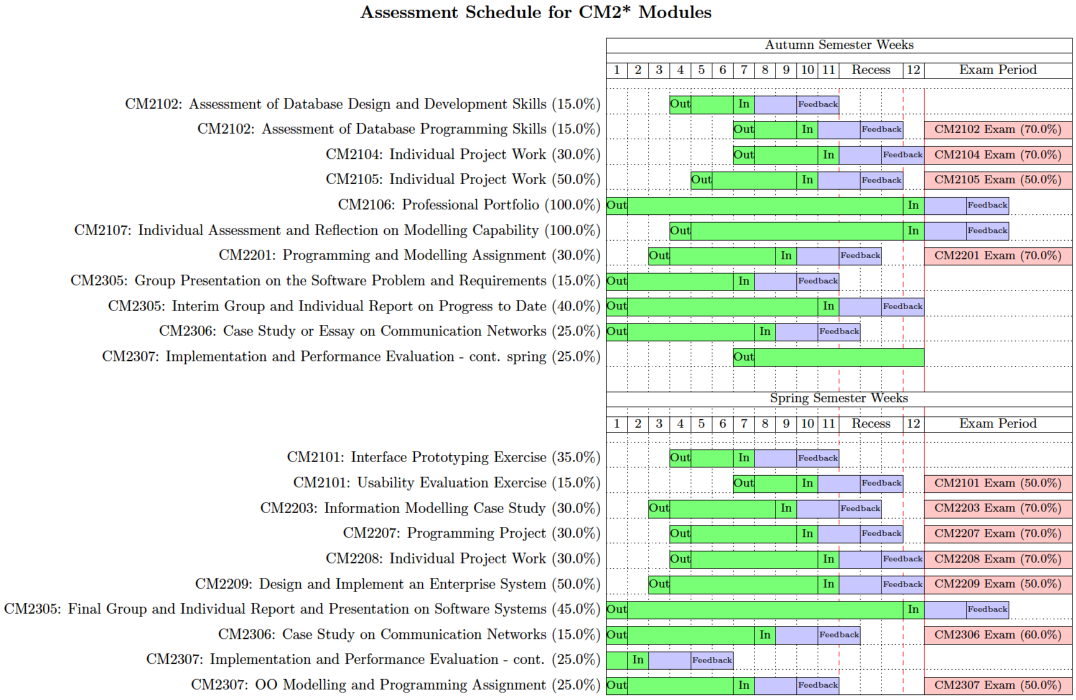
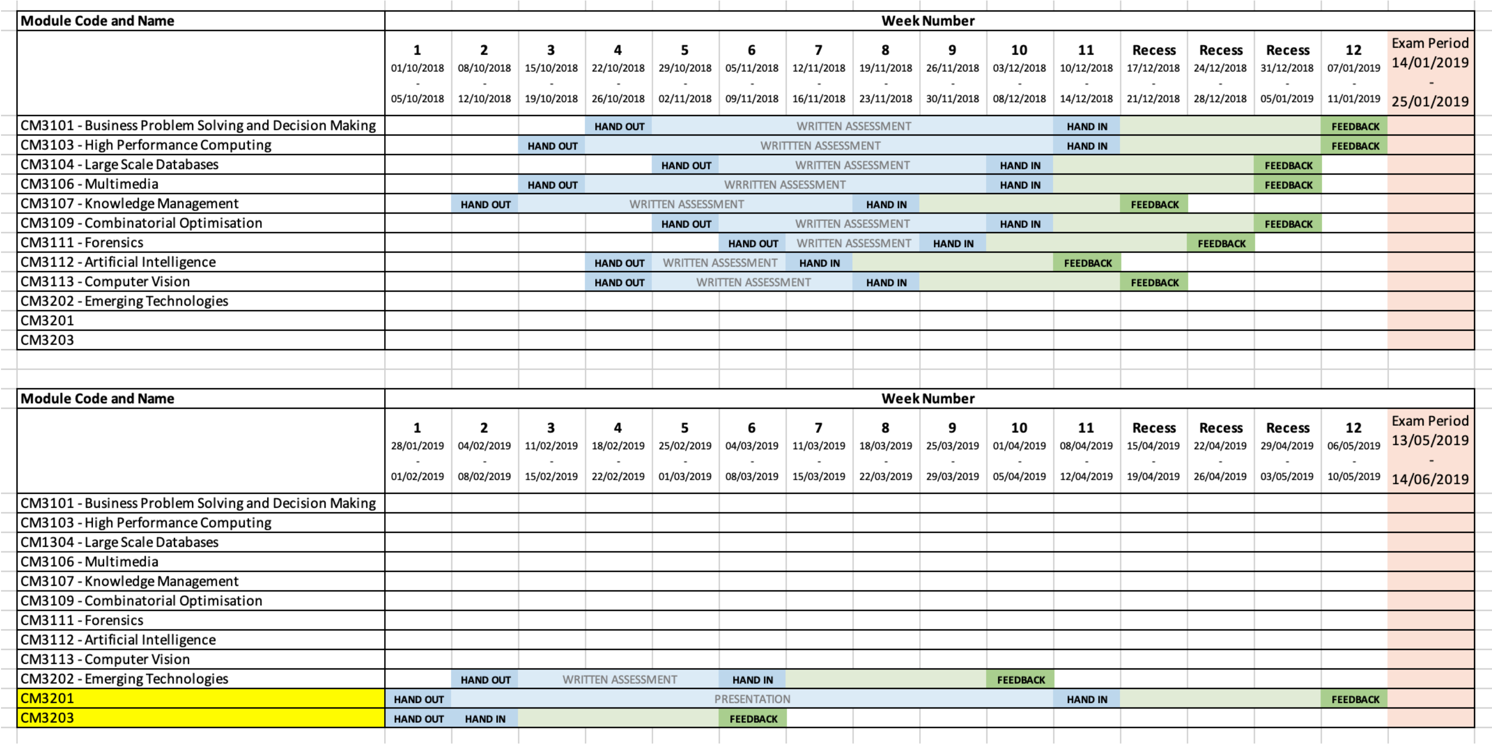


Figure 3.4 ComSc Year 2 assessment timetable from Python script

Figure 3.3. shows the assessment timetable for year 3 computer science students in the 2018/2019 academic year and figure 3.4 shows the assessment timetable generated by the python script for year 2 computer science students in the same academic year. For the draft design that I have made as the output for my assessment timetabling system I have tried to take into account the things that make these two timetables difficult to use.
These being:

- Difficulty when reading and trying to use the timetable. This is particularly a problem in figure 3.3 because the entire year has been put into one table meaning the font size used has to be small in order to fit into the cells.
- Both timetables fail to give the user the corresponding module name for a module code. This makes it difficult for students and staff to know what module a piece of assessment is for. Whilst both the module code and module name are used in learning materials in a module, it is typically referred to by the module name in speech and general conversation regarding the module, making identifying the module using a code from the timetable harder than it could be.



Module Code and Name	Week Number															
	1 01/10/2018 - 05/10/2018	2 08/10/2018 - 12/10/2018	3 15/10/2018 - 19/10/2018	4 22/10/2018 - 26/10/2018	5 29/10/2018 - 02/11/2018	6 05/11/2018 - 09/11/2018	7 12/11/2018 - 16/11/2018	8 19/11/2018 - 23/11/2018	9 26/11/2018 - 30/11/2018	10 03/12/2018 - 08/12/2018	11 10/12/2018 - 14/12/2018	Recess 17/12/2018 - 21/12/2018	Recess 24/12/2018 - 28/12/2018	Recess 31/12/2018 - 05/01/2019	12 07/01/2019 - 11/01/2019	Exam Period 14/01/2019 - 25/01/2019
CM3101 - Business Problem Solving and Decision Making				HAND OUT	WRITTEN ASSESSMENT						HAND IN				FEEDBACK	
CM3103 - High Performance Computing			HAND OUT	WRITTTEN ASSESSMENT							HAND IN				FEEDBACK	
CM3104 - Large Scale Databases					HAND OUT	WRITTEN ASSESSMENT				HAND IN				FEEDBACK		
CM3106 - Multimedia			HAND OUT	WRRITTEN ASSESSMENT						HAND IN				FEEDBACK		
CM3107 - Knowledge Management		HAND OUT	WRITTEN ASSESSMENT					HAND IN				FEEDBACK				
CM3109 - Combinatorial Optimisation					HAND OUT	WRITTEN ASSESSMENT				HAND IN				FEEDBACK		
CM3111 - Forensics						HAND OUT	WRITTEN ASSESSMENT		HAND IN				FEEDBACK			
CM3112 - Artificial Intelligence				HAND OUT	WRITTEN ASSESSMENT		HAND IN				FEEDBACK					
CM3113 - Computer Vision				HAND OUT	WRITTEN ASSESSMENT			HAND IN				FEEDBACK				
CM3202 - Emerging Technologies																
CM3201																
CM3203																

Module Code and Name	Week Number															
	1 28/01/2019 - 01/02/2019	2 04/02/2019 - 08/02/2019	3 11/02/2019 - 15/02/2019	4 18/02/2019 - 22/02/2019	5 25/02/2019 - 01/03/2019	6 04/03/2019 - 08/03/2019	7 11/03/2019 - 15/03/2019	8 18/03/2019 - 22/03/2019	9 25/03/2019 - 29/03/2019	10 01/04/2019 - 05/04/2019	11 08/04/2019 - 12/04/2019	Recess 15/04/2019 - 19/04/2019	Recess 22/04/2019 - 26/04/2019	Recess 29/04/2019 - 03/05/2019	12 06/05/2019 - 10/05/2019	Exam Period 13/05/2019 - 14/06/2019
CM3101 - Business Problem Solving and Decision Making																
CM3103 - High Performance Computing																
CM1304 - Large Scale Databases																
CM3106 - Multimedia																
CM3107 - Knowledge Management																
CM3109 - Combinatorial Optimisation																
CM3111 - Forensics																
CM3112 - Artificial Intelligence																
CM3113 - Computer Vision																
CM3202 - Emerging Technologies		HAND OUT	WRITTEN ASSESSMENT			HAND IN				FEEDBACK						
CM3201	HAND OUT	PRESENTATION									HAND IN				FEEDBACK	
CM3203	HAND OUT	HAND IN				FEEDBACK										

Figure 3.5 Initial timetable design

As seen in the figure 3.5 above the system will also highlight any missing information from the csv file that is needed to complete this table but currently not present. This is done to ensure that all the timetables will be consistent in their layout and content and also hopefully prevent having to update the timetable during term time for small changes that do not affect an assessment's dates. As in the previous timetable layouts this timetable shows the hand out, hand in and feedback dates via week numbers. I have chosen to put in the dates for the weeks as students often lose track of what week number they are on, especially after a recess since they do not count towards the week count and the week count resumes from what it was prior to the break. Whilst figure 3.4 shows that not having the dates in does make the table more compact and smaller, it does make the timetable easier to use and understand and since the year has been split into two tables it does not make the cells crowded and hard to read.

Since the requirement of having the system produce multiple different versions of the timetable is a 'could have' and therefore not a high priority for this project I have not designed another format that the system could produce.

Following the timetable design, the layout for the GUI could then be planned out as the table layout decided what information would need to be input by the user. For each assessment it is assumed what information about it is stored in the csv, as mentioned in the assumptions part of this report, meaning that the only information the user has to input to be able to create the timetable is the term dates. The GUI also has to allow for users to change an assessments dates and choose which master's modules are put into the timetable if they wish as these are part of the requirements for this system.

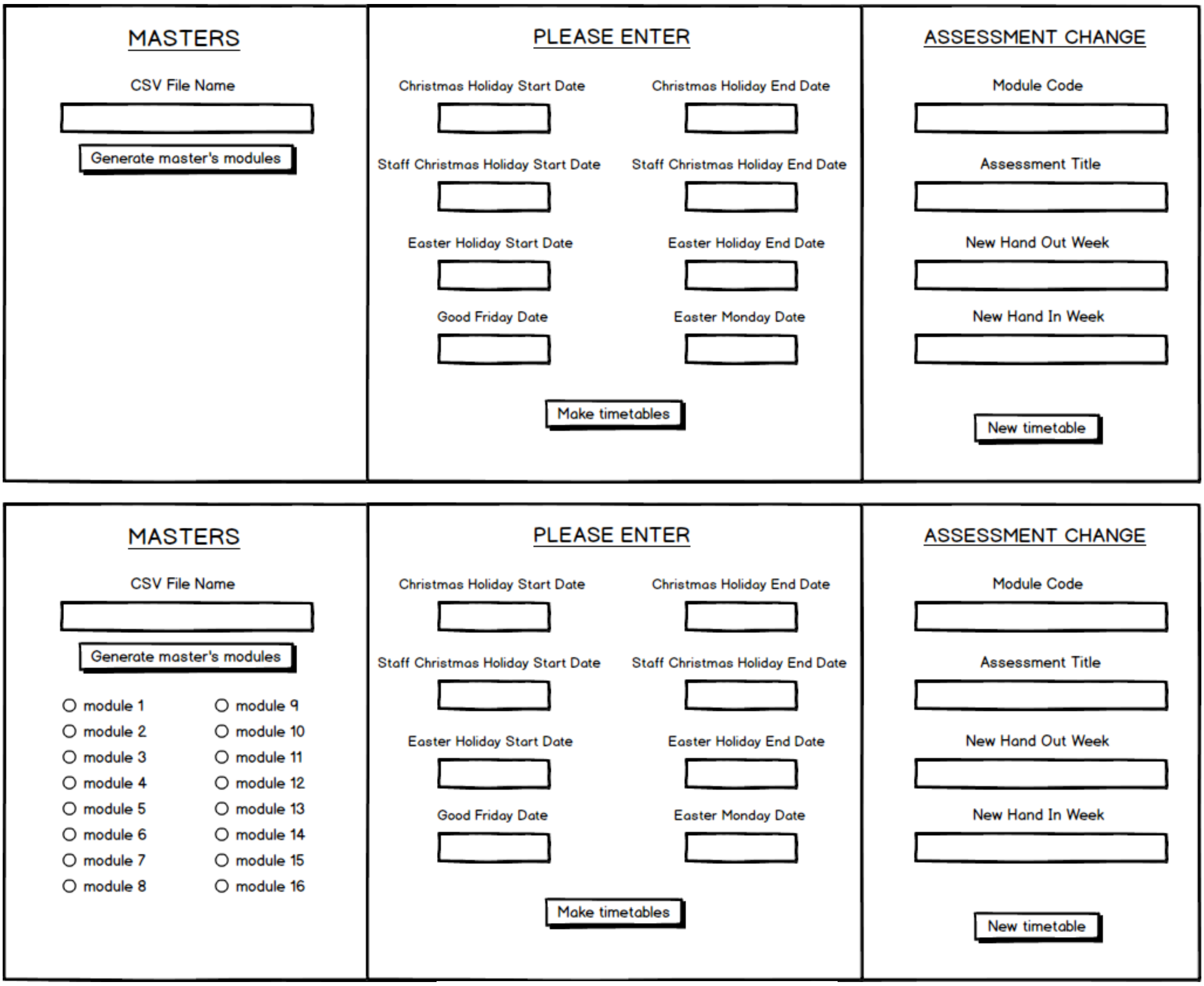


Figure 3.6 Initial GUI design

Above is the initial design of the system that has been created in Balsamiq. It shows the system having 3 panels, each of which is responsible for a different aspect of the system. The panel on the left allows the user to choose the master's modules that will be put into the assessment timetable, the middle panel allows the user to actually create the assessment timetables once they have input all the relevant term data and the right panel allows the user to alter an assessment's hand out or hand in date and create a new timetable for the year containing that assessment. The initial design has been designed with Nielsen's heuristics (Nielsen, J. 1994) in mind and the final design of the system will be evaluated against these in the Evaluation section of this report.

3.5. Java Vs Python

The programming language that will be used to implement this system has been decided by multiple factors. The first being what programming language I thought would best be able to fulfil the requirements and secondly by evaluating a small script that had previously been written by another member of staff as a temporary solution but does not fulfil all the requirements the client has requested in this project.

Despite Python being used in the temporary script and both Python and Java being platform independent (Medium, B. 2019) I have chosen to use Java as the programming language for this project. This is because I am slightly more familiar with Java than Python and I believe that Java will allow me to fulfil the requirements in a more elegant, efficient way, especially in regard to the GUI which is one of the most important requirements for this project. The importance of the GUI was highlighted by the temporary script as without a GUI it made it difficult for anyone who had not written the code to understand what needed to be done to make it create the timetables.

Additionally, although Python has a standard GUI library called Tkinter I have decided not to use it as this would require the machine running the program to have the Tkinter package downloaded in addition to Python (Tutorials Point. 2015) leading to some possible platform compatibility issues if the user is unable to download the additional package for whatever reasons, such as not having admin rights on a computer. With this in mind, Java has Swing which is a GUI widget toolkit that is written entirely in Java and therefore platform independent (Techopedia. 2011). Additionally, Java Swing only requires the machine to have Java downloaded as the import of the Swing elements will be handled in the code itself (javatpoint. 2014).

3.6. iText

Before starting to actually code my system I first had to make sure that there was some way for me to create a pdf from a Java program. Having done some research I found iText which is “A powerful PDF toolkit for PDF generation, PDF programming, handling & manipulation. The preferred PDF engine for Java…” (iText, 2019a). I also found a couple of other PDF generators that could be used with Java however I chose to use iText for 2 main reasons. The first being that there are plenty of resources to help those using iText, for example they have API documentation that covers all of the packages, classes, methods etc. that iText provides and they have FAQ’s that are answered with code snippets that help you to fully understand how the code works and what it does. The second reason for choosing iText is that it is free to use. iText allows you a 30 day free trial but will automatically revert to their AGPL open source version after this time frame. (iText, 2019b).

This means the iText Community code is freely available to use as long as the AGPL license is used correctly. The restrictions are as follows:

- When distributing an application using iText, the full source code must be disclosed.
- Any modifications made to the iText source code must be made aware to iText.
- iText Community can only be used in an AGPL environment.
- iText, the iText copyright and AGPL license must be included in the output file metadata, and the producer line must be retained in every PDF that is created or manipulated using iText.

(iText, 2019c).

3.7. Risk Assessment

In any software engineering project, there is always an element of risk. By completing a risk assessment prior to implementation and finding mitigations for them it allows me to achieve the aims of this project as effectively as possible.

Risk	Severity	Likelihood	Action to mitigate risk
Project requires technical knowledge beyond my expertise level.	Low	Medium	Communicate with supervisor regularly to ensure that requirements for this project are within my expertise or can be learned quickly. If not, then the requirement giving me an issue can be altered.
Initial plan with work plan underestimates the time taken to complete each stage of the project.	Medium	Medium	Weekly supervisor meetings to ensure things are staying on track and regular development milestones are set to align with longer review meetings. The cause of any timing set

			back will be discussed with my supervisor and changes made where necessary.
The implementation of the system breaks after final release of implementation.	Medium	Medium	Ensure all errors are handled properly and do not break the functionality of the system.
Poor feedback from user testing.	Medium	Medium	Design the GUI and the layout of the timetable with Nielsen's usability heuristics in mind (Nielsen, J. 1994).
Computer being used for implementation of the system crashes/is broken and code base lost.	High	Low	The code base will be uploaded to GitHub every time a new functionality is successfully implemented.
Any personal extenuating circumstances such as illness or an increase workload.	Low	Low	The initial plan has allowed for spare time and any assessment handouts have been taken into account with smaller targets around that time.

Table 3.13

4. Implementation

As mentioned in the Approach part of this report I chose to use an iterative approach when implementing my system. This means that the screenshots of the code in the following section are the result of the final iteration, however from the table below providing a brief overview of what was done in each iteration, it can be seen to certain extent how the final implementation was reached.

Iteration 1 18/02/2019 – 04/03/2019
• Got iText working with Java from the terminal/command line. • Getting familiarised with the iText library and getting a java script to create a blank table that is saved in a pdf document. • Java script is able to read in and store data from the assessment csv data file. • Initial GUI frame created but not linked to the java script.
Iteration 2 05/03/2019 – 18/03/2019
• Autumn term of an assessment timetable made for a single year group, holiday days for feedback period not taken into account yet. • GUI updated to have all required text fields, buttons and text added. • GUI and java script linked up and main class created to be able to run the system as a whole from the terminal/command line.
Iteration 3 19/03/2019 – 01/04/2019
• Missing information from the csv file is highlighted in the pdf timetables. • The spring term table is now also made and added to the pdf output. • System is made so that it produces a pdf for all year groups in the School of Computer Science and Informatics. • Functionality of choosing which master's module to be used in producing the master's timetable added.
Iteration 4 02/04/2019 – 11/04/2019
• Length of feedback period now takes into account holiday days. • GUI made to give feedback if any required text fields for a function is missing user input. • System allows the user to choose where they wish their files to be saved via a pop-up window.

Table 4.1

Code Practices

Throughout the implementation of this system I have used several coding practices to help ensure I was writing readable code and to make the system as efficient as possible. These include:

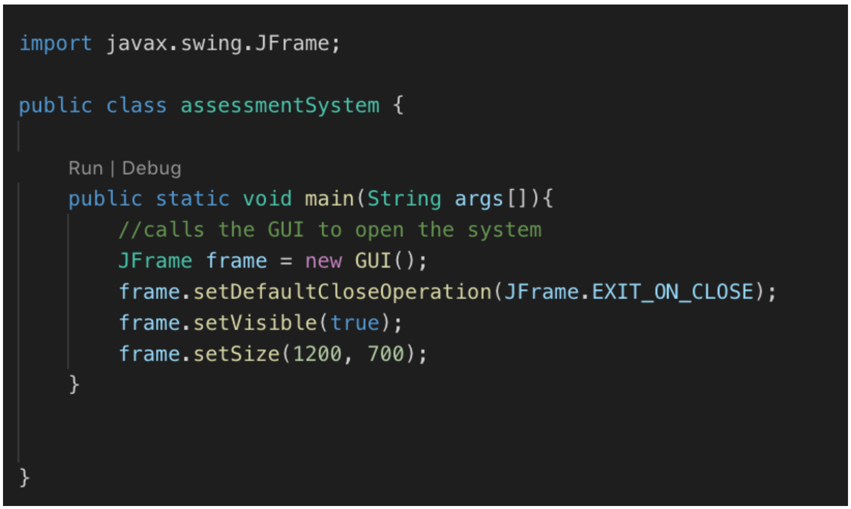
- Using an object-oriented approach that includes making use of encapsulation and abstraction.
- Commenting my code so that if someone new is working on or looking at the system code it will help them to understand any aspects they may be confused on.
- Reducing code duplication where possible, i.e. creating methods to carry out tasks which will often need to be done in the system.
- Declaring variables that may need to be changed by the user at a later stage such as each year groups code or variables that will used throughout multiple methods in the class at the top of each class file.

Source Code

Since this system is comprised of over 1000 lines of code, I will not be discussing every single aspect of the code but will instead focus on areas of the code that particularly contribute to the functionality of the system. If you wish to look at the code in its entirety, please refer to the source code zip file that has been uploaded with this report.

assessmentSystem.java

This class contains the main method and is what is will be run by the user when they wish to use the system. It simply calls the GUI class and instantiates a new instance of it that is set to visible and open on the user's computer to allow them to actually use the system.



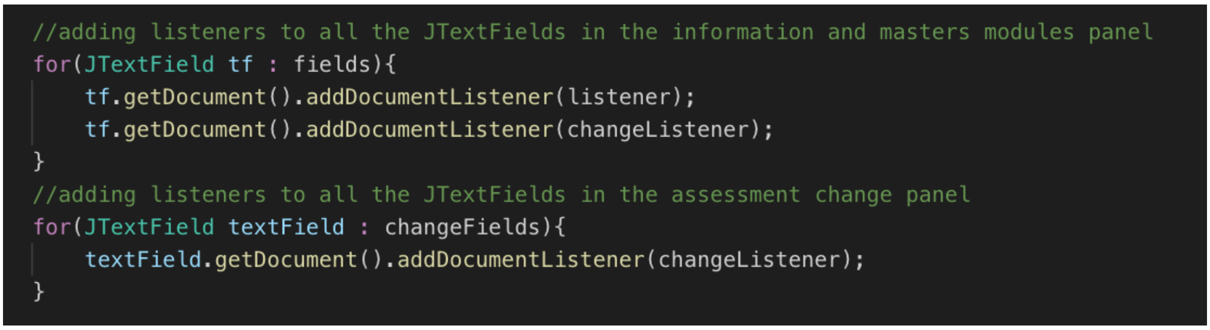
```
import javax.swing.JFrame;

public class assessmentSystem {

    Run | Debug
    public static void main(String args[]){
        //calls the GUI to open the system
        JFrame frame = new GUI();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
        frame.setSize(1200, 700);
    }


}
```

Figure 4.1

GUI.java

For this GUI class about half of the code is used to create the panels that form the GUI as well as the creation and addition of the elements that make up the GUI, i.e. the text fields, labels and buttons for each panel. This part of the code will not be discussed in this section as the code itself is fairly self-explanatory and this report intends to focus more on the implementation of features of the system.

The remainder of the class consists of defining the document listeners that are added to the text fields upon creation as well as defining what the action listeners should do when the event of a button being clicked activates them.

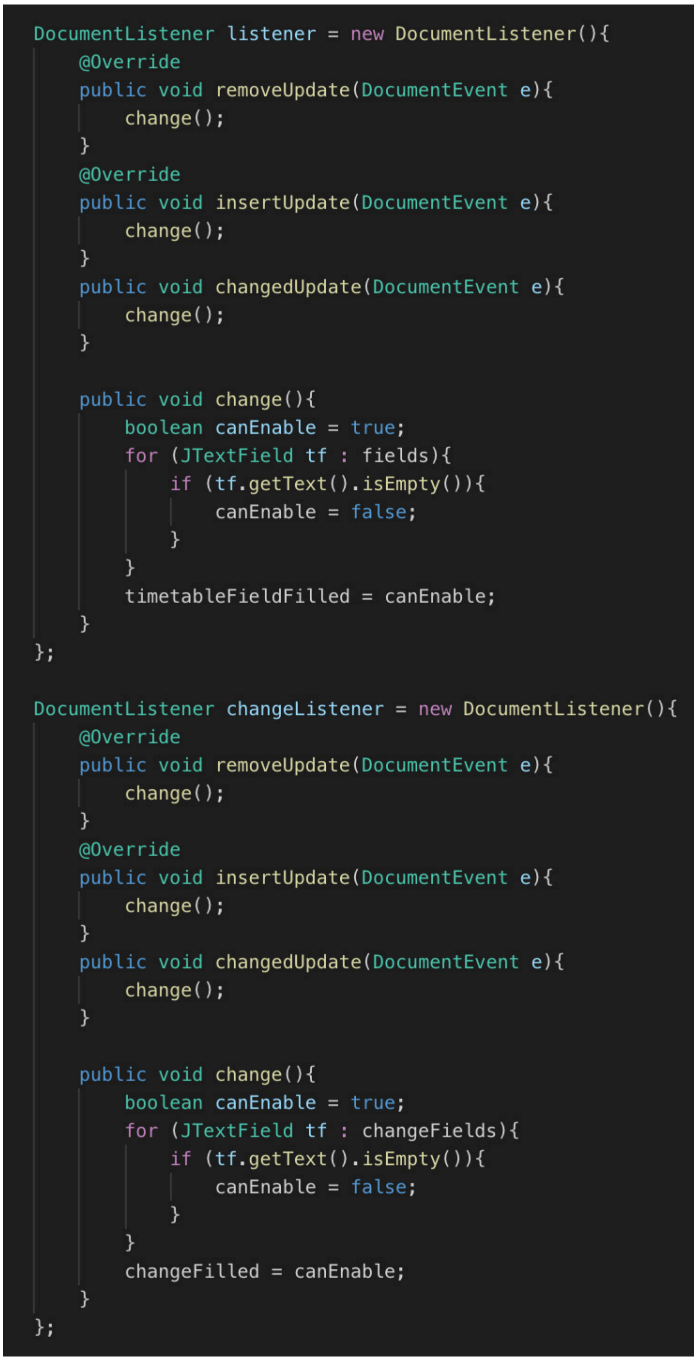


```
//adding listeners to all the JTextFields in the information and masters modules panel
for(JTextField tf : fields){
    tf.getDocument().addDocumentListener(listener);
    tf.getDocument().addDocumentListener(changeListener);
}
//adding listeners to all the JTextFields in the assessment change panel
for(JTextField textField : changeFields){
    textField.getDocument().addDocumentListener(changeListener);
}
```

Figure 4.2

Since the system requires user input to be able to make the assessment timetables, the system has been programmed so that if any of the required text fields are empty the system will alert the user to this and not make the timetables until they have all been filled in. This has been done by using document listeners that keep track of any updates, deletions and insertions into these text fields. There are 2 separate document listeners in this system as there 2 sets of text fields that need to be filled in for different functionalities within the system. The first being the set of text fields that is required for the creation of all year groups timetables; this consists of all the term dates input by the user as well as the name of the csv file containing the assessment data. The second set consists of the text fields in the first set as well as the text fields that need to be filled in by the user in order to change an assessment's hand out/hand in dates.

Figure 4.2 above shows the relevant document listeners being added to these sets of text fields. Since the term dates and csv file name need to be filled in for both functions, both document listeners, listener and changeListener, are added to it. The additional text fields required for changing an assessment's dates then also have the changeListener added to them.



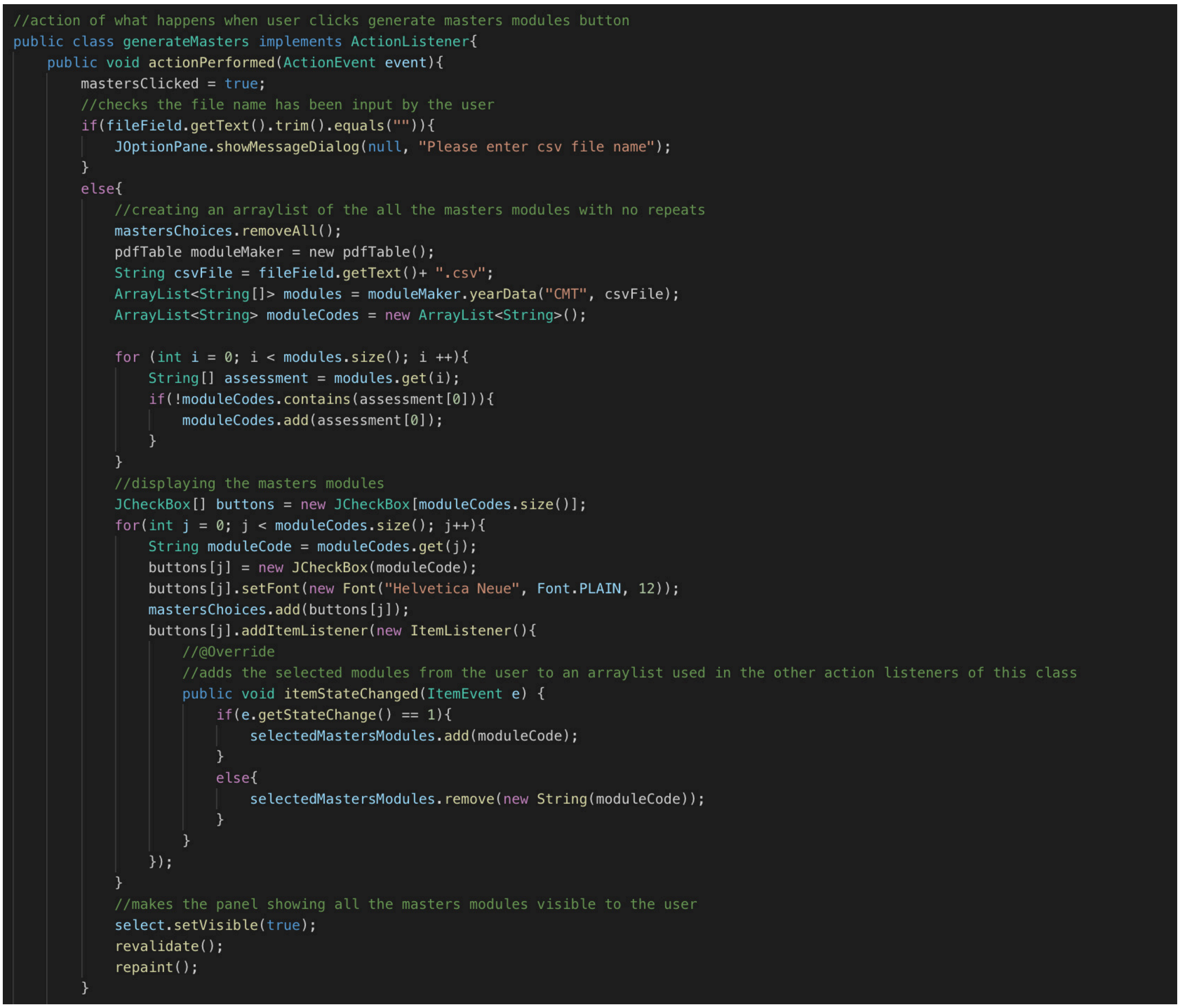
```
DocumentListener listener = new DocumentListener(){
    @Override
    public void removeUpdate(DocumentEvent e){
        change();
    }
    @Override
    public void insertUpdate(DocumentEvent e){
        change();
    }
    public void changedUpdate(DocumentEvent e){
        change();
    }

    public void change(){
        boolean canEnable = true;
        for (JTextField tf : fields){
            if (tf.getText().isEmpty()){
                canEnable = false;
            }
        }
        timetableFieldFilled = canEnable;
    }
};

DocumentListener changeListener = new DocumentListener(){
    @Override
    public void removeUpdate(DocumentEvent e){
        change();
    }
    @Override
    public void insertUpdate(DocumentEvent e){
        change();
    }
    public void changedUpdate(DocumentEvent e){
        change();
    }

    public void change(){
        boolean canEnable = true;
        for (JTextField tf : changeFields){
            if (tf.getText().isEmpty()){
                canEnable = false;
            }
        }
        changeFilled = canEnable;
    }
};
```

Figure 4.3

The listeners themselves both follow the same format when being defined however differ in what variables used elsewhere in the class they control. As mentioned, these listeners are defined to keep track of any insertions, deletions and updates in a required text field. In both listeners a Boolean value called canEnable is used to keep track of whether or not all of the relevant text fields have been filled in. Each time an insertion, deletion or update occurs, all the text fields that have the listener attached will be checked to see if it is empty. If any of them are empty then this value is set to false, otherwise it is true.

This canEnable value is then used to set another Boolean variable that determines if the functionality that requires the text fields with the relevant listener attached, i.e. creating all year groups tables for 'listener' and changing an assessment and making an updated timetable for 'changeListener', can be run.

Within the GUI of this system there are 3 buttons, each of which have their own action listener attached. Each action listener is defined by a class containing a method called actionPerformed that determines what happens when an event activates that specific action listener.



```java
//action of what happens when user clicks generate masters modules button
public class generateMasters implements ActionListener{
    public void actionPerformed(ActionEvent event){
        mastersClicked = true;
        //checks the file name has been input by the user
        if(fileField.getText().trim().equals("")){
            JOptionPane.showMessageDialog(null, "Please enter csv file name");
        }
        else{
            //creating an arraylist of the all the masters modules with no repeats
            mastersChoices.removeAll();
            pdfTable moduleMaker = new pdfTable();
            String csvFile = fileField.getText()+ ".csv";
            ArrayList<String[]> modules = moduleMaker.yearData("CMT", csvFile);
            ArrayList<String> moduleCodes = new ArrayList<String>();

            for (int i = 0; i < modules.size(); i ++){
                String[] assessment = modules.get(i);
                if(!moduleCodes.contains(assessment[0])){
                    moduleCodes.add(assessment[0]);
                }
            }
            //displaying the masters modules
            JCheckBox[] buttons = new JCheckBox[moduleCodes.size()];
            for(int j = 0; j < moduleCodes.size(); j++){
                String moduleCode = moduleCodes.get(j);
                buttons[j] = new JCheckBox(moduleCode);
                buttons[j].setFont(new Font("Helvetica Neue", Font.PLAIN, 12));
                mastersChoices.add(buttons[j]);
                buttons[j].addItemListener(new ItemListener(){
                    //@Override
                    //adds the selected modules from the user to an arraylist used in the other action listeners of this class
                    public void itemStateChanged(ItemEvent e) {
                        if(e.getStateChange() == 1){
                            selectedMastersModules.add(moduleCode);
                        }
                        else{
                            selectedMastersModules.remove(new String(moduleCode));
                        }
                    }
                });
            }
            //makes the panel showing all the masters modules visible to the user
            select.setVisible(true);
            revalidate();
            repaint();
        }
```

Figure 4.4

One of the functionalities of the system is allowing the user to choose which master's modules they wish to be included in the master's timetable which requires all of the master's modules to be displayed to the user. Since this feature is optional and not necessary to be completed for the timetables to be made, the system is designed so that the modules are only shown to the user once they click the generate master's modules button and is completed via the code shown in figure 4.4.

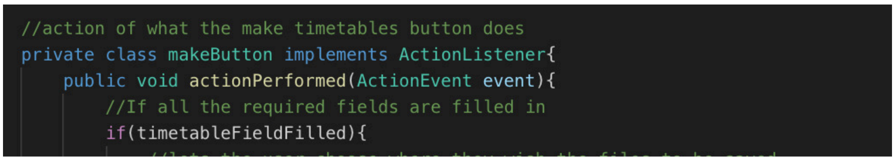
Before the master's modules are added to the GUI it will first set a Boolean variable called mastersClicked to true so that when it comes to making the timetables it knows whether to include all master's modules or just the ones chosen by the user. Following this it will then check to make sure the user has input the csv file name that it will collect the master's module codes from. If the user has not, then a pop-up dialog box will alert them to this and ask the user to complete this text field in order to carry out this functionality.

If the csv file name is present, then it will go on to display all the master's module codes to the user in the GUI as check boxes. Since the module codes are read in from the csv file using the master's code of 'CMT', there is the possibility of repeated module codes for modules that have more than one assessment. This problem is solved by creating a second array list (moduleCodes)that will eventually store all the master's module codes with no repeats. This is achieved by looping through all of the master's assessments and checking if the module code is in the second array list already. If it is not, then it is added, otherwise the next master's assessment is checked.

Now that there is a list of module codes with no repeats, the check boxes for these modules can be created and added to the GUI. Each module code is represented with a checkbox that has an item listener attached to it. This item listener for each checkbox indicates if it has been selected or deselected and allows the system to keep an up to date list of which modules the user has chosen to be included in the timetable. The addition of the checkboxes also means that the GUI can now set a previously hidden label explaining what to do with the checkboxes to be visible.

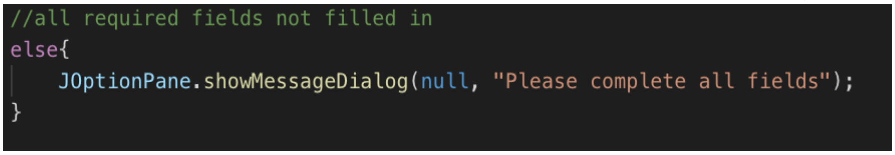
In order to make sure the GUI is up to date and that the user can now see these module checkboxes once they click the generate master's modules button, the GUI is revalidated and repainted.

The two remaining buttons in the GUI are responsible for the actual creation of timetables and are therefore very similar. The one that had the action performed method implemented first out of the two was the button that when clicked makes the timetable pdfs for all year groups.



```
//action of what the make timetables button does
private class makeButton implements ActionListener{
    public void actionPerformed(ActionEvent event){
        //If all the required fields are filled in
        if(timetableFieldFilled){
```

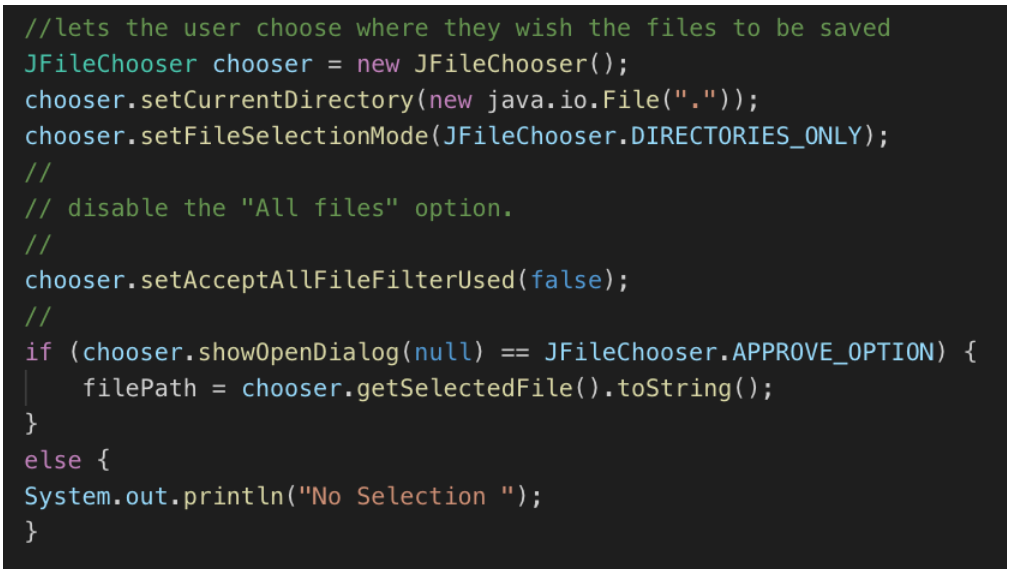
Figure 4.5



```
//all required fields not filled in
else{
    JOptionPane.showMessageDialog(null, "Please complete all fields");
}
```

Figure 4.6

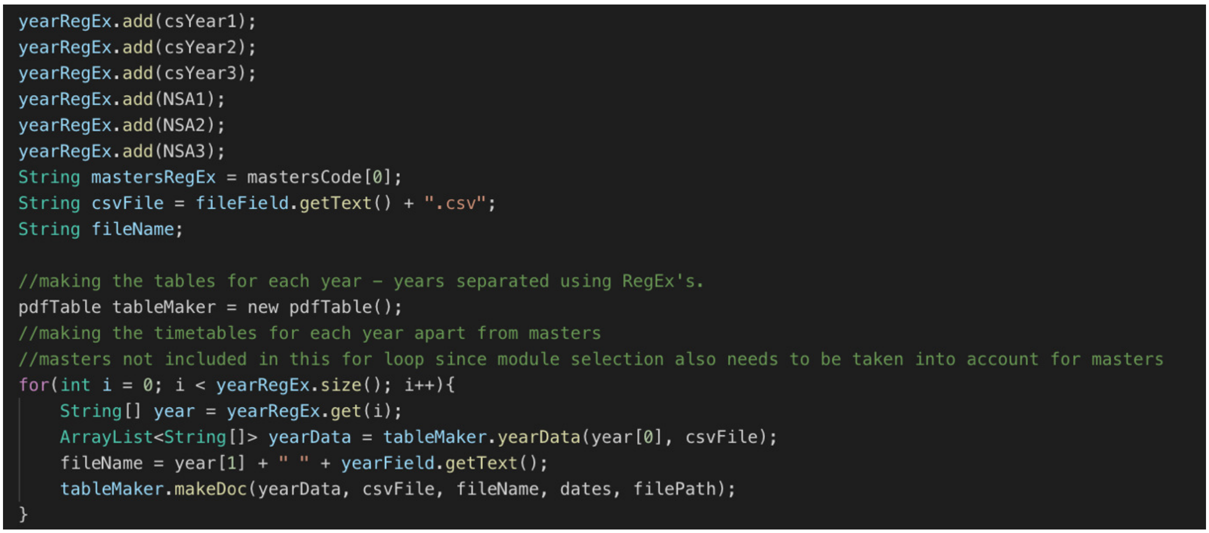
To avoid any errors that may occur from incomplete text fields it will first ensure that all the required text fields have been filled in. This is done by checking the Boolean variable that is constantly being updated in the document listener attached to the text fields required for this function (the one called listener). In the case where this is not true, the system will show a pop-up dialog box reminding them to complete all the text fields before trying to make the timetables.



```
//lets the user choose where they wish the files to be saved
JFileChooser chooser = new JFileChooser();
chooser.setCurrentDirectory(new java.io.File("."));
chooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
//
// disable the "All files" option.
//
chooser.setAcceptAllFileFilterUsed(false);
//
if (chooser.showOpenDialog(null) == JFileChooser.APPROVE_OPTION) {
    filePath = chooser.getSelectedFile().toString();
}
else {
System.out.println("No Selection ");
}
```

Figure 4.7

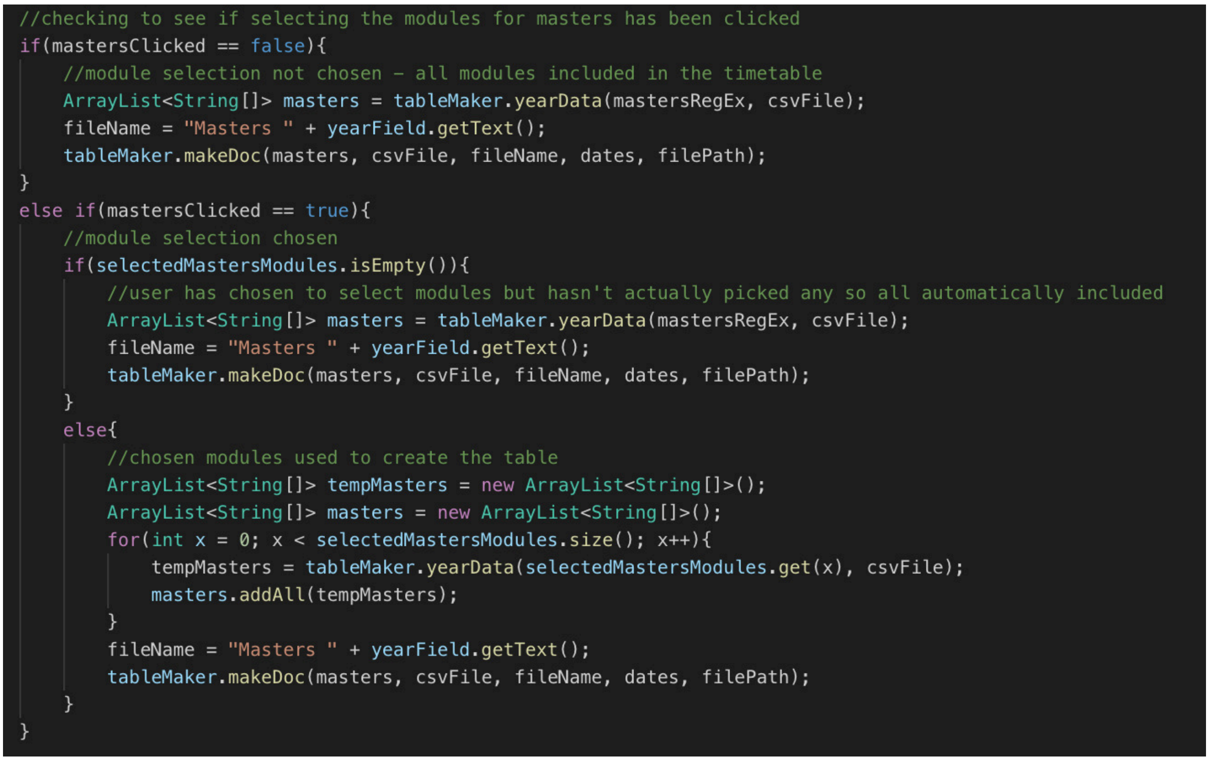
A mechanism within java and swing that I had not used prior to this project is the JFileChooser package. This package is how the user is able to be presented with a pop-up screen that allows them to navigate to the directory they wish to save the timetable pdfs in, and once they have selected a directory the file path to it is saved as a string to be used as an argument later when calling the method that will create the pdfs.



```
yearRegEx.add(csYear1);
yearRegEx.add(csYear2);
yearRegEx.add(csYear3);
yearRegEx.add(NSA1);
yearRegEx.add(NSA2);
yearRegEx.add(NSA3);
String mastersRegEx = mastersCode[0];
String csvFile = fileField.getText() + ".csv";
String fileName;

//making the tables for each year - years separated using RegEx's.
pdfTable tableMaker = new pdfTable();
//making the timetables for each year apart from masters
//masters not included in this for loop since module selection also needs to be taken into account for masters
for(int i = 0; i < yearRegEx.size(); i++){
    String[] year = yearRegEx.get(i);
    ArrayList<String[]> yearData = tableMaker.yearData(year[0], csvFile);
    fileName = year[1] + " " + yearField.getText();
    tableMaker.makeDoc(yearData, csvFile, fileName, dates, filePath);
}
```

Figure 4.8

The creation of the timetable pdfs themselves are handled by calling methods from within the pdfTable.java class. Each year has its pdf created by looping through a list of all the year groups names and year codes and creating an array list of that year's assessment data and then using this in the method that makes the timetable pdf. However, the master's year is not included in this list.



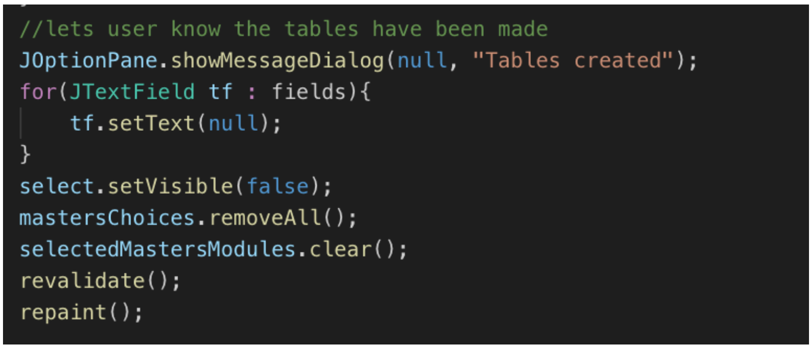
```
//checking to see if selecting the modules for masters has been clicked
if(mastersClicked == false){
    //module selection not chosen - all modules included in the timetable
    ArrayList<String[]> masters = tableMaker.yearData(mastersRegEx, csvFile);
    fileName = "Masters " + yearField.getText();
    tableMaker.makeDoc(masters, csvFile, fileName, dates, filePath);
}
else if(mastersClicked == true){
    //module selection chosen
    if(selectedMastersModules.isEmpty()){
        //user has chosen to select modules but hasn't actually picked any so all automatically included
        ArrayList<String[]> masters = tableMaker.yearData(mastersRegEx, csvFile);
        fileName = "Masters " + yearField.getText();
        tableMaker.makeDoc(masters, csvFile, fileName, dates, filePath);
    }
    else{
        //chosen modules used to create the table
        ArrayList<String[]> tempMasters = new ArrayList<String[]>();
        ArrayList<String[]> masters = new ArrayList<String[]>();
        for(int x = 0; x < selectedMastersModules.size(); x++){
            tempMasters = tableMaker.yearData(selectedMastersModules.get(x), csvFile);
            masters.addAll(tempMasters);
        }
        fileName = "Masters " + yearField.getText();
        tableMaker.makeDoc(masters, csvFile, fileName, dates, filePath);
    }
}
```

Figure 4.9

Since the master's year has the option of restricting which modules are included in the table, the creation of the pdf for it is handled differently. Firstly, it will check to see if the user has chosen to restrict which modules are used to create the timetable. This is done by checking the Boolean value that was set when the generate master's modules button is clicked. If the user has not clicked it then the master's module is made the same way as the other years with all of the year's modules being included in the timetable.

If the master's button has been clicked however the calling of the method to create the timetable pdf is altered. Despite the master's modules button being clicked, there is still the possibility of the user changing their minds and not actually wanting to restrict which modules are to be included in the timetable. If this is the case, then the array list containing all the selected module code check boxes will be empty and the method to create the timetable pdf will again be called as before. If the list is found not to be empty, then the system makes the assumption the user has chosen certain modules to be used in the creation of the timetable pdf. Whilst the same method is still called for the actual creation of the timetable pdf, the assessment data for the year must be altered to contain only the modules selected by the user.

This is achieved in a similar fashion to generating the list of master's module codes with no repeats by creating a second array list that will eventually contain only the assessments for modules chosen by the user. All of the assessments for the master's year are still taken from the csv file but rather than immediately using this in the timetable pdf creator method the second array list that is created by gathering only the assessments for modules that match those selected by the user is used.

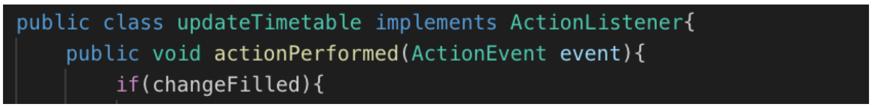


```
//lets user know the tables have been made
JOptionPane.showMessageDialog(null, "Tables created");
for(JTextField tf : fields){
    tf.setText(null);
}
select.setVisible(false);
mastersChoices.removeAll();
selectedMastersModules.clear();
revalidate();
repaint();
```

Figure 4.10

Having made all the timetable pdfs, the system will then let the user know it has completed this task and will reset the GUI.

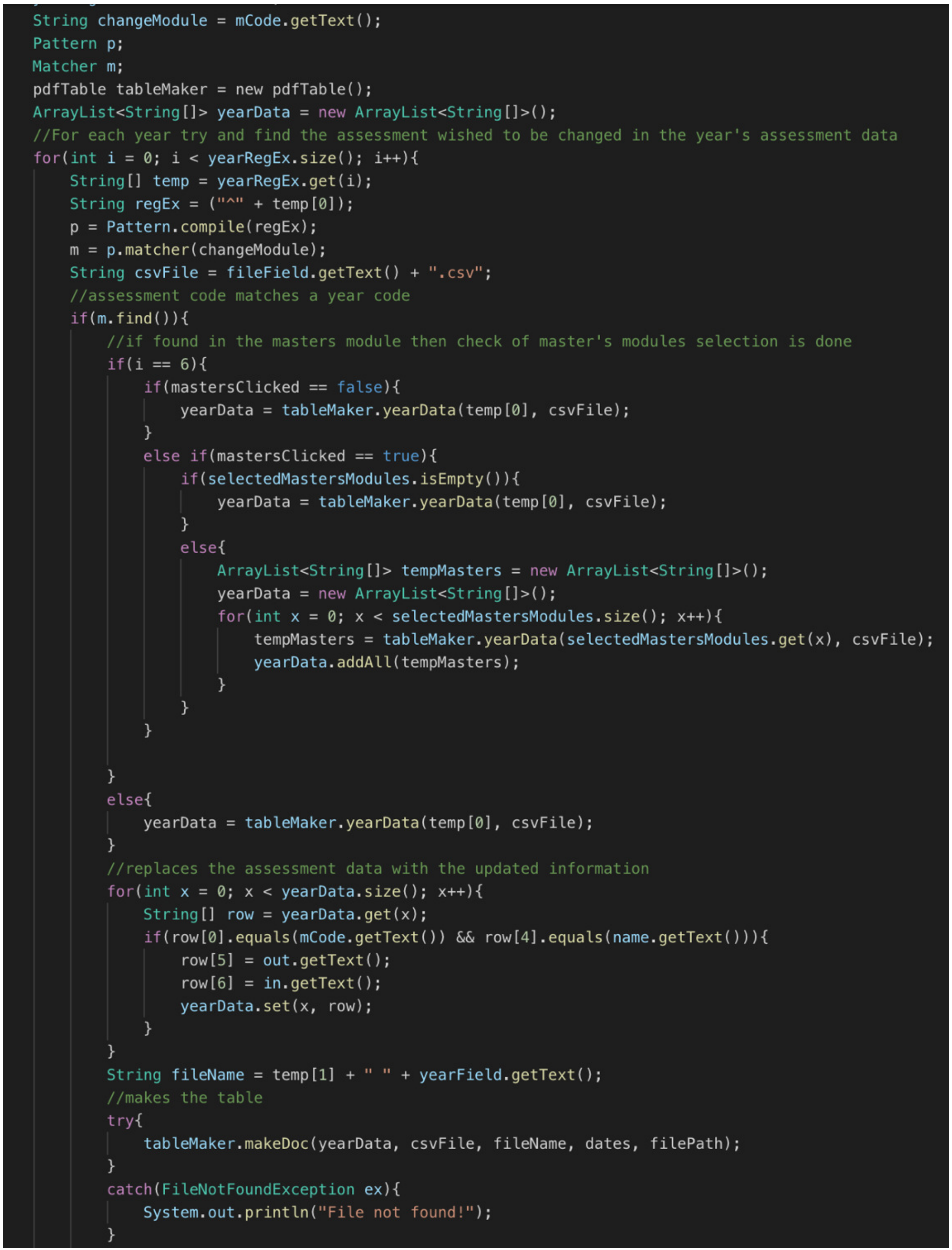
Upon completion of the implementation for the make timetables button, the code for the action listener of the button to update a timetable was then written. Since it follows the same process and is therefore very similar, I will not be discussing all of it again, but instead the areas in which it differs.



```
public class updateTimetable implements ActionListener{
    public void actionPerformed(ActionEvent event){
        if(changeFilled){
```

Figure 4.11

Once again, a Boolean variable is checked to ensure that all the required text field for this function are filled in. However, this is not the same variable as before but is instead the Boolean variable that is constantly being updated in the changeListener document listener.



```
String changeModule = mCode.getText();
Pattern p;
Matcher m;
pdfTable tableMaker = new pdfTable();
ArrayList<String[]> yearData = new ArrayList<String[]>();
//For each year try and find the assessment wished to be changed in the year's assessment data
for(int i = 0; i < yearRegEx.size(); i++){
    String[] temp = yearRegEx.get(i);
    String regEx = ("^" + temp[0]);
    p = Pattern.compile(regEx);
    m = p.matcher(changeModule);
    String csvFile = fileField.getText() + ".csv";
    //assessment code matches a year code
    if(m.find()){
        //if found in the masters module then check of master's modules selection is done
        if(i == 6){
            if(mastersClicked == false){
                yearData = tableMaker.yearData(temp[0], csvFile);
            }
            else if(mastersClicked == true){
                if(selectedMastersModules.isEmpty()){
                    yearData = tableMaker.yearData(temp[0], csvFile);
                }
                else{
                    ArrayList<String[]> tempMasters = new ArrayList<String[]>();
                    yearData = new ArrayList<String[]>();
                    for(int x = 0; x < selectedMastersModules.size(); x++){
                        tempMasters = tableMaker.yearData(selectedMastersModules.get(x), csvFile);
                        yearData.addAll(tempMasters);
                    }
                }
            }

        }
        else{
            yearData = tableMaker.yearData(temp[0], csvFile);
        }
        //replaces the assessment data with the updated information
        for(int x = 0; x < yearData.size(); x++){
            String[] row = yearData.get(x);
            if(row[0].equals(mCode.getText()) && row[4].equals(name.getText())){
                row[5] = out.getText();
                row[6] = in.getText();
                yearData.set(x, row);
            }
        }
        String fileName = temp[1] + " " + yearField.getText();
        //makes the table
        try{
            tableMaker.makeDoc(yearData, csvFile, fileName, dates, filePath);
        }
        catch(FileNotFoundException ex){
            System.out.println("File not found!");
        }
```

Figure 4.12

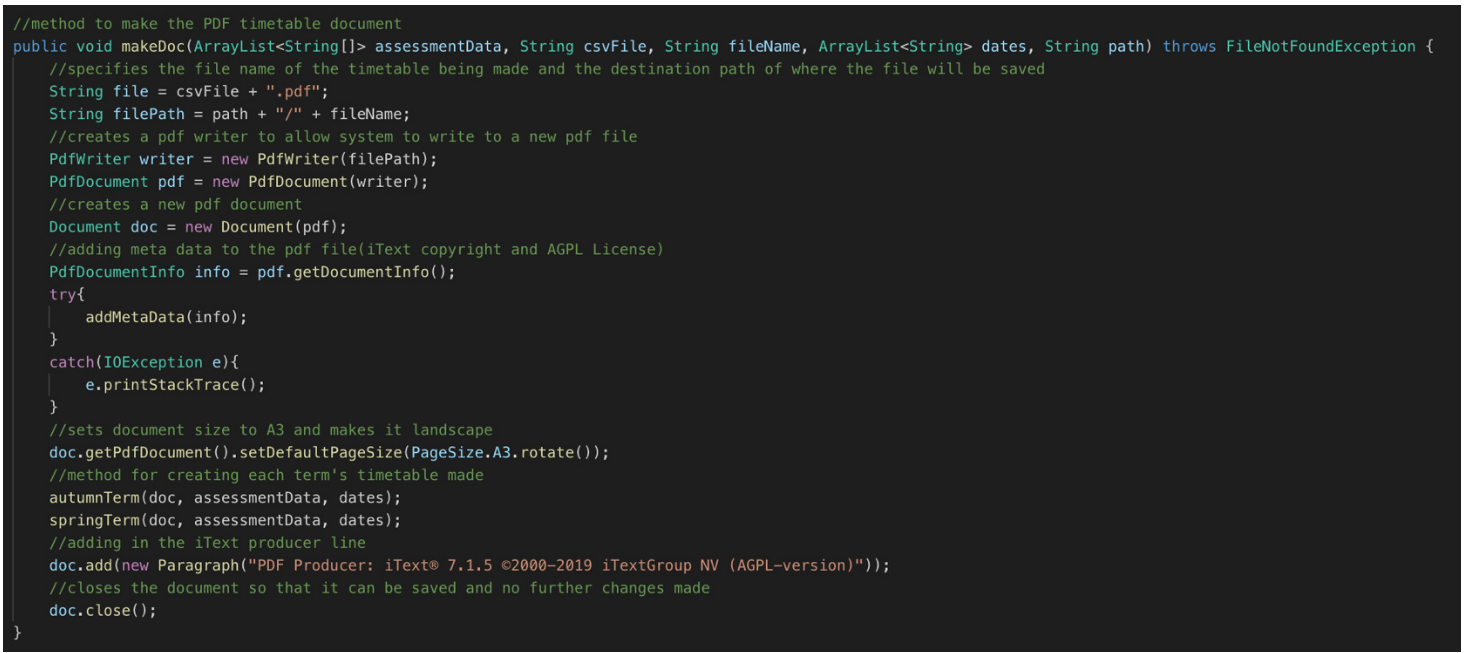
Where this action listener differs is in the calling of the method that creates the timetable pdf. Since the pdfs do not need to be made for every year group, just the one having an assessment updated, this method must first determine which year group it is making an updated timetable for. The year group is determined by using a pattern and matcher instance to check the module code of the assessment being changed against each year group's code. If a match to a year group is found, then the list containing the assessment data for that year is created and the assessment to be updated is found and the dates changed before calling the method to make the timetable pdf.

If this update of an assessment is within the master's year then the user also has the option of once again choosing which of the master's modules are to be included in the timetable. If the year that the changing assessment's module code is the master's year then the same process of checking mastersClicked in the make timetables button action performed method is used.

Once the correct year group's assessment data has been gathered, the changing assessment is updated in the list to reflect the user's input. With the assessment data list now updated, the method to create the timetable pdf is called.

pdfTable.java

This class is responsible for the creation of the pdfs themselves. It contains the methods called in the GUI class to create the timetables for each year group as well as additional methods that are called internally to help aid the process of making the pdfs. In order to try and keep this report concise, only the code which implements features of the system will be discussed.

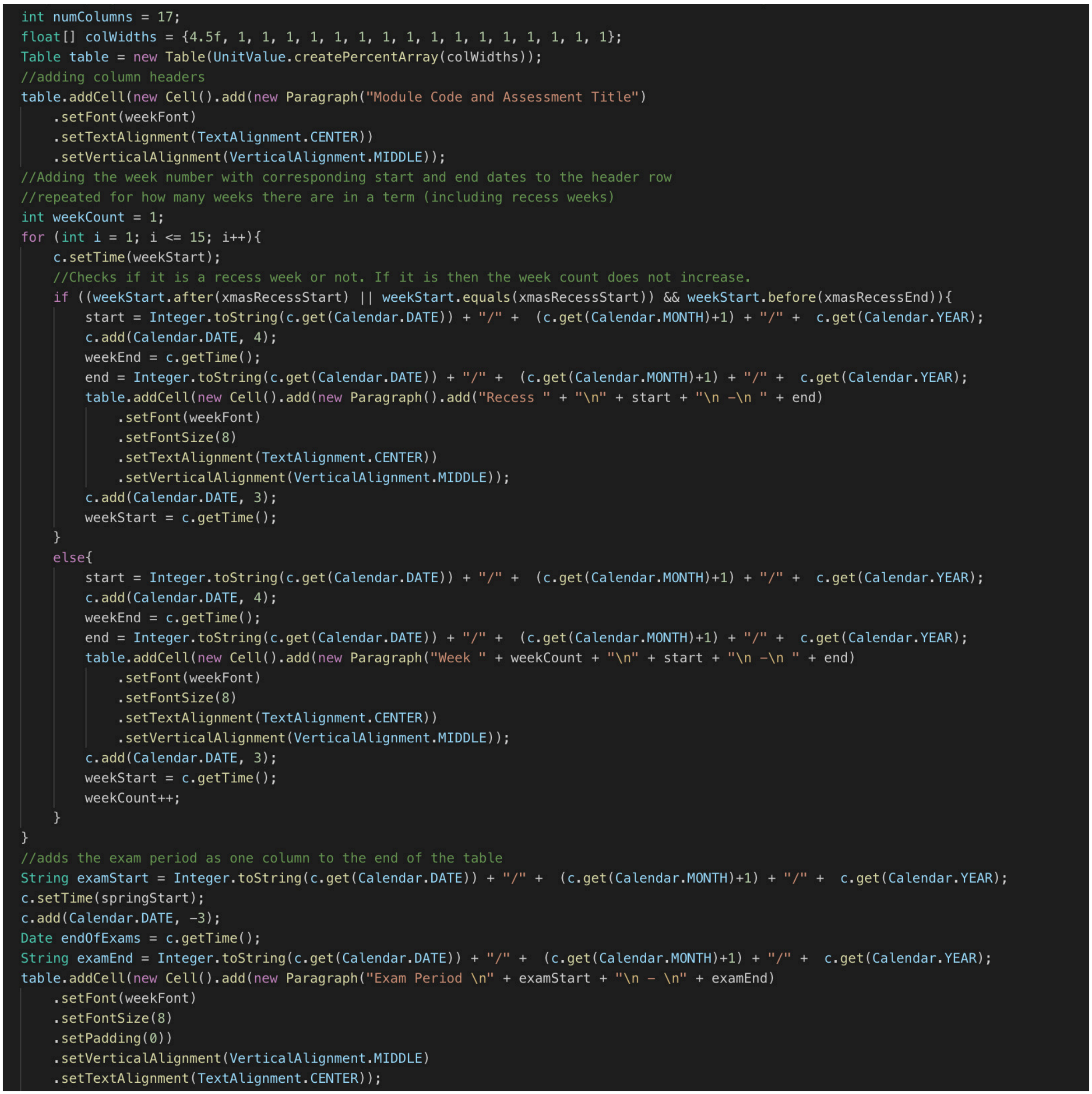


```
//method to make the PDF timetable document
public void makeDoc(ArrayList<String[]> assessmentData, String csvFile, String fileName, ArrayList<String> dates, String path) throws FileNotFoundException {
    //specifies the file name of the timetable being made and the destination path of where the file will be saved
    String file = csvFile + ".pdf";
    String filePath = path + "/" + fileName;
    //creates a pdf writer to allow system to write to a new pdf file
    PdfWriter writer = new PdfWriter(filePath);
    PdfDocument pdf = new PdfDocument(writer);
    //creates a new pdf document
    Document doc = new Document(pdf);
    //adding meta data to the pdf file(iText copyright and AGPL License)
    PdfDocumentInfo info = pdf.getDocumentInfo();
    try{
        addMetaData(info);
    }
    catch(IOException e){
        e.printStackTrace();
    }
    //sets document size to A3 and makes it landscape
    doc.getPdfDocument().setDefaultPageSize(PageSize.A3.rotate());
    //method for creating each term's timetable made
    autumnTerm(doc, assessmentData, dates);
    springTerm(doc, assessmentData, dates);
    //adding in the iText producer line
    doc.add(new Paragraph("PDF Producer: iText® 7.1.5 ©2000-2019 iTextGroup NV (AGPL-version)"));
    //closes the document so that it can be saved and no further changes made
    doc.close();
}
```

Figure 4.13

The makeDoc method called in the GUI class to create the tables does not actually create the term assessment tables that populate the pdf but instead calls two other methods defined in this class that do. It does however create the document that these tables are added to as well as adding in all the required metadata as outlined in the iText AGPL license restrictions by calling the addMetaData method defined later on in this class. The assessment tables for each term are added to the pdf document by calling methods that create a single table for each term. Whilst both of the term methods are very similar in how they create the tables, there are still different methods for each term because the two terms differ in how they add cells to the tables and in the information required to make them.

Since both term methods are similar, I will be explaining the autumn term method more in depth and only explaining how the spring term method differs from the this to avoid repeated explanations.



```
int numColumns = 17;
float[] colWidths = {4.5f, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1};
Table table = new Table(UnitValue.createPercentArray(colWidths));
//adding column headers
table.addCell(new Cell().add(new Paragraph("Module Code and Assessment Title")
    .setFont(weekFont)
    .setTextAlignment(TextAlignment.CENTER))
    .setVerticalAlignment(VerticalAlignment.MIDDLE));
//Adding the week number with corresponding start and end dates to the header row
//repeated for how many weeks there are in a term (including recess weeks)
int weekCount = 1;
for (int i = 1; i <= 15; i++){
    c.setTime(weekStart);
    //Checks if it is a recess week or not. If it is then the week count does not increase.
    if ((weekStart.after(xmasRecessStart) || weekStart.equals(xmasRecessStart)) && weekStart.before(xmasRecessEnd)){
        start = Integer.toString(c.get(Calendar.DATE)) + "/" +  (c.get(Calendar.MONTH)+1) + "/" +  c.get(Calendar.YEAR);
        c.add(Calendar.DATE, 4);
        weekEnd = c.getTime();
        end = Integer.toString(c.get(Calendar.DATE)) + "/" +  (c.get(Calendar.MONTH)+1) + "/" +  c.get(Calendar.YEAR);
        table.addCell(new Cell().add(new Paragraph().add("Recess " + "\n" + start + "\n -\n " + end)
            .setFont(weekFont)
            .setFontSize(8)
            .setTextAlignment(TextAlignment.CENTER))
            .setVerticalAlignment(VerticalAlignment.MIDDLE));
        c.add(Calendar.DATE, 3);
        weekStart = c.getTime();
    }
    else{
        start = Integer.toString(c.get(Calendar.DATE)) + "/" +  (c.get(Calendar.MONTH)+1) + "/" +  c.get(Calendar.YEAR);
        c.add(Calendar.DATE, 4);
        weekEnd = c.getTime();
        end = Integer.toString(c.get(Calendar.DATE)) + "/" +  (c.get(Calendar.MONTH)+1) + "/" +  c.get(Calendar.YEAR);
        table.addCell(new Cell().add(new Paragraph("Week " + weekCount + "\n" + start + "\n -\n " + end)
            .setFont(weekFont)
            .setFontSize(8)
            .setTextAlignment(TextAlignment.CENTER))
            .setVerticalAlignment(VerticalAlignment.MIDDLE));
        c.add(Calendar.DATE, 3);
        weekStart = c.getTime();
        weekCount++;
    }
}
//adds the exam period as one column to the end of the table
String examStart = Integer.toString(c.get(Calendar.DATE)) + "/" +  (c.get(Calendar.MONTH)+1) + "/" +  c.get(Calendar.YEAR);
c.setTime(springStart);
c.add(Calendar.DATE, -3);
Date endOfExams = c.getTime();
String examEnd = Integer.toString(c.get(Calendar.DATE)) + "/" +  (c.get(Calendar.MONTH)+1) + "/" +  c.get(Calendar.YEAR);
table.addCell(new Cell().add(new Paragraph("Exam Period \n" + examStart + "\n - \n" + examEnd)
    .setFont(weekFont)
    .setFontSize(8)
    .setPadding(0))
    .setVerticalAlignment(VerticalAlignment.MIDDLE)
    .setTextAlignment(TextAlignment.CENTER));
```

Figure 4.14

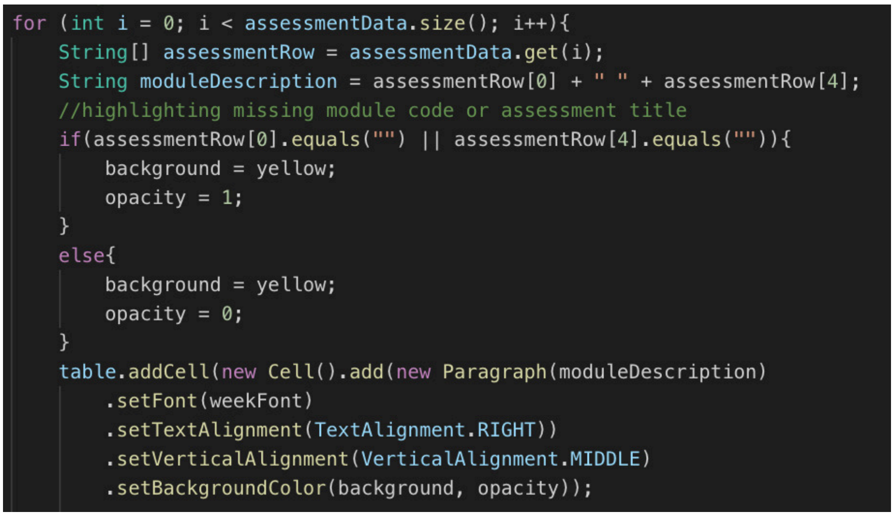
Having initialised all the variables that will be needed throughout the autumnTerm method including the relevant term dates and fonts that will be used in the table, it will then go on to create the table that will be populated with the autumn term assessment data. This table is made to have 17 columns as designed in the Specification and Design section of this report. There is a column for an assessment's module code and assessment title plus one for each term week (12) plus one for each week of a term's recess (3) plus one column to represent the exam period at the end of a term.

Since iText creates tables by adding cells one by one from left to right until the number of columns in that row is equal to what it has been set to, I had to ensure that each row was adding the correct number of cells to each row. This is because if too many cells are added to a row then iText will automatically add the additional cells to a new row, and then when the intended start of the next row is being added, it is instead however many additional cells you had in the previous row along in its row. This then has a knock-on effect as each row will continue to displace the next row and cells will not line up with the columns as intended. This is also the case if not enough cells are added to a row.

To ensure I had the correct number of cells being added to the header row, which is the first row added to the table, I used a for loop that is repeated a set number of times. However, before this is started the column header for an assessment's module code and title is added as the first cell to the table. The loop then adds in the term weeks and recess weeks for the term, followed by the addition of one more single cell for the exam period at the end of this term. Since the for loop adds in cells for each week number and recess week it had to be able to determine which cell it was adding in so the correct text was added to it. This is done by having two date variables that keep track of the start and end date of the working week (i.e. the date of Monday as the start and Friday as the end) currently being added to the table. These two date variables use a calendar instance that is updated each time a cell representing a week is added to the table and compares these dates to the Christmas holiday dates input by the user and formatted earlier on in the method to see if it should be a recess or term week added to the table.

These start and end dates of the week are also used as contents within the week cells so that when users are looking at the table once it has been created, they are easily able to see which week number corresponds to which dates. The exam period cell added after this for loop also uses the start date variable since it is assumed the exam period immediately follows week 12 in a term and due to the way the dates are updated, at the end of the for loop the start date will be equal to the date the exam period starts. The end date of the exam period is assumed to be the Friday before the start of the spring term, which is a required input from the user, so the exam end date is calculated using the user's input.

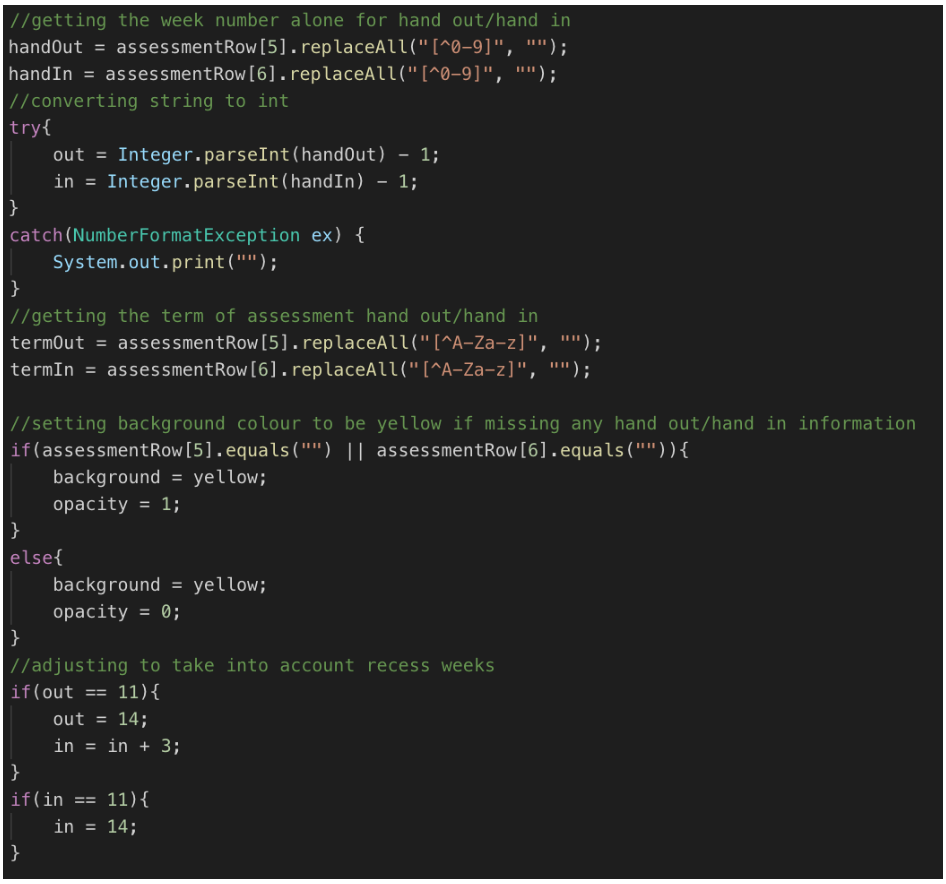
Following the addition of the header row, the assessments for the year group this table is being made for are then added in one by one as a row. The addition of assessments is done by using a loop that is set to iterate as many times as there are assessments in the list of assessments sent into this method.



```
for (int i = 0; i < assessmentData.size(); i++){
    String[] assessmentRow = assessmentData.get(i);
    String moduleDescription = assessmentRow[0] + " " + assessmentRow[4];
    //highlighting missing module code or assessment title
    if(assessmentRow[0].equals("") || assessmentRow[4].equals("")){
        background = yellow;
        opacity = 1;
    }
    else{
        background = yellow;
        opacity = 0;
    }
    table.addCell(new Cell().add(new Paragraph(moduleDescription)
        .setFont(weekFont)
        .setTextAlignment(TextAlignment.RIGHT))
        .setVerticalAlignment(VerticalAlignment.MIDDLE)
        .setBackgroundColor(background, opacity));
```

Figure 4.15

Each assessment row starts with its module code and title being added as a cell. If either or both of these are missing from the csv file, the background colour of this cell will be set to yellow at full opacity so that when the user looks at the timetable pdf it knows the assessment is missing some information. If both are present then the background colour is still set to yellow, but at 0 opacity so that it will not show up in the table.



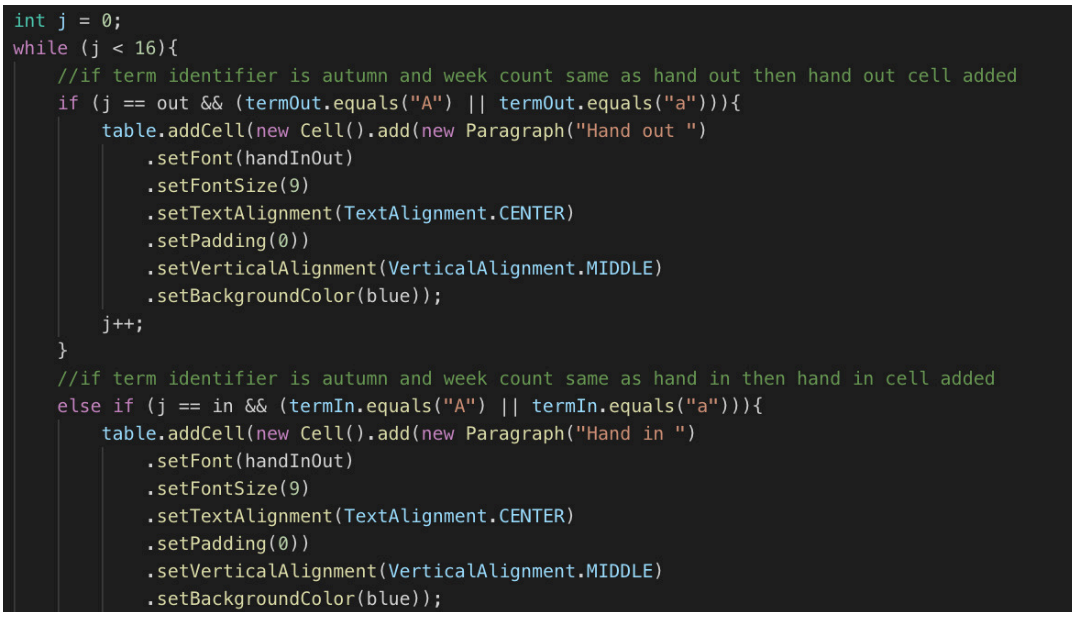
```java
//getting the week number alone for hand out/hand in
handOut = assessmentRow[5].replaceAll("[^0-9]", "");
handIn = assessmentRow[6].replaceAll("[^0-9]", "");
//converting string to int
try{
    out = Integer.parseInt(handOut) - 1;
    in = Integer.parseInt(handIn) - 1;
}
catch(NumberFormatException ex) {
    System.out.print("");
}
//getting the term of assessment hand out/hand in
termOut = assessmentRow[5].replaceAll("[^A-Za-z]", "");
termIn = assessmentRow[6].replaceAll("[^A-Za-z]", "");

//setting background colour to be yellow if missing any hand out/hand in information
if(assessmentRow[5].equals("") || assessmentRow[6].equals("")){
    background = yellow;
    opacity = 1;
}
else{
    background = yellow;
    opacity = 0;
}
//adjusting to take into account recess weeks
if(out == 11){
    out = 14;
    in = in + 3;
}
if(in == 11){
    in = 14;
}
```

Figure 4.16

Once the assessment title and module code has been added to the row the cells for each week in the term can be added. In order to be able to determine later on in the method what type of cell needs to be added the hand out and hand in information for an assessment is processed and separated into the week number and term identifier for each date.

Again, if any of this information is missing the background colour for cells added in this row will be yellow at full opacity to alert the user to missing information within the data used to create the tables.

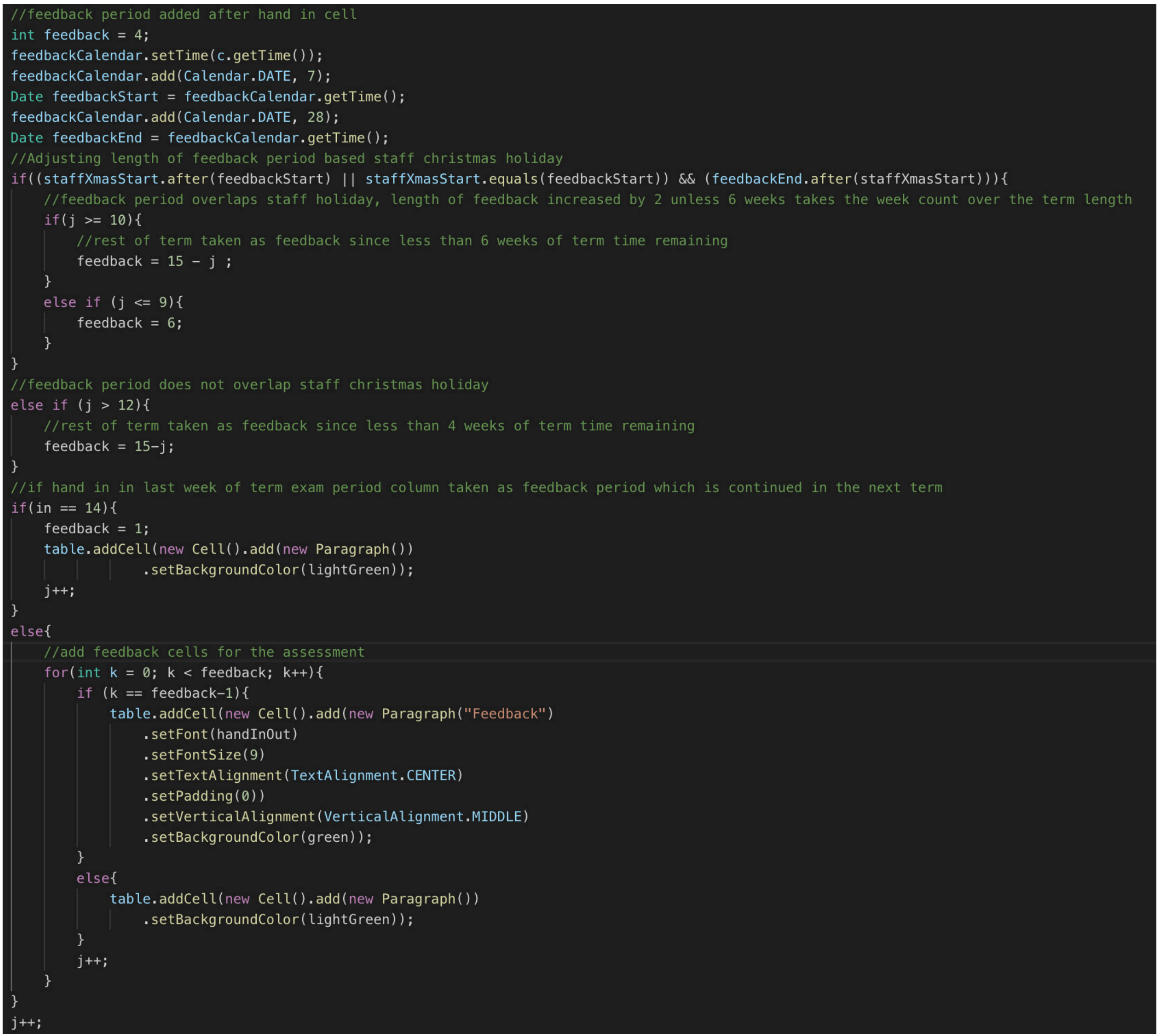
Because each assessment's weeks are added in cell by cell later on using a count starting from 0, the int variables representing the week number for hand out and hand in need to be adjusted to take into account recess weeks. The loop following the addition of the assessment title and module code is repeated 16 times and the count value is taken as the week number of the cell being added to the row. So, a count value of 0 refers to week 1 in a term, a count value of 1 refers to week 2 in a term etc. However, because there are 3 recess weeks following week 11 (which has the count value of 10), that do not count towards the term's week count, week 12 in a term will actually be added when the count is 14. With this in mind, if the hand out or hand in week number of an assessment is 12 it is altered to still be shown as this in the table.



```java
int j = 0;
while (j < 16){
    //if term identifier is autumn and week count same as hand out then hand out cell added
    if (j == out && (termOut.equals("A") || termOut.equals("a"))){
        table.addCell(new Cell().add(new Paragraph("Hand out ")
            .setFont(handInOut)
            .setFontSize(9)
            .setTextAlignment(TextAlignment.CENTER)
            .setPadding(0))
            .setVerticalAlignment(VerticalAlignment.MIDDLE)
            .setBackgroundColor(blue));
        j++;
    }
    //if term identifier is autumn and week count same as hand in then hand in cell added
    else if (j == in && (termIn.equals("A") || termIn.equals("a"))){
        table.addCell(new Cell().add(new Paragraph("Hand in ")
            .setFont(handInOut)
            .setFontSize(9)
            .setTextAlignment(TextAlignment.CENTER)
            .setPadding(0))
            .setVerticalAlignment(VerticalAlignment.MIDDLE)
            .setBackgroundColor(blue));
```

Figure 4.17

In order to determine what type of cell to add to the table a series of checks are done using if and else if statements. The first check is to see if it is a hand out cell that needs to be added. This is the case if the term identifier for the assessment being added in this row is autumn and the count of the loop is on is equal to the hand out week number. If this is not the case, then it goes on to check if it is a hand in cell that needs to be added. This follows the same principle of the hand out cell and a hand in cell is added if the term identifier is autumn and the count of the loop is equal to the assessment's hand in week number.



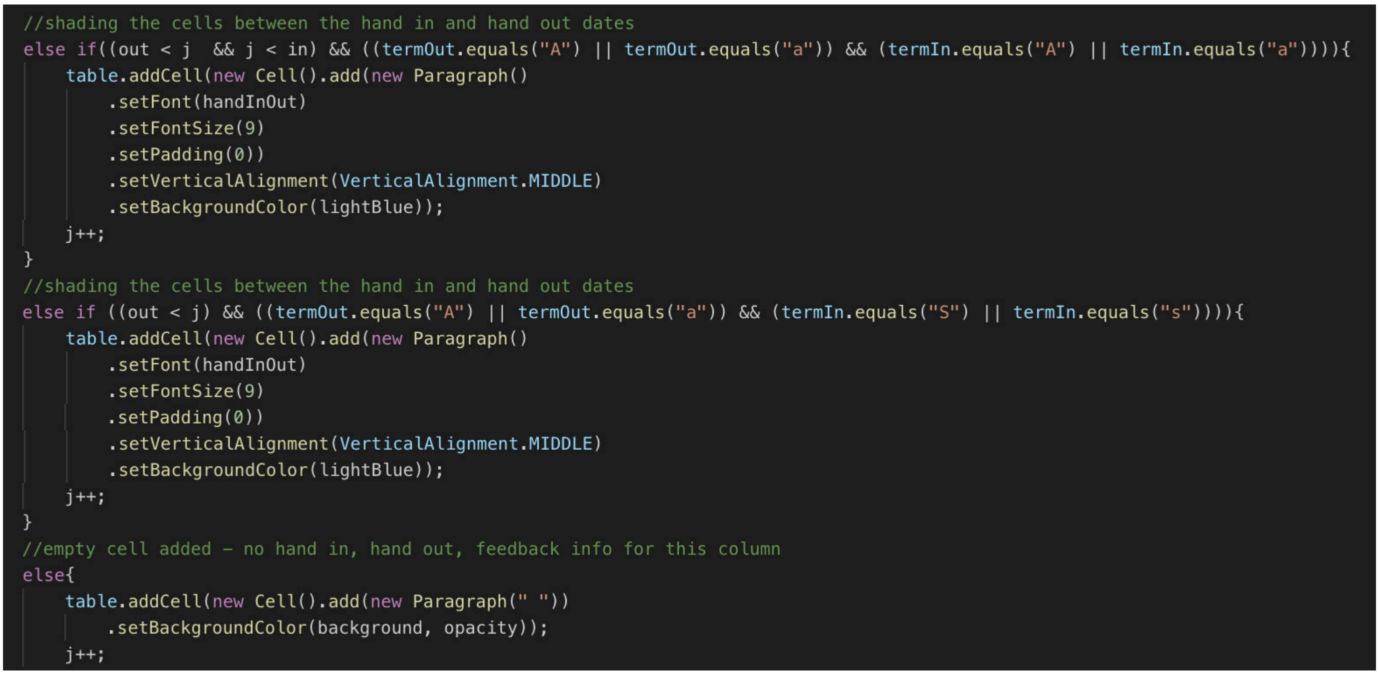
```java
//feedback period added after hand in cell
int feedback = 4;
feedbackCalendar.setTime(c.getTime());
feedbackCalendar.add(Calendar.DATE, 7);
Date feedbackStart = feedbackCalendar.getTime();
feedbackCalendar.add(Calendar.DATE, 28);
Date feedbackEnd = feedbackCalendar.getTime();
//Adjusting length of feedback period based staff christmas holiday
if((staffXmasStart.after(feedbackStart) || staffXmasStart.equals(feedbackStart)) && (feedbackEnd.after(staffXmasStart))){
    //feedback period overlaps staff holiday, length of feedback increased by 2 unless 6 weeks takes the week count over the term length
    if(j >= 10){
        //rest of term taken as feedback since less than 6 weeks of term time remaining
        feedback = 15 - j ;
    }
    else if (j <= 9){
        feedback = 6;
    }
}
//feedback period does not overlap staff christmas holiday
else if (j > 12){
    //rest of term taken as feedback since less than 4 weeks of term time remaining
    feedback = 15-j;
}
//if hand in in last week of term exam period column taken as feedback period which is continued in the next term
if(in == 14){
    feedback = 1;
    table.addCell(new Cell().add(new Paragraph())
                .setBackgroundColor(lightGreen));
    j++;
}
else{
    //add feedback cells for the assessment
    for(int k = 0; k < feedback; k++){
        if (k == feedback-1){
            table.addCell(new Cell().add(new Paragraph("Feedback")
                .setFont(handInOut)
                .setFontSize(9)
                .setTextAlignment(TextAlignment.CENTER)
                .setPadding(0))
                .setVerticalAlignment(VerticalAlignment.MIDDLE)
                .setBackgroundColor(green));
        }
        else{
            table.addCell(new Cell().add(new Paragraph())
                .setBackgroundColor(lightGreen));
        }
        j++;
    }
}
j++;
```

Figure 4.18

The addition of a hand in cell means that a feedback period must also be added to the table immediately following the hand in cell. Whilst this is typically 4 working weeks, so 4 cells, this can be altered in length if a staff holiday overlaps this feedback period. Using two calendar instances, one that keeps track of the start date for the week being added, and one that is used to calculate the start and end dates of a 4-week feedback period following the hand in date, the length of the feedback period is adjusted as necessary. Calculating the start and end date of the typical 4-week feedback period following a hand in provides the dates necessary to compare against the staff holiday dates to check for any overlap. If there is an overlap then the feedback period has an additional 2 weeks added to it, making it 6 weeks in length.

No matter the outcome of if the staff holiday overlaps the feedback period, there needs to be a check to see if there are enough cells remaining in the row to add the feedback period to ensure no additional cells are added to a row. If there are not enough cells remaining in the row, then the rest of the row is taken as the length of the feedback period. Apart from a hand in in week 12 which is handled separately and always set to 1 week for this term and continued in the next term's table, this does not affect the layout or shorten the feedback period length due to the assumption that hand in must be during term weeks and not recess weeks. This is because excluding week 12, the latest a hand in can be is week 11 which leaves 5 columns to be added after this, 3 recess week cells, a week 12 cell and a cell to represent the 2-week autumn exam period. Because this last cell represents 2 weeks, as long as the hand in for an assessment is not week 12 taking the remainder of the row as the feedback period when there are not enough remaining cells in a row will not shorten the length. The only potential issue with this solution is that is makes the feedback period for hand ins in week 10 and 11 appear to end at the same time. This is because the feedback period for hand ins in week 10 will end halfway through the exam period, i.e. halfway through the last cell in a row which is when feedback periods for hand ins in week 11 end. However, because it is not possible to only apply a background colour to half of a cell and time constraints not allowing the change of the exam period into 2 separate columns it has been left as this for now.

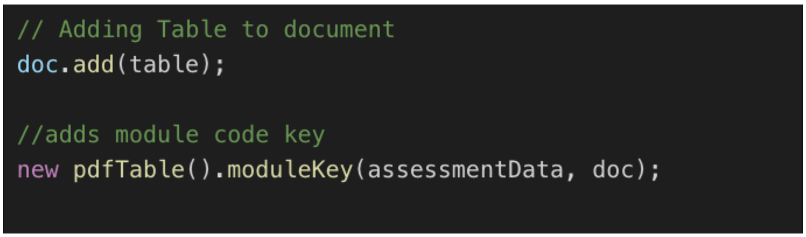
Since the feedback period immediately follows the hand in, the cells for the feedback period are added into the table within the same else if check for a hand in cell.



```
//shading the cells between the hand in and hand out dates
else if((out < j  && j < in) && ((termOut.equals("A") || termOut.equals("a")) && (termIn.equals("A") || termIn.equals("a")))){
    table.addCell(new Cell().add(new Paragraph()
        .setFont(handInOut)
        .setFontSize(9)
        .setPadding(0))
        .setVerticalAlignment(VerticalAlignment.MIDDLE)
        .setBackgroundColor(lightBlue));
    j++;
}
//shading the cells between the hand in and hand out dates
else if ((out < j) && ((termOut.equals("A") || termOut.equals("a")) && (termIn.equals("S") || termIn.equals("s")))){
    table.addCell(new Cell().add(new Paragraph()
        .setFont(handInOut)
        .setFontSize(9)
        .setPadding(0))
        .setVerticalAlignment(VerticalAlignment.MIDDLE)
        .setBackgroundColor(lightBlue));
    j++;
}
//empty cell added - no hand in, hand out, feedback info for this column
else{
    table.addCell(new Cell().add(new Paragraph(" "))
        .setBackgroundColor(background, opacity));
    j++;
```

Figure 4.19

If it has been neither a hand out or hand in cell that needs to be added, then the method will check to see if it is a shaded cell indicating the time frame given to compete an assessment that needs to be added. In the autumn term there are two cases where this are required. The first being when both the hand out and hand in term identifiers are autumn and the count of the loop is in-between the hand out week number and hand in week number. The second situation where this type of cell would need to be added in is when the loop count is greater than the hand out week number and the hand out term identifier is autumn but the hand in term identifier is spring. Failing these last checks, an empty filler cell with no contents is added to the table to ensure the correct number of cells is being added to each row.

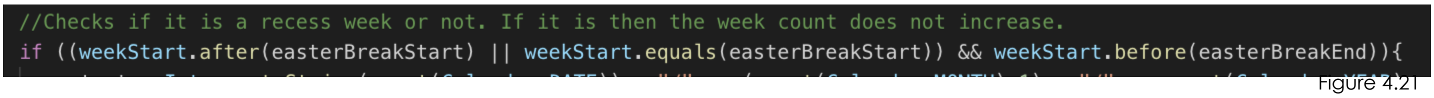


```
// Adding Table to document
doc.add(table);

//adds module code key
new pdfTable().moduleKey(assessmentData, doc);
```

Figure 4.20

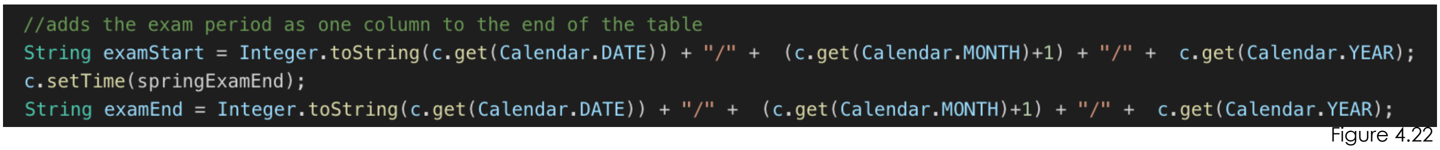
With the autumn term table complete it is then added to the pdf document along with a table that acts as the key for module codes.

As previously mentioned, the spring term method that creates the assessment table for the spring term is very similar so only the differences between the two will be explained. The set up and initial creation of the table with adding the header row in is the same with the only thing differing from the autumn term method being the dates used to calculate the recess weeks and exam period.



```
//Checks if it is a recess week or not. If it is then the week count does not increase.
if ((weekStart.after(easterBreakStart) || weekStart.equals(easterBreakStart)) && weekStart.before(easterBreakEnd)){
```

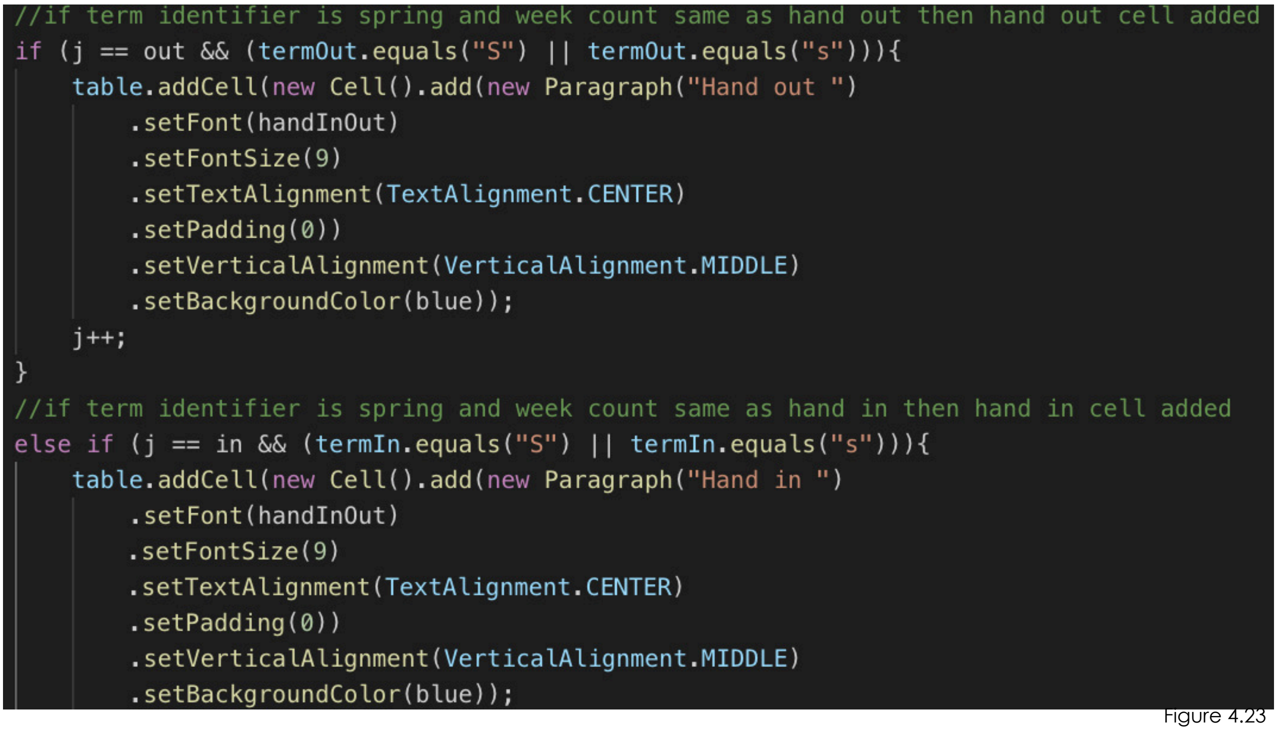
Figure 4.21



```
//adds the exam period as one column to the end of the table
String examStart = Integer.toString(c.get(Calendar.DATE)) + "/" +  (c.get(Calendar.MONTH)+1) + "/" +  c.get(Calendar.YEAR);
c.setTime(springExamEnd);
String examEnd = Integer.toString(c.get(Calendar.DATE)) + "/" +  (c.get(Calendar.MONTH)+1) + "/" +  c.get(Calendar.YEAR);
```

Figure 4.22

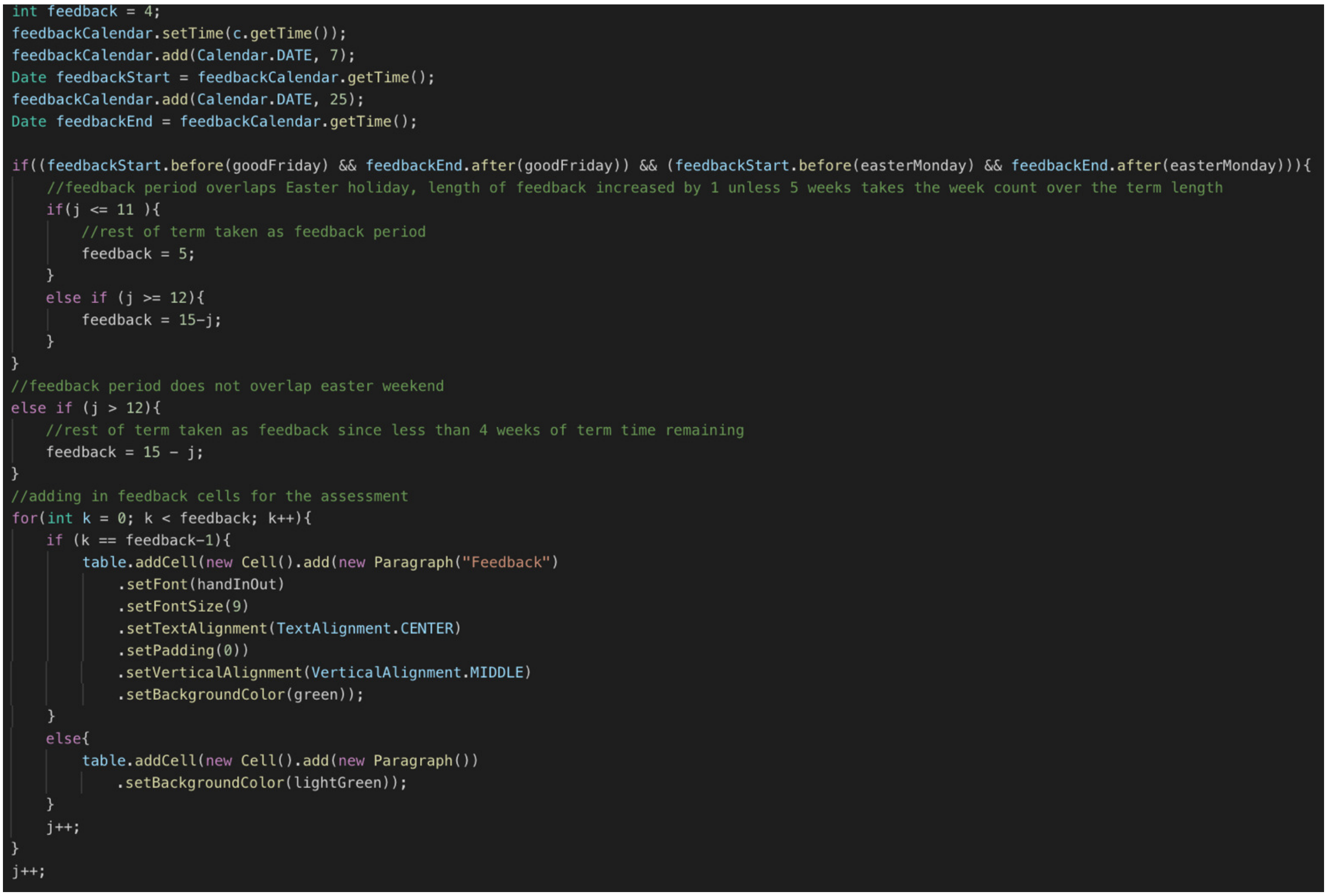
Where the spring method next differs, although only slightly, is in deciding what type of cell to add into the table for each assessment.



```
//if term identifier is spring and week count same as hand out then hand out cell added
if (j == out && (termOut.equals("S") || termOut.equals("s"))){
    table.addCell(new Cell().add(new Paragraph("Hand out ")
        .setFont(handInOut)
        .setFontSize(9)
        .setTextAlignment(TextAlignment.CENTER)
        .setPadding(0))
        .setVerticalAlignment(VerticalAlignment.MIDDLE)
        .setBackgroundColor(blue));
    j++;
}
//if term identifier is spring and week count same as hand in then hand in cell added
else if (j == in && (termIn.equals("S") || termIn.equals("s"))){
    table.addCell(new Cell().add(new Paragraph("Hand in ")
        .setFont(handInOut)
        .setFontSize(9)
        .setTextAlignment(TextAlignment.CENTER)
        .setPadding(0))
        .setVerticalAlignment(VerticalAlignment.MIDDLE)
        .setBackgroundColor(blue));
```

Figure 4.23

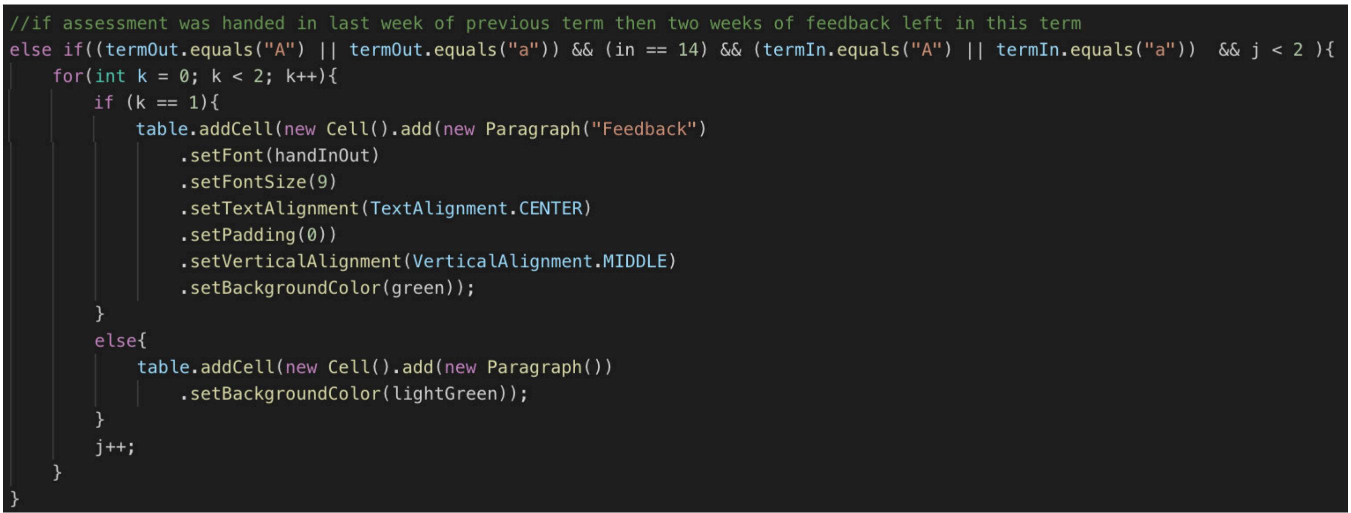
The type of cell added to the spring term table is determined using the same system of if and else if statements, however for hand out and hand in cells instead of checking the term identifier is autumn, in this method it is checked to see if it is spring.



```
int feedback = 4;
feedbackCalendar.setTime(c.getTime());
feedbackCalendar.add(Calendar.DATE, 7);
Date feedbackStart = feedbackCalendar.getTime();
feedbackCalendar.add(Calendar.DATE, 25);
Date feedbackEnd = feedbackCalendar.getTime();

if((feedbackStart.before(goodFriday) && feedbackEnd.after(goodFriday)) && (feedbackStart.before(easterMonday) && feedbackEnd.after(easterMonday))){
    //feedback period overlaps Easter holiday, length of feedback increased by 1 unless 5 weeks takes the week count over the term length
    if(j <= 11 ){
        //rest of term taken as feedback period
        feedback = 5;
    }
    else if (j >= 12){
        feedback = 15-j;
    }
}
//feedback period does not overlap easter weekend
else if (j > 12){
    //rest of term taken as feedback since less than 4 weeks of term time remaining
    feedback = 15 - j;
}
//adding in feedback cells for the assessment
for(int k = 0; k < feedback; k++){
    if (k == feedback-1){
        table.addCell(new Cell().add(new Paragraph("Feedback")
            .setFont(handInOut)
            .setFontSize(9)
            .setTextAlignment(TextAlignment.CENTER)
            .setPadding(0))
            .setVerticalAlignment(VerticalAlignment.MIDDLE)
            .setBackgroundColor(green));
    }
    else{
        table.addCell(new Cell().add(new Paragraph())
            .setBackgroundColor(lightGreen));
    }
    j++;
}
j++;
```

Figure 4.24

The addition of a hand in cell once again means the feedback period must be added in following it. This follows the same process as before with altering the length of the feedback period where necessary, this time it is checked to see if it overlaps the Easter Bank Holiday Weekend. If it does, then the feedback period is extended by 1 week to be 5 weeks in length. Like in the autumn term there are checks in place to ensure no extra cells are added to each row when the feedback period is added.

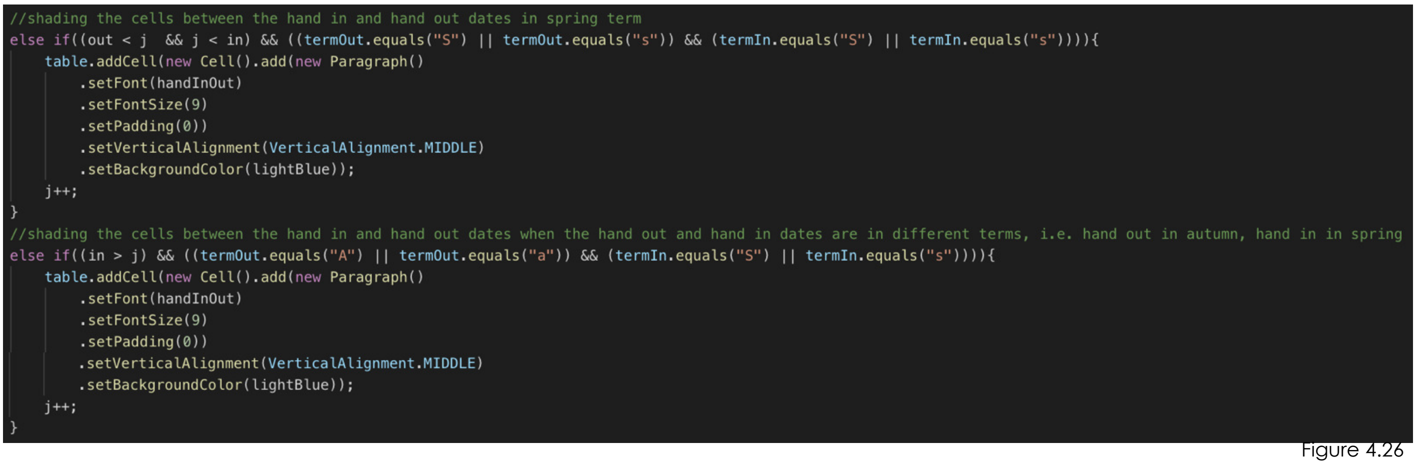


```
//if assessment was handed in last week of previous term then two weeks of feedback left in this term
else if((termOut.equals("A") || termOut.equals("a")) && (in == 14) && (termIn.equals("A") || termIn.equals("a"))  && j < 2 ){
    for(int k = 0; k < 2; k++){
        if (k == 1){
            table.addCell(new Cell().add(new Paragraph("Feedback")
                .setFont(handInOut)
                .setFontSize(9)
                .setTextAlignment(TextAlignment.CENTER)
                .setPadding(0))
                .setVerticalAlignment(VerticalAlignment.MIDDLE)
                .setBackgroundColor(green));
        }
        else{
            table.addCell(new Cell().add(new Paragraph())
                .setBackgroundColor(lightGreen));
        }
        j++;
    }
}
```

Figure 4.25

The spring term table does also have an additional else if statement in place that covers the addition of feedback cells that follow on from the previous term. In order to check if these need to be added the system will check if the hand out of the assessment was in the autumn term along with the hand in of the assessment being in term week 12 of the autumn term.

When this is the case the assessment requires the first 2 weeks of the spring term to be the remaining length of the feedback period, where the first 2 weeks have been over the exam period at the end of the autumn term. This also means the count of the week cell currently being added to the table must be less than 2 otherwise the entire row for that assessment would be added as feedback cells.



```
//shading the cells between the hand in and hand out dates in spring term
else if((out < j  && j < in) && ((termOut.equals("S") || termOut.equals("s")) && (termIn.equals("S") || termIn.equals("s")))){
    table.addCell(new Cell().add(new Paragraph()
        .setFont(handInOut)
        .setFontSize(9)
        .setPadding(0))
        .setVerticalAlignment(VerticalAlignment.MIDDLE)
        .setBackgroundColor(lightBlue));
    j++;
}
//shading the cells between the hand in and hand out dates when the hand out and hand in dates are in different terms, i.e. hand out in autumn, hand in in spring
else if((in > j) && ((termOut.equals("A") || termOut.equals("a")) && (termIn.equals("S") || termIn.equals("s")))){
    table.addCell(new Cell().add(new Paragraph()
        .setFont(handInOut)
        .setFontSize(9)
        .setPadding(0))
        .setVerticalAlignment(VerticalAlignment.MIDDLE)
        .setBackgroundColor(lightBlue));
    j++;
}
```

Figure 4.26

The last way in which the spring term method differs is in checking when to add the shaded cells to show the timeframe to complete an assessment. There are again two situations in which these cells need to be added but they differ from the autumn term. The first being when both the hand out and hand in are in the spring term and the loop count is in-between the hand out and hand in week numbers. The second is when the hand out of an assessment is in the autumn term but the hand in is in the spring term, and the loop count is less than the hand in week number.

The pdfTable.java class also contains an additional 3 methods:

- yearData – this reads in and creates a list of the assessment data for a specific year group that is used when creating the tables
- moduleKey – this method creates and adds the module key table to the pdf after each term table
- addMetaData – this adds in the iText copyright and AGPL license to the output pdf metadata so that the restrictions of using iText are complied with.

Since these 3 methods are fairly self-explanatory and do not directly add to the term assessment tables they will not be discussed further in this report. If you wish to look at these methods in depth, please refer to the source code uploaded with this report.

5. Results

Due to the combined Agile-Waterfall methodology used in this project, testing was performed each time a new functionality or feature was introduced to the system. This ensured that any new changes did not break any existing functionality and helped to guarantee the pass of the final integration test cases made in the Design and Specification section of this report.

Upon the completion of the implementation of the system both the test cases and user testing were conducted, the results of which can be seen below.

5.1. Test Cases

As before, because there are a lot of test cases the ones that best show the completion of the aims of this project are shown below with the remaining completed test cases found in appendix 6.

Test Case 02			
Requirement:	The system must create pdfs containing each year groups assessment timetable as the output.		
Pre-condition(s):	The user has the system downloaded on the machine they are using. The user has java downloaded on the machine they are using the system on. The user has the system open. The csv file with the assessment data in is in the same directory as the code for the system.		
Test Case Steps			
Steps	Expected Outcome	Actual Outcome	Pass/Fail
4. The user inputs all the required data into the GUI text fields. 5. The user clicks the make timetables button. 6. User opens the created timetable(s).	The system lets the user know the timetables have been made and user is able to open the pdf timetables with a pdf reader.	The system alerts the user to let them know the timetables have been made. Files are saved as a pdf and can be opened with Adobe Acrobat Reader.	PASS
Checker & Date of Check: Nicole Kan 16/04/2019		**Author:** Nicole Kan	

Table 5.1

Test Case 03			
Requirement:	The system must be able to let the user alter an assessment's hand in/hand out dates.		
Pre-condition(s):	The user has the system downloaded on the machine they are using. The user has java downloaded on the machine they are using the system on. The user has the system open. The csv file with the assessment data in is in the same directory as the code for the system.		
Test Case Steps			
Steps	Expected Outcome	Actual Outcome	Pass/Fail
4. The user fills in the required text fields in the GUI.	The system lets the user know a new timetable and user is	System lets the user know a new timetable has been	PASS

5. The user enters the details of the assessment they wish to change in the change module section of the GUI. 6. The user clicks the new timetable button.	able to open the updated timetable pdf.	made and the updated timetable can be opened.	
Checker & Date of Check: Nicole Kan 16/04/2019		**Author:** Nicole Kan	

Table 5.2

Test Case 05			
Requirement:	The system must have a graphical user interface (GUI).		
Pre-condition(s):	The user has the system downloaded on the machine they are using. The user has java downloaded on the machine they are using the system on.		
Test Case Steps			
Steps	Expected Outcome	Actual Outcome	Pass/Fail
5. User opens the terminal/command line. 6. User changes to the directory containing the source code for the system. 7. User enters the compile command as found in the ReadMe.txt. 8. User enters the run command as found in the ReadMe.txt.	The system will open a GUI on the user's machine allowing them to create the timetables.	A Java GUI opens.	PASS
Checker & Date of Check: Nicole Kan 16/04/2019		**Author:** Nicole Kan	

Table 5.3

Test Case 11			
Requirement:	The system should have a suitable response time for creating the timetable pdfs.		
Pre-condition(s):	The user has the system downloaded on the machine they are using. The user has java downloaded on the machine they are using the system on. The user has the system open. The csv file with the assessment data in is in the same directory as the code for the system.		
Test Case Steps			
Steps	Expected Outcome	Actual Outcome	Pass/Fail
1. User enters all the required information into the text fields in the GUI. 2. The user clicks the make timetables button. 3. User opens the timetable(s) created.	The user is waiting less than 10 seconds between telling the system to make the timetables and being told they have been made.	Less than 10 seconds wait between choosing the directory to save the timetables in and being told by the system they have been made.	PASS
Checker & Date of Check: Nicole Kan 16/04/2019		**Author:** Nicole Kan	

Table 5.4

5.2 User Testing

For the requirements that focused more on the usability of the system I had to conduct some user testing. This was done by creating a questionnaire that asked users for their opinions on how easy it is to use and understand the timetable created by the system and the ease of use of the GUI. The questionnaire was made on Google Forms and the link to it posted online so has been completed anonymously by volunteers. Screenshots of the questionnaire can be found in Appendix 4.

The results of the questionnaire will be compared to the initial acceptance criteria for the requirements they test in the Evaluation section of this report to determine if the requirements have passed or failed.

Question 1

It is easy to understand hand out/hand in dates and use the timetable.
Requirement tested: The system must create an output of timetables that are consistent in design and easy to use and understand data from.
Results:

Question 1	
Agree	14
Disagree	4
Total num. responses	18

Table 5.5

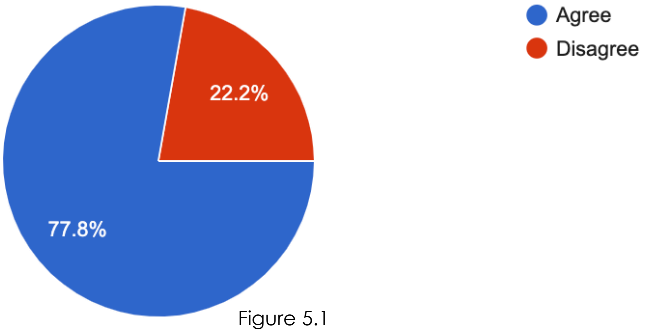


Figure 5.1

Question 2

Just by looking at the system I would know how to use it.
Requirement tested: The system must have a GUI that is easy to use, learnable and overall user-friendly.
Results:

Question 2	
Agree	13
Disagree	1
Unsure of the question	4
Total num. responses	18

Table 5.6

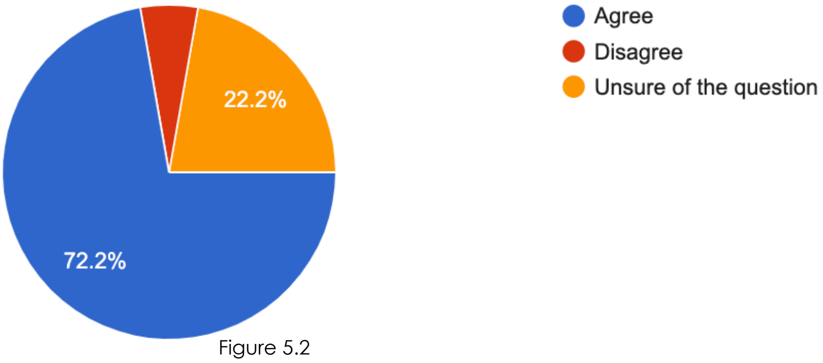


Figure 5.2

Question 3

The system looks like it would be easy to learn how to use.
Requirement tested: The system must have a GUI that is easy to use, learnable and overall user-friendly.
Results:

Question 3	
Agree	13
Disagree	2
Unsure of the question	3
Total num. responses	18

Table 5.7

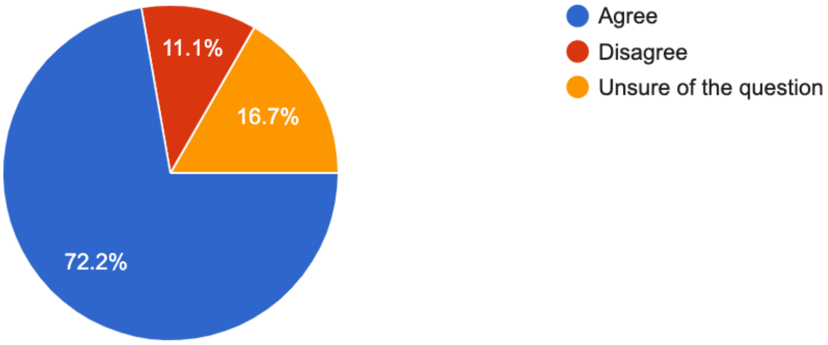


Figure 5.3

6. Evaluation

6.1. Success of the software

Having conducted the user testing and completed the test cases it can be seen that the system has received positive feedback in regard to its usability and manages to achieve the initial goal of creating a system that is able to automate the process of making assessment timetables. This in combination with all of the test cases passing and the acceptance criteria of the must have and should have requirements being met indicate that the software created in this project is a success.

Additionally, the problems regarding the current way the timetables are made have also been addressed with the creation of this system further making it a success. First and foremost, the system allows the timetables to be made in a much shorter timeframe allowing for more time for bunching checks etc. prior to handing them out. This reduction in time to produce a timetable along with the functionality of altering an assessment's hand out/hand in dates and creating an updated timetable also addresses the issue of not updating the timetable when a change is confirmed. Whilst the system is not directly linked to the change request forms and the process in which changes are approved, it does make it much easier and quicker to create a new updated timetable meaning it is less hassle for those that are responsible for alerting staff and students of the change.

By creating the system so that every year's timetable is made the same way once the user has input the academic year's term dates into the GUI, they are no longer inconsistent from year to year and are guaranteed to show the correct dates, unless there is a human error of putting the incorrect dates in. The input from the user for holiday dates also allows the timetables to now take into account holiday days and extend the feedback periods accordingly.

However, something to take into consideration is due to the nature of testing for the usability of the system it may not be an accurate representation of how users will actually feel about the system. This is because instead of having them load the system themselves and being able to actually use the system to create new tables, they were instead presented with a screenshot of the system already loaded and asked if they would know how to use it and if it would be easy to learn to use. As well as this, because the questionnaire was filled in anonymously there was no way to determine how familiar they would have been with the idea of an assessment timetable as a whole and their technical knowledge and experience with systems like this previously, meaning they could have misinterpreted or misunderstood the questions.

Despite this, the usability testing still passes what was set out in the acceptance criteria with over 70% of users agreeing they would know how to use it just by looking at it and again over 70% of people saying the system looks like it would be easy to learn to use, indicating if they were to actually use the system, it would still pass these usability tests. The only potential failure in the usability testing would be the user not knowing how to load and run the system using command line/terminal.

In regard to the test cases, since they were carried out by me and not potential users these results have the potential to be biased. Since I was the one to code and create the system this meant that I already knew how the system operated and therefore I was more likely to know the best way to get the expected outcome. However, this was somewhat pre-empted when writing the test cases since they have a step by step procedure on how to carry out the test case and when testing the system it was these steps I followed as opposed to just using the system without guidance to create the actual outcomes.

An error in the system that was not discovered in the testing of this project but rather in the use of the system after completion is that it is unable to handle assessment titles with commas in. This is because the system splits the csv data file into array lists using commas as the separators and when an assessment title contains commas it is splits up the text. This then means the assessment data following this title is not read in correctly, thus stopping an accurate assessment timetable to be made. This error is handled by a late addition to the assumptions of this project of each field within the csv file not containing any commas.

6.2 Nielsen's Heuristics

Following the usability testing, the GUI has been evaluated against Nielsen's 10 usability heuristics to ensure they were adhered to as part of the acceptance criteria for the must have non-functional requirement of: "The system must have a GUI that is easy to use, learnable and overall user-friendly.". The table below evaluates this and shows that all 10 heuristics were applied. A screenshot of the GUI which these heuristics are evaluated against can be found in appendix 7.

Heuristic	Description	System Evaluation
Visibility of system status.	The system provides the user with informative feedback about what is going on.	The system has dialog boxes that will appear once a user has performed an action that may require confirmation, i.e. the successful creation of the timetables.
Match between system and the real world.	System should use user's real-world language, phrases and concepts familiar to the user.	The system is written in English and uses date formats that are very common in day to day use.
User control and freedom.	System is able to reverse their action if needed and is fault-tolerant.	The user is able to delete and alter the input into the text fields as they wish prior to clicking the buttons that make the timetables.
Consistency and standards.	The system is consistent in design and the phrases/words used.	The GUI is consistent in its layout with the same font used throughout and input style for the term dates. Additionally, the timetables produced all follow the same template.
Error prevention.	System has error prevention in place to eliminate situations where the user may be error prone.	The system will alert the user when there is missing information that is required for making the timetables.
Recognition rather than recall.	Object, actions and options of the system are visible and clear to use and understand.	There are no hidden features in the GUI. All options of what can be done using the system are clearly displayed.
Flexibility and efficiency of use.	System can cater to both inexperienced and experience users.	The GUI is designed to be simple and straightforward to use allowing users of all skill levels to use it.
Aesthetic and minimalist design.	No irrelevant information in dialogues and design of the system kept to a minimum.	The system has not been unnecessarily coloured and does not include any additional requirements or features.
Help users recognize, diagnose and recover from errors.	Error messages in plain language to precisely indicate the problem and help user fix the error.	When the user does not enter all of the required information the system alerts them of this. Additionally, if the user enters an incorrect csv file name, it will let the user know that the file cannot be found.
Help and documentation.	Help and documentation for the system is made easily available to the user.	The system has a README.txt file that is incorporated into the download of the system.

Table 6.1

7. Future Works

Whilst this project has created a usable system and all of the initial must have and should have requirements have been met, there is still the possibility for the system to be expanded or improved in several ways. There are the should have features that were chosen not to be implemented in this project due to time constraints as well as additional features discussed in meetings that were chosen not to be pursued since they were not a priority for the basic functionality of the system.

One of these features included the could have requirements is the ability for the system to be able to create different versions of the timetable depending on the level of information the user wished to be in the timetable. This feature would allow the user to choose a timetable with additional information such as the weighting of the assessment, the lecturer who set the assessment and the type of assessment to also be in the timetable. This feature could be implemented by showing a pop up when the user clicks the make timetables button that gives the user the two options, the basic timetable or the timetable with additional details. The system would then make the timetables in the format that the user has chosen. This feature was decided to not be added in since it would have created a fair amount of code repetition for making the timetables, and it was decided that multiple formats of the timetable was of a lower priority than other features within the system.

Another feature that was discussed but chosen to be implemented was having the system be hosted as a website on the university servers instead of a standalone java application. Implementing the system this way would have meant that the assessment data could be stored in a database and dynamically updated as an assessment's information is updated and changed and even have the possibility of automatically being synced up with people's learning central timetables. Having the data saved and stored in a database would also solve the earlier mentioned problem of the system not being able to handle commas in the assessment title as it could be read into the database tables using a different method. However, this was chosen not to be done as the use of this system is restricted to a small group of users and hosting the website on the university servers would not allow control over who has access to the system. Although, if this project was to be revisited in the future then the system could be transferred over and made to have log in system that allows only authorised users from the university network to log in. The log in system could be linked to staffs pre-existing profiles and log ins so that they wouldn't have yet another log in to remember. This was not considered for the duration of this project as it would have been outside the scope of the project and access to the university network for these profiles and log ins would not have been granted.

One of the main motivations for creating this system was that making the timetables by hand was time consuming and therefore when a change request came in during the academic term it was time-consuming to create a new table and send it out to staff and students. Therefore, a could have requirement that was set was linking the change request process with the system. Whilst this has somewhat been achieved with the system produced in this project allowing the user to create an updated timetable when adjusting an assessment's hand out/hand in dates, it is not fully linked with sending out the change request forms and keeping track of whether the changes have been approved to automatically make the updated timetable without user input. The system created in this project also only allows for the change of one assessment within a year group, meaning if you require multiple assessments to be changed the user has to update them within the csv file and then go back to the system to make the timetable pdfs again.

Thus, if this project was to be continued then fully automating this system would be one of the higher priority requirements. This could be achieved by further expanding the previously mentioned future works idea of a log in system. The use of a log in system could be applied so that staff and students can use their university log in’s to view the timetable; but staff would have access to another area that allows them to send a change request form to the assessment and exam lead who has admin privileges allowing them to make new timetables and update the timetable that everyone sees when they log in to the system. Since this system would allow different types of users to communicate with each other, it could then be programmed to keep track of the progress of an ongoing change request form and once the exam and assessment lead has approved the change automatically create a new updated timetable and replace and update all occurrences of the old timetable.

8. Conclusion and Reflection on Learning

8.1. Conclusion

This project has been successful in achieving the initial aims outlined in the brief and implementing the requirements set in the design phase of this project creating a useful and helpful assessment timetable making system. All of the requirements that were prioritised to be implemented in this project have been completed and pass their test cases and acceptance criteria by using a combination of an Agile and Waterfall methodology. This allowed for the initial planning of the project, including requirement analysis and system design to be completed first, followed by the actual development of the project which was carried out in short sprints, allowing for constant user feedback to improve the system.

In conclusion, the system developed achieves the goal of automating the creation of assessment timetables for year groups within the School of Computer Science and Informatics and is a standalone java application that can be run from the command line/terminal of a machine with a GUI making the system easy to use. Despite there being a couple of minor issues encountered it does solve the initial problem stated at the start of this report as well as addressing problems and concerns that the current workflow of creating the timetables had.

8.2. Reflection on Learning

Upon completion of this project I am able to look back and see how I managed it in terms of project management and in terms of my technical knowledge throughout the 3 months this project spanned.

Project Management

Since this is the biggest individual project that I had worked on I found it hard at times to manage and gauge the progress I was making. Despite planning a workflow in the initial report, this was quickly disregarded since it was made before all the system requirements had been agreed upon and therefore did not take them all into consideration. However, in the initial plan I specified I would meet my supervisor once a week to discuss progress and answer any questions or address any concerns that had been raised in the week. By having these weekly meetings, the lack of an official workplan timeline had less of an impact since we would regularly discuss what the goals and targets for that given week would be based on what had been completed the previous week.

These meetings also helped a great deal if I was to fall behind on a weekly target that had not been met. In cases where this happened my supervisor and I would discuss how big of an impact this would have on the project timeline and what areas it would affect. If it was going to hinder the overall completion of the project then potential ways to solve it would be discussed, however in the majority of if not all of the cases where targets were met later than intended it did not affect the overall completion of the system. Taking notes in the meetings is a large factor in why this was the case. Since I took notes in every meeting nothing was forgotten, and I would be able to refer back to them when I was unsure of how to tackle a certain problem that had been discussed in meetings. These notes have also helped me to write this report as I was able to refer back to problems and discussions that happened right at the beginning of this project that I might have otherwise forgotten.

I think that the weekly meetings in combination with my use of version control for the cycles of implementation also helped a lot to ensure this project was completed in time. For this project I chose to use GitHub via GitKraken to ensure my code was backed up and implement some kind of version control. I chose to use this method as I have used it in previous workplaces and have found it to be very effective and easy to use. Additionally, because I have used it before it meant that I did not have a steep learning curve with it allowing me to focus more on the actual implementation of the code for this project.

I believe that my time management skills have also been improved throughout this project. For my weekly meetings I had to ensure I was on time and for the development of the project itself I would have to manage my time wisely when I fell behind to ensure it would all be done in time. In particular, my time management skills for implementing features of a system have improved. Prior to starting I was fairly ambitious about the timeframe in which I could complete the implementation and did not factor in enough time for unexpected errors or just certain features taking longer to implement than expected. Having discussed this with my supervisor throughout the project I believe that I will be now be much more realistic with the time I give myself to complete a certain area of implementation and if I am to fall behind, I would now know much better how to handle it and manage to catch up.

Technical Reflection

As mentioned in the Implementation section of this report I chose to use java since it is a language that I am familiar with and comfortable using. By choosing Java I believed it would cut down some time spent on learning new languages and processes which led to some amount of over-confidence in how quickly I would get the system completed, causing me to sometimes fall behind schedule.

Something which took me longer to solve then I factored in was getting the external pdf creator library to work with java from the command line. Since all the tutorials from iText themselves showed the library being used in conjunction with java via an IDE such as Eclipse and I had never used an external library in conjunction with java from the command line/terminal before I had to research how to do this. However, since I had never done this before I wasn't quite sure what to look up when trying to get iText to work with java. This meant the research for finding the solution of adding the iText jar files to the class path when compiling and running the java files took longer than expected.

Additionally, using java meant that I would use swing to create the GUI for this system. Whilst I had briefly used swing a couple years prior to this project I had not had to design an entire system from scratch and format the layout how I wanted it to be. This meant that I had to familiarise myself with the different ways in which the GUI could be formatted and whilst there was an option that maybe would have allowed me to have more freedom creating the layout of the GUI, I ultimately decided to go for the one I did since I was able to understand how it worked in shorter time span allowing me to spend more time on the implementation of making the timetables.

Although certain areas of the technical aspect of this project did take longer than projected and I feel this was the main area I struggled with in the technical development of this project it helped me to develop my and improve my research skills and creative thinking skills as if I was unable to find a solution in the route I was currently taking I would have to explore alternative ways to solve my problem and see if that would work instead.

References

Phillips, H. 2019a. Meeting with Kan, N. 31 January 2019a.

Phillips, H. 2019b. Email to Kan, N. 26 March 2019b.

Cardiff University 2018. Academic Regulations Handbook. Available at: https://www.cardiff.ac.uk/__data/assets/pdf_file/0009/432666/ARH-2018-19-English-Complete.pdf [Accessed 18/04/2019]

LucidChart Content Team. 2017. What the Waterfall Project Management Methodology Can (and Can't) Do for You. Available at: https://www.lucidchart.com/blog/waterfall-project-management-methodology [Accessed 18/04/2019]

Kevin Lonergan. 2016. The Pros and Cons of Agile and Waterfall. Available at: https://www.pmis-consulting.com/agile-versus-waterfall [Accessed 18/04/2019]

Pivotal Tracker. 2017. What is agile project management? Available at: https://www.pivotaltracker.com/campaigns/what-is-agile [Accessed 19/04/2019]

Inflectra. 2015. What is Waterfall and Hybrid Development? Available at: https://www.inflectra.com/methodologies/waterfall.aspx [Accessed 19/04/2019]

Nico Budi Darmawan. 2014. Balance Between Agile and Waterfall Software Development. Available at: http://www.nicobudidarmawan.com/2014/03/balance-between-agile-and-waterfall.html [Accessed 19/04/2019]

Ulf Eriksson. 2012. Functional Vs Non Functional Requirements. Available at: https://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/ [Accessed 19/04/2019]

Agile Business. 2014. MoSCoW Prioritisations. Available at: https://www.agilebusiness.org/content/moscow-prioritisation [Accessed 19/04/2019]

Jakob Nielsen. 1994. 10 Usability Heuristics for User Interface Design. Available at: https://www.nngroup.com/articles/ten-usability-heuristics/ [Accessed 19/04/2019]

Barry at Medium. 2019. Why Use Java That Requires Even More Lines Of Code Than Python. Available at: https://medium.com/@trungluongquang/why-use-java-that-even-requires-more-lines-of-code-than-python-7805703e4763 [Accessed: 19/04/2019]

Tutorials Point. 2015. Python – GUI Programming (Tkinter). Available at: https://www.tutorialspoint.com/python/python_gui_programming.htm [Accessed 19/04/2019]

Techopedia. 2011. Java Swing. Available at: https://www.techopedia.com/definition/26102/java-swing [Accessed 19/04/2019]

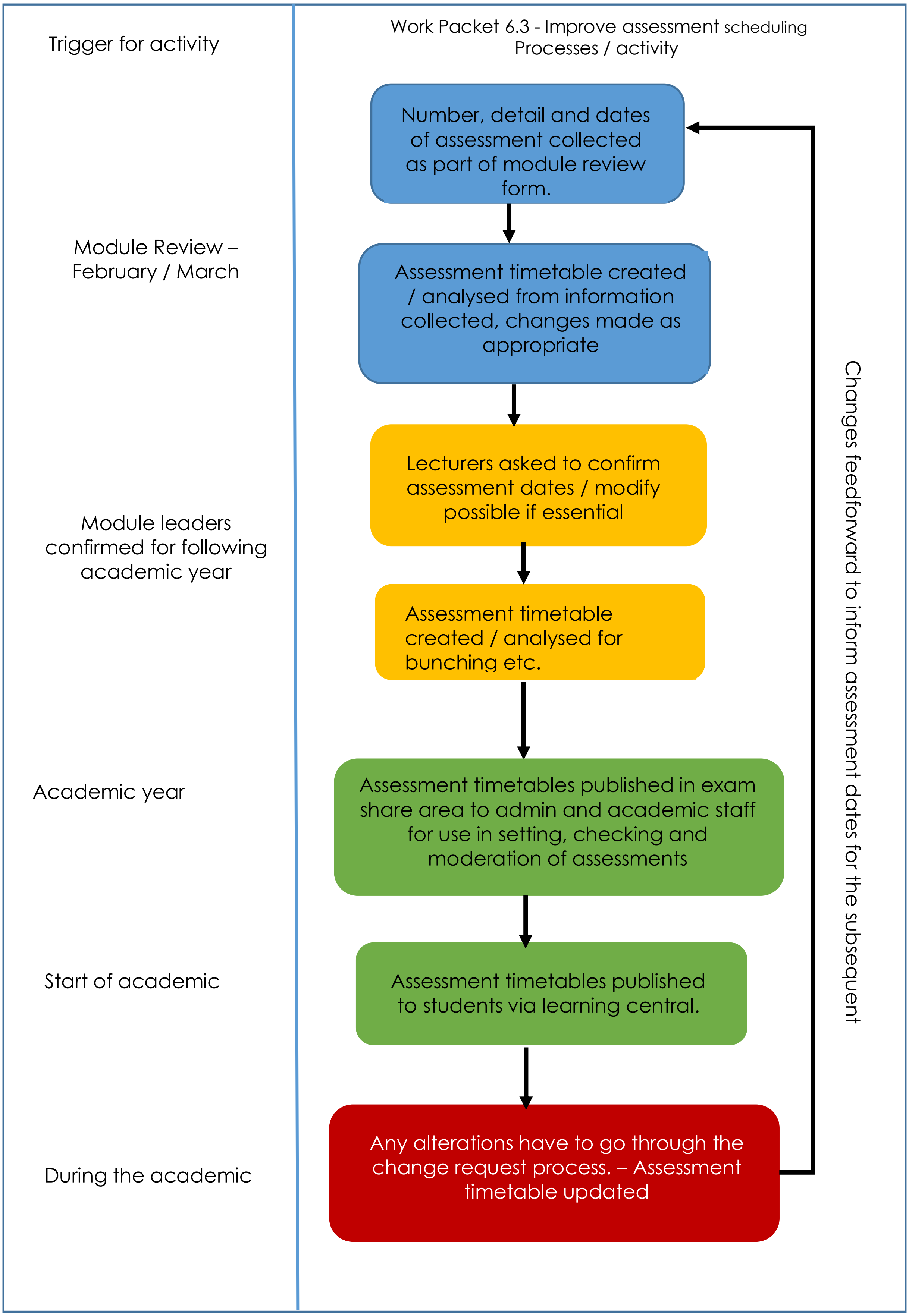
javatpoint. 2014. Java Swing Tutorial. Available at: https://www.javatpoint.com/java-swing [Accessed 19/04/2019]

iText. 2019a. iText Homepage. Available at: https://itextpdf.com/en [Accessed: 23/04/2019] iText. 2019b. iText Free Trial. Available at: https://itextpdf.com/en/free-trial [Accessed: 23/04/2019]

iText. 2019b. AGPL License. Available at: https://itextpdf.com/en/how-buy/agpl-license [Accessed: 23/04/2019]

Appendices

Appendix 1



Trigger for activity

Work Packet 6.3 - Improve assessment scheduling
Processes / activity

Number, detail and dates of assessment collected as part of module review form.

Module Review – February / March

Assessment timetable created / analysed from information collected, changes made as appropriate

Module leaders confirmed for following academic year

Lecturers asked to confirm assessment dates / modify possible if essential

Assessment timetable created / analysed for bunching etc.

Academic year

Assessment timetables published in exam share area to admin and academic staff for use in setting, checking and moderation of assessments

Start of academic

Assessment timetables published to students via learning central.

During the academic

Any alterations have to go through the change request process. – Assessment timetable updated

Changes feedforward to inform assessment dates for the subsequent

Appendix 2

Guidance on the Nature and Volume of Assessment in Modules on Taught Programmes of Study

For staff across the University involved in taught programmes of study.

Purpose:

- **To help schools develop a better balance between formative and summative assessment;**
- **To assist schools in reducing the volume of summative assessment;**
- **To help staff introduce a wider variety of suitable assessment methods within programmes of study; and**
- **To improve the alignment between teaching, learning and assessment.**

1. Background

> *"A number of factors have conspired to increase the number of assessment tasks students are required to complete. … The result is a significant increase in assessment load for students and marking load for staff."* [1]

1.1. This guidance has been put together to help staff develop and/or revise assessment strategies within and across modular programmes of study. It has been developed through the University's 'Assessment Matters' project and has been endorsed by Senate.

1.2. The role that assessment plays in supporting and shaping learning cannot be underestimated. Its importance has increased, particularly at a time when more students are adopting strategic approaches to learning. The introduction of modularisation and adoption of a learning outcomes based approach have both contributed to an increase in the volume of summative assessment that students undertake and a consequent increase in staff marking loads.

1.3. Following the Guidance will benefit student learning and help ensure that students can develop the high level knowledge, understanding, and skills from engagement with their studies. It will help academic staff to improve the quality of their teaching and assure staff that assessment is testing the learning that students have acquired through teaching and engagement with their studies. It will also help the University to demonstrate the high quality and standards of our awards.

1.4. Reducing the volume of summative assessment will give staff more time and space to provide better feedback to students on academic work and to amend the balance between formative and summative assessments. It should result in an academic experience that places more emphasis on learning and that provides students with better and more varied learning opportunities.

[1] Hornby W. (2005) Dogs, stars, Rolls Royces and old double-decker buses: efficiency and effectiveness in assessment. In *Reflections on Assessment Volume 1*. QAA, p. 11 [WWW] <www.enhancementthemes.ac.uk/documents/assessment/Reflections_on_Assessment_Volume_1FINAL.pdf> [Accessed 21/02/11]

Appendix 3

Test Case 01			
Requirement:	The system must be able to take csv data as the input for creating the timetable pdfs.		
Pre-condition(s):	The user has the system downloaded on the machine they are using. The user has java downloaded on the machine they are using the system on. The user has the system open. The csv file with the assessment data in is in the same directory as the code for the system.		
Test Case Steps			
Steps	Expected Outcome	Actual Outcome	Pass/Fail
1. The user inputs the name of the csv file they wish to make the timetables from into the file name field. 2. The user inputs all the other necessary data into the GUI. 3. The user clicks the make timetables button. 4. User opens the timetable(s) created.	The system lets the user know the timetables have been made and user is able to open the pdf timetables with a pdf reader.		
Checker & Date of Check:		**Author:** Nicole Kan	

Test Case 04			
Requirement:	The system must let the user know when information used in creating the table for an assessment is missing.		
Pre-condition(s):	The user has the system downloaded on the machine they are using. The user has java downloaded on the machine they are using the system on. The user has the system open. The csv file with the assessment data in is in the same directory as the code for the system.		
Test Case Steps			
Steps	Expected Outcome	Actual Outcome	Pass/Fail
1. The user inputs the required text fields in the GUI text fields. 2. The user clicks the make timetables button. 3. User opens the timetable(s) created.	If there is any missing information for an assessment in the timetable it will be highlighted.		
Checker & Date of Check:		**Author:** Nicole Kan	

Test Case 06	
Requirement:	The system should take into account days the university is closed when showing feedback dates on the timetable.
Pre-condition(s):	The user has the system downloaded on the machine they are using. The user has java downloaded on the machine they are using the system on.

	The user has the system open. The csv file with the assessment data in is in the same directory as the code for the system.		
Test Case Steps			
Steps	Expected Outcome	Actual Outcome	Pass/Fail
1. User enters all the required information into the text fields in the GUI. 2. The user clicks the make timetables button. 3. User opens the timetable(s) created.	Any assessment that has feedback overlapping the holiday dates input by the user has been adjusted from the standard 4 weeks to 6 weeks if over the Christmas Holiday and 5 weeks if over the Easter Weekend.		
Checker & Date of Check:		**Author:** Nicole Kan	

Test Case 07			
Requirement:	The system should let the user generate and choose which Master's modules they want included in the Master's assessment timetable.		
Pre-condition(s):	The user has the system downloaded on the machine they are using. The user has java downloaded on the machine they are using the system on. The user has the system open. The csv file with the assessment data in is in the same directory as the code for the system.		
Test Case Steps			
Steps	Expected Outcome	Actual Outcome	Pass/Fail
1. User enters the csv file name containing the assessment data into the file name text field. 2. User clicks the generate master's button. 3. User chooses the master's modules they want in the master's timetable. 4. User enters all the other required information into the text fields. 5. User opens the master's timetable created by the system.	The master's timetable contains only assessments from the modules selected by the user in step 3.		
Checker & Date of Check:		**Author:** Nicole Kan	

Test Case 08	
Requirement:	The system should remind the user to alert staff members and students of an updated table when they update an assessment and create a new table.
Pre-condition(s):	The user has the system downloaded on the machine they are using. The user has java downloaded on the machine they are using the system on. The user has the system open. The csv file with the assessment data in is in the same directory as the code for the system.

Test Case Steps			
Steps	Expected Outcome	Actual Outcome	Pass/Fail
1. The user fills in the required text fields in the GUI. 2. The user enters the details of the assessment they wish to change in the change module section of the GUI. 3. The user clicks the new timetable button.	The system shows a dialog box to let the user know a new timetable has been made and reminds the user to alert the relevant staff and students to the new updated timetable.		
Checker & Date of Check:		**Author:** Nicole Kan	

Test Case 9			
Requirement:	The system must be reliable.		
Pre-condition(s):	The user has the system downloaded on the machine they are using. The user has java downloaded on the machine they are using the system on. The user has the system open. The csv file with the assessment data in is in the same directory as the code for the system. Each csv file used in the test case follows the assumed level of information found in the Assumptions section of this report.		
Test Case Steps			
Steps	Expected Outcome	Actual Outcome	Pass/Fail
1. The user inputs the required information in the GUI text fields. 2. The user clicks the make timetables button. 3. User repeats steps 1-2 9 times with a different csv file each time.	The system produces timetable(s) for all the csv files mostly without failing or crashing. A success rate of at least 75% is achieved.		
Checker & Date of Check:		**Author:** Nicole Kan	

Test Case 10			
Requirement:	The system must be able to run on multiple different operating systems.		
Pre-condition(s):	The user has the system downloaded on the machine they are using. The user has java downloaded on the machine they are using the system on. The user has the system open. The csv file with the assessment data in is in the same directory as the code for the system.		
Test Case Steps			
Steps	Expected Outcome	Actual Outcome	Pass/Fail
1. On a windows machine the user inputs the required information in the GUI text fields. 2. The user clicks the make timetables button. 3. User opens the timetables created. 4. On a Macintosh machine the user inputs the required	The user is able to create and open the timetables on all 3 machines with no error.		

information in the GUI text fields. 5. The user clicks the make timetables button. 6. User opens the timetables created. 7. On a Unix machine the user inputs the required information in the GUI text fields. 8. The user clicks the make timetables button. 9. User opens the timetables created.			
Checker & Date of Check:		**Author:** Nicole Kan	

Test Case 12			
Requirement:	The system code should be easily maintained and updated in the future.		
Pre-condition(s):	The user has the system downloaded on the machine they are using.		
Test Case Steps			
Steps	Expected Outcome	Actual Outcome	Pass/Fail
4. For each file in the source code of the system the user opens the file and checks each file is no longer than 1000 lines.	Each file has less than 1000 lines of code.		
Checker & Date of Check:		**Author:** Nicole Kan	

Appendix 4

Timetable assessment and Scheduling System

* Required

Questionnaire

Using the attached image it is easy to understand coursework hand out/hand in dates and use the timetable. *

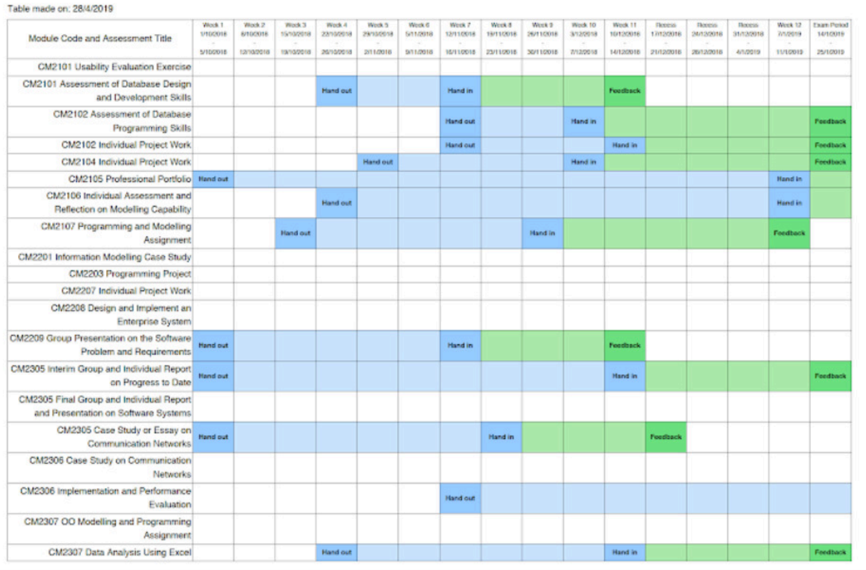


Table made on: 28/4/2019

Module Code and Assessment Title	Week 1 1/10/2018 - 5/10/2018	Week 2 8/10/2018 - 12/10/2018	Week 3 15/10/2018 - 19/10/2018	Week 4 22/10/2018 - 26/10/2018	Week 5 29/10/2018 - 2/11/2018	Week 6 5/11/2018 - 9/11/2018	Week 7 12/11/2018 - 16/11/2018	Week 8 19/11/2018 - 23/11/2018	Week 9 26/11/2018 - 30/11/2018	Week 10 3/12/2018 - 7/12/2018	Week 11 10/12/2018 - 14/12/2018	Recess 17/12/2018 - 21/12/2018	Recess 24/12/2018 - 28/12/2018	Recess 31/12/2018 - 4/1/2019	Week 12 7/1/2019 - 11/1/2019	Exam Period 14/1/2019 - 25/1/2019
CM2101 Usability Evaluation Exercise																
CM2101 Assessment of Database Design and Development Skills				Hand out			Hand in				Feedback					
CM2102 Assessment of Database Programming Skills							Hand out			Hand in						Feedback
CM2102 Individual Project Work							Hand out				Hand in					Feedback
CM2104 Individual Project Work					Hand out					Hand in						Feedback
CM2105 Professional Portfolio	Hand out														Hand in	
CM2106 Individual Assessment and Reflection on Modelling Capability				Hand out											Hand in	
CM2107 Programming and Modelling Assignment			Hand out						Hand in						Feedback	
CM2201 Information Modelling Case Study																
CM2203 Programming Project																
CM2207 Individual Project Work																
CM2208 Design and Implement an Enterprise System																
CM2209 Group Presentation on the Software Problem and Requirements	Hand out						Hand in				Feedback					
CM2305 Interim Group and Individual Report on Progress to Date	Hand out										Hand in					Feedback
CM2305 Final Group and Individual Report and Presentation on Software Systems																
CM2305 Case Study or Essay on Communication Networks	Hand out							Hand in				Feedback				
CM2306 Case Study on Communication Networks																
CM2306 Implementation and Performance Evaluation							Hand out									
CM2307 OO Modelling and Programming Assignment																
CM2307 Data Analysis Using Excel				Hand out							Hand in					Feedback

○ Agree

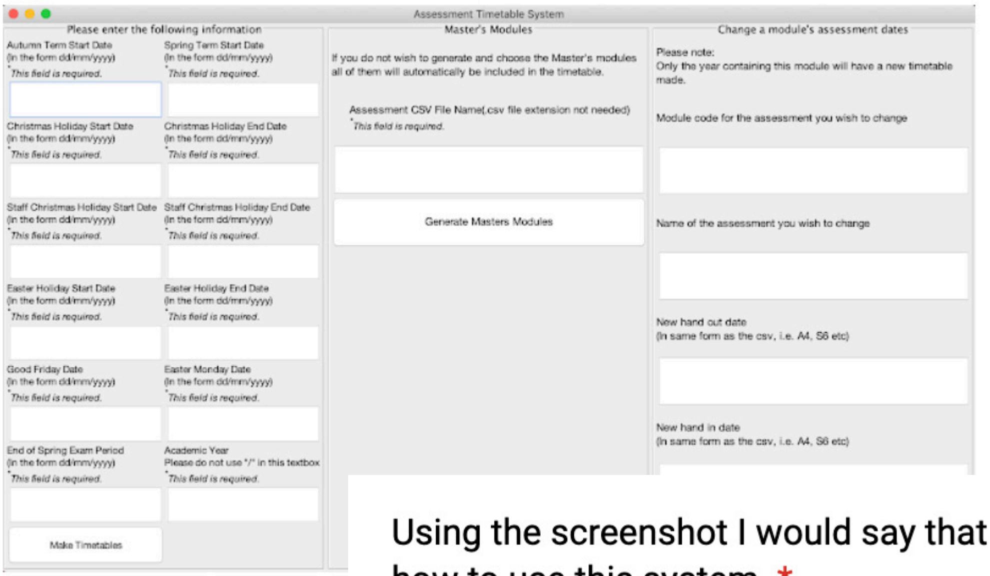
○ Disagree

BACK NEXT

User Interface

The following questions will ask you your opinions on the ease of use and learnability of a user interface for an assessment timetable system. To give some context to the system it is currently limited in scope to the School of Computer Science and Informatics at Cardiff University and aims to automatically create assessment timetables in the form of pdfs from data in a csv file.

Looking at the screenshot below I would know how to use the system create the assessment timetables. *



Assessment Timetable System

Please enter the following information

Autumn Term Start Date
(In the form dd/mm/yyyy)
This field is required.

Spring Term Start Date
(In the form dd/mm/yyyy)
This field is required.

Christmas Holiday Start Date
(In the form dd/mm/yyyy)
This field is required.

Christmas Holiday End Date
(In the form dd/mm/yyyy)
This field is required.

Staff Christmas Holiday Start Date
(In the form dd/mm/yyyy)
This field is required.

Staff Christmas Holiday End Date
(In the form dd/mm/yyyy)
This field is required.

Easter Holiday Start Date
(In the form dd/mm/yyyy)
This field is required.

Easter Holiday End Date
(In the form dd/mm/yyyy)
This field is required.

Good Friday Date
(In the form dd/mm/yyyy)
This field is required.

Easter Monday Date
(In the form dd/mm/yyyy)
This field is required.

End of Spring Exam Period
(In the form dd/mm/yyyy)
This field is required.

Academic Year
Please do not use "/" in this textbox
This field is required.

Master's Modules

If you do not wish to generate and choose the Master's modules all of them will automatically be included in the timetable.

Assessment CSV File Name(.csv file extension not needed)
This field is required.

Change a module's assessment dates

Please note:
Only the year containing this module will have a new timetable made.

Module code for the assessment you wish to change

Name of the assessment you wish to change

New hand out date
(In same form as the csv, i.e. A4, S6 etc)

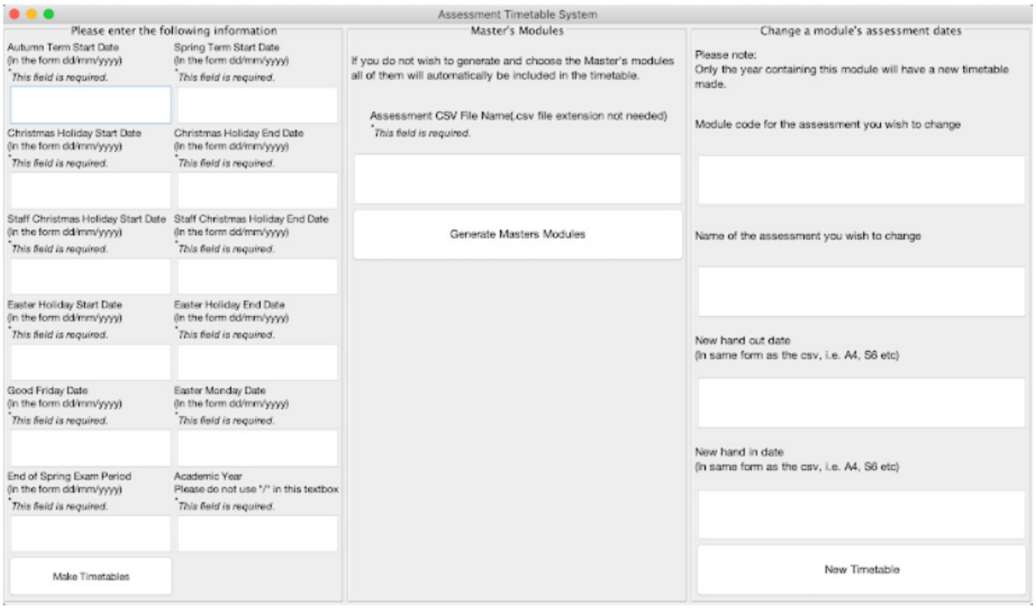
New hand in date
(In same form as the csv, i.e. A4, S6 etc)

○ Agree

○ Disagree

○ Unsure of the question

Using the screenshot I would say that it would be easy to learn how to use this system. *



Assessment Timetable System

Please enter the following information

Autumn Term Start Date
(In the form dd/mm/yyyy)
This field is required.

Spring Term Start Date
(In the form dd/mm/yyyy)
This field is required.

Christmas Holiday Start Date
(In the form dd/mm/yyyy)
This field is required.

Christmas Holiday End Date
(In the form dd/mm/yyyy)
This field is required.

Staff Christmas Holiday Start Date
(In the form dd/mm/yyyy)
This field is required.

Staff Christmas Holiday End Date
(In the form dd/mm/yyyy)
This field is required.

Easter Holiday Start Date
(In the form dd/mm/yyyy)
This field is required.

Easter Holiday End Date
(In the form dd/mm/yyyy)
This field is required.

Good Friday Date
(In the form dd/mm/yyyy)
This field is required.

Easter Monday Date
(In the form dd/mm/yyyy)
This field is required.

End of Spring Exam Period
(In the form dd/mm/yyyy)
This field is required.

Academic Year
Please do not use "/" in this textbox
This field is required.

Master's Modules

If you do not wish to generate and choose the Master's modules all of them will automatically be included in the timetable.

Assessment CSV File Name(.csv file extension not needed)
This field is required.

Change a module's assessment dates

Please note:
Only the year containing this module will have a new timetable made.

Module code for the assessment you wish to change

Name of the assessment you wish to change

New hand out date
(In same form as the csv, i.e. A4, S6 etc)

New hand in date
(In same form as the csv, i.e. A4, S6 etc)

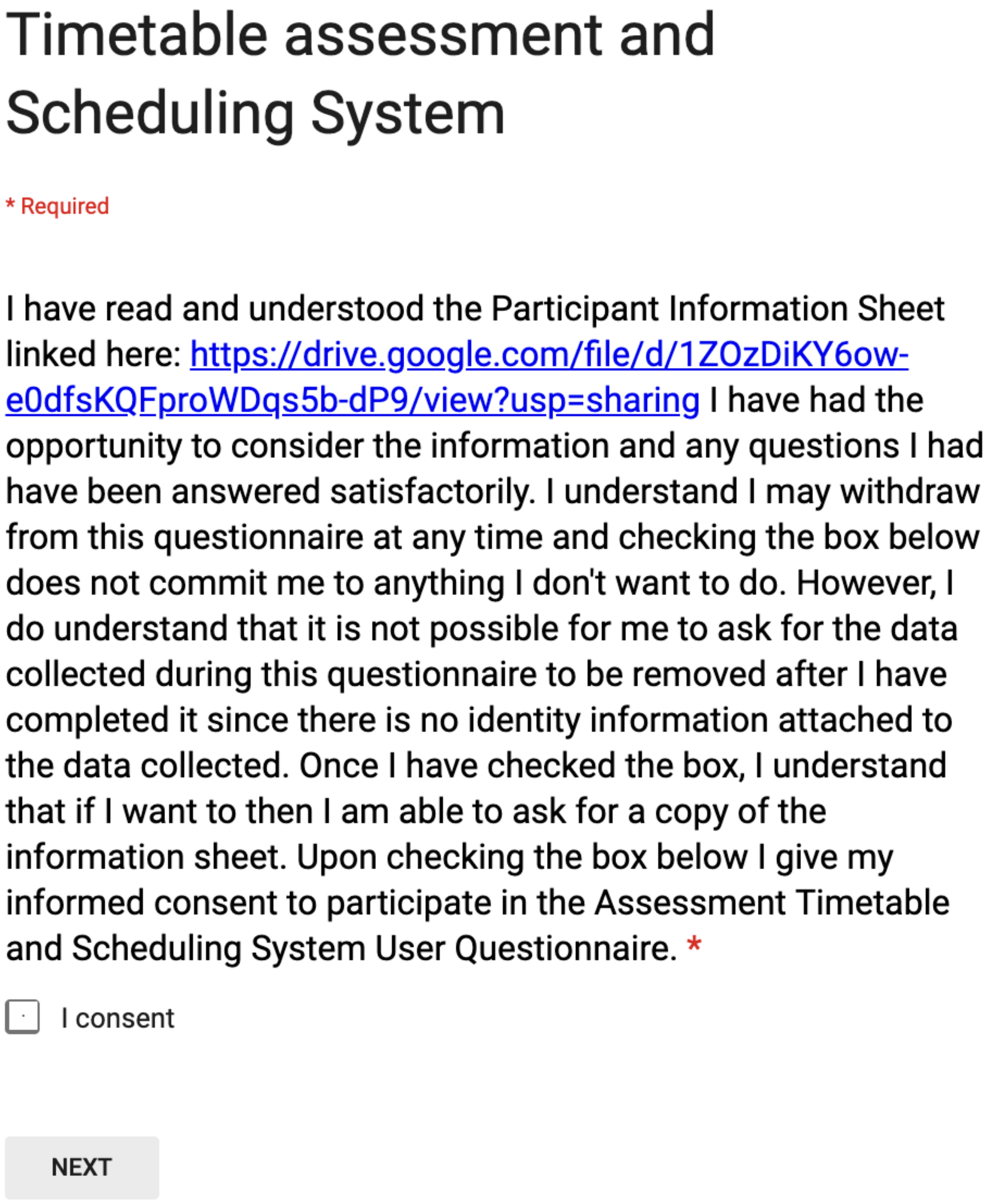
○ Agree

○ Disagree

○ Unsure of the question

BACK SUBMIT

Appendix 5



Timetable assessment and Scheduling System

* Required

I have read and understood the Participant Information Sheet linked here: https://drive.google.com/file/d/1ZOzDiKY6ow-e0dfsKQFproWDqs5b-dP9/view?usp=sharing I have had the opportunity to consider the information and any questions I had have been answered satisfactorily. I understand I may withdraw from this questionnaire at any time and checking the box below does not commit me to anything I don't want to do. However, I do understand that it is not possible for me to ask for the data collected during this questionnaire to be removed after I have completed it since there is no identity information attached to the data collected. Once I have checked the box, I understand that if I want to then I am able to ask for a copy of the information sheet. Upon checking the box below I give my informed consent to participate in the Assessment Timetable and Scheduling System User Questionnaire. *

□ I consent

Consent Form

Assessment Timetable and Scheduling System Information Sheet and Consent Information

You are invited to take part in user testing for an ongoing project. Before you decide if you wish to take part please read the following information about the project and your consent in taking part in this user research.

What is the purpose of this testing?

This purpose of this testing is to see if the output (a pdf document) of an assessment timetable and scheduling system is clear and easy to understand. As well as if the user interface of the system is self-explanatory and easy to use.
You will be shown a screenshot/image of a pdf timetable and the system itself and asked to choose how strongly you agree with the statement in front of you. i.e. The timetable is easy to understand with the choices: strongly agree, agree, indifferent, disagree, strongly disagree.
It is part of a student's undergraduate final year project in the Cardiff Uni School of Computer Science and Informatics and is not funded.

What does participating consist of?

Participating in this user testing consists of being asked a series of multiple-choice questions about your opinion on various factors about the screenshots that will be shown to you. If you wish to know further information please email KanNW@cardiff.ac.uk.

If at any point you wish to withdraw your consent and information from this questionnaire for any reason see **What if I want to remove my information?** below.

If you are still not satisfied with this answer then please email KanNW@cardiff.ac.uk.

Why have I been chosen?

As a volunteer you have responded to our request for participants to take part via a mailing list or seeing a link posted by the student conducting the project online.

How will data be collected and stored?

The data collected will be kept confidential. The information will be collected via the Google Form you complete and stored securely on a Cardiff University installation of OneDrive that is protected via a username and password combination.
We may share the data we collect with researchers at other institutions, but any information that leaves Cardiff University will have your personal details removed. In any sort of output we might publish, we will not include information that will make it possible for other people to know your name or identify you in any way.
The data collected will be stored by the student for the duration of the project which ends when the student conducting this project has graduated (19th July 2019). However, it will be stored for 5 years and the duration of the project by the supervisor. (Helen Phillips, PhillipsHR@cardiff.ac.uk)

What if I don't want to remove my information?

Your participation in this study is entirely voluntary and you may exit/close the google form at any time you wish without having to give a reason. However it is not possible to ask for the data you have given us to be removed after completing the questionnaire since no identity information is attached to the data we are collecting.

Complaints regarding this study

In the first instance you should contact the leader of the project:

Nicole Kan
KanNW@cardiff.ac.uk

If you are still unhappy, please contact the relevant Ethics Committee:
Cardiff School of Computer Science & Informatics Ethics Committee
Email: comsc-ethics@cardiff.ac.uk

The data controller is Cardiff University and Data Protection Officer is Matt Cooper. CooperM1@cardiff.ac.uk. The lawful basis for the processing of the data you provide is consent.

Who has reviewed the study?
The study has been reviewed and approved by the Ethics Committee of the School of Computer Science and Informatics, Cardiff University. Application number:

Further information
If you require further information about the project please contact Nicole Kan using KanNW@cardiff.ac.uk.

Appendix 6

<table>
<tr><th colspan="4">Test Case 01</th></tr>
<tr><td><b>Requirement:</b></td><td colspan="3">The system must be able to take csv data as the input for creating the timetable pdfs.</td></tr>
<tr><td><b>Pre-condition(s):</b></td><td colspan="3">The user has the system downloaded on the machine they are using.
The user has java downloaded on the machine they are using the system on.
The user has the system open.
The csv file with the assessment data in is in the same directory as the code for the system.</td></tr>
<tr><td colspan="4"><b>Test Case Steps</b></td></tr>
<tr><td>Steps</td><td>Expected Outcome</td><td>Actual Outcome</td><td>Pass/Fail</td></tr>
<tr><td>The user inputs the name of the csv file they wish to make the timetables from into the file name field.
The user inputs all the other necessary data into the GUI.
The user clicks the make timetables button.
User opens the timetable(s) created.</td><td>The system lets the user know the timetables have been made and user is able to open the pdf timetables with a pdf reader.</td><td>System does not throw any errors and makes the timetable. User is able to open timetables made with the system.</td><td>PASS</td></tr>
<tr><td colspan="2"><b>Checker & Date of Check:</b> Nicole Kan 16/04/2019</td><td colspan="2"><b>Author:</b> Nicole Kan</td></tr>
</table>

<table>
<tr><th colspan="4">Test Case 04</th></tr>
<tr><td><b>Requirement:</b></td><td colspan="3">The system must let the user know when information used in creating the table for an assessment is missing.</td></tr>
<tr><td><b>Pre-condition(s):</b></td><td colspan="3">The user has the system downloaded on the machine they are using.
The user has java downloaded on the machine they are using the system on.
The user has the system open.
The csv file with the assessment data in is in the same directory as the code for the system.</td></tr>
<tr><td colspan="4">Test Case Steps</td></tr>
<tr><td>Steps</td><td>Expected Outcome</td><td>Actual Outcome</td><td>Pass/Fail</td></tr>
<tr><td>4. The user inputs the required text fields in the GUI text fields.
5. The user clicks the make timetables button.
6. User opens the timetable(s) created.</td><td>If there is any missing information for an assessment in the timetable it will be highlighted.</td><td>An assessment with no hand out or hand in date in a year’s timetable had the row highlighted in yellow.</td><td>PASS</td></tr>
<tr><td colspan="2"><b>Checker & Date of Check:</b> Nicole Kan 16/04/2019</td><td colspan="2"><b>Author:</b> Nicole Kan</td></tr>
</table>

<table>
<tr><th colspan="2">Test Case 06</th></tr>
<tr><td><b>Requirement:</b></td><td>The system should take into account days the university is closed when showing feedback dates on the timetable.</td></tr>
<tr><td><b>Pre-condition(s):</b></td><td>The user has the system downloaded on the machine they are using.
The user has java downloaded on the machine they are using the system on.</td></tr>
</table>

	The user has the system open. The csv file with the assessment data in is in the same directory as the code for the system.		
Test Case Steps			
Steps	Expected Outcome	Actual Outcome	Pass/Fail
4. User enters all the required information into the text fields in the GUI. 5. The user clicks the make timetables button. 6. User opens the timetable(s) created.	Any assessment that has feedback overlapping the holiday dates input by the user has been adjusted from the standard 4 weeks to 6 weeks if over the Christmas Holiday and 5 weeks if over the Easter Weekend.	Feedback periods in the timetables are adjusted depending on the dates input by the user.	PASS
Checker & Date of Check: Nicole Kan 16/04/2019		**Author:** Nicole Kan	

Test Case 07			
Requirement:	The system should let the user generate and choose which Master's modules they want included in the Master's assessment timetable.		
Pre-condition(s):	The user has the system downloaded on the machine they are using. The user has java downloaded on the machine they are using the system on. The user has the system open. The csv file with the assessment data in is in the same directory as the code for the system.		
Test Case Steps			
Steps	Expected Outcome	Actual Outcome	Pass/Fail
6. User enters the csv file name containing the assessment data into the file name text field. 7. User clicks the generate master's button. 8. User chooses the master's modules they want in the master's timetable. 9. User enters all the other required information into the text fields. 10. User opens the master's timetable created by the system.	The master's timetable contains only assessments from the modules selected by the user in step 3.	The master's timetable had a smaller table with only the modules selected in the GUI present.	PASS
Checker & Date of Check: Nicole Kan 16/04/2019		**Author:** Nicole Kan	

Test Case 08	
Requirement:	The system should remind the user to alert staff members and students of an updated table when they update an assessment and create a new table.
Pre-condition(s):	The user has the system downloaded on the machine they are using. The user has java downloaded on the machine they are using the system on. The user has the system open. The csv file with the assessment data in is in the same directory as the code for the system.

Test Case Steps			
Steps	Expected Outcome	Actual Outcome	Pass/Fail
4. The user fills in the required text fields in the GUI. 5. The user enters the details of the assessment they wish to change in the change module section of the GUI. 6. The user clicks the new timetable button.	The system shows a dialog box to let the user know a new timetable has been made and reminds the user to alert the relevant staff and students to the new updated timetable.	A dialog box opens with the text: "Table Updated. Please remember to email out the updated timetable to staff and students."	PASS
Checker & Date of Check: Nicole Kan 16/04/2019		**Author:** Nicole Kan	

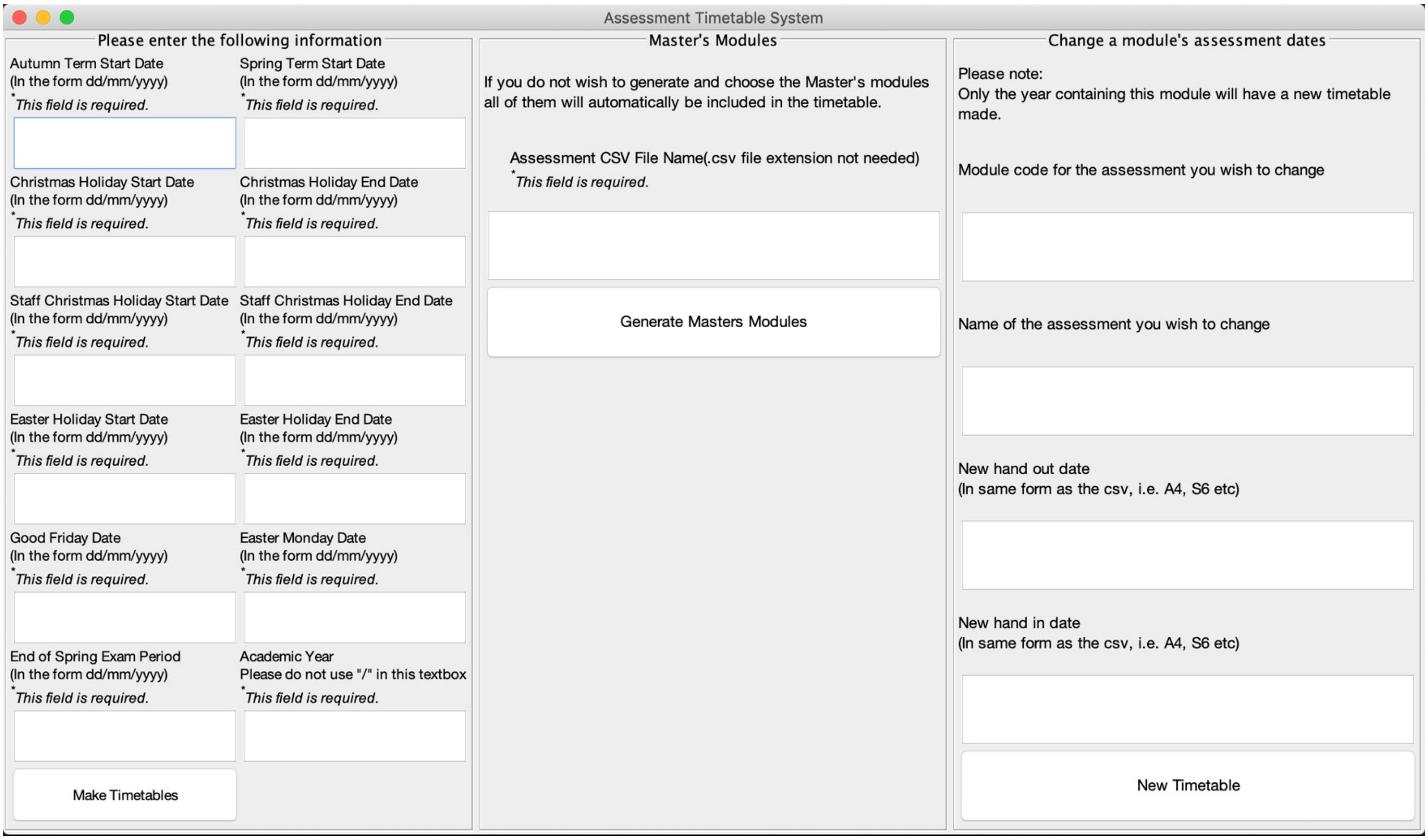
Test Case 09			
Requirement:	The system must be reliable.		
Pre-condition(s):	The user has the system downloaded on the machine they are using. The user has java downloaded on the machine they are using the system on. The user has the system open. The csv file with the assessment data in is in the same directory as the code for the system. Each csv file used in the test case follows the assumed level of information found in the Assumptions section of this report.		
Test Case Steps			
Steps	Expected Outcome	Actual Outcome	Pass/Fail
4. The user inputs the required information in the GUI text fields. 5. The user clicks the make timetables button. 6. User repeats steps 1-2 9 times with a different csv file each time.	The system produces timetable(s) for all the csv files mostly without failing or crashing. A success rate of at least 75% is achieved.	The system was able to produce timetables for all csv files given to it.	PASS
Checker & Date of Check: Nicole Kan 16/04/2019		**Author:** Nicole Kan	

Test Case 10			
Requirement:	The system must be able to run on multiple different operating systems.		
Pre-condition(s):	The user has the system downloaded on the machine they are using. The user has java downloaded on the machine they are using the system on. The user has the system open. The csv file with the assessment data in is in the same directory as the code for the system.		
Test Case Steps			
Steps	Expected Outcome	Actual Outcome	Pass/Fail
10. On a windows machine the user inputs the required information in the GUI text fields. 11. The user clicks the make timetables button. 12. User opens the timetables created. 13. On a Macintosh machine the user inputs the required	The user is able to create and open the timetables on all 3 machines with no error.	All 3 machines allowed the system to run and produce the timetables and open them.	PASS

information in the GUI text fields. 14. The user clicks the make timetables button. 15. User opens the timetables created. 16. On a Unix machine the user inputs the required information in the GUI text fields. 17. The user clicks the make timetables button. 18. User opens the timetables created.			
Checker & Date of Check: Nicole Kan 16/04/2019		**Author:** Nicole Kan	

Test Case 12			
Requirement:	The system code should be easily maintained and updated in the future.		
Pre-condition(s):	The user has the system downloaded on the machine they are using.		
Test Case Steps			
Steps	Expected Outcome	Actual Outcome	Pass/Fail
1. For each file in the source code of the system the user opens the file and checks each file is no longer than 1000 lines.	Each file has less than 1000 lines of code.	Each file has less than 1000 lines of code.	PASS
Checker & Date of Check: Nicole Kan 16/04/2019		**Author:** Nicole Kan	

Appendix 7



Please enter the following information

Autumn Term Start Date
(In the form dd/mm/yyyy)
This field is required.

Spring Term Start Date
(In the form dd/mm/yyyy)
This field is required.

Christmas Holiday Start Date
(In the form dd/mm/yyyy)
This field is required.

Christmas Holiday End Date
(In the form dd/mm/yyyy)
This field is required.

Staff Christmas Holiday Start Date
(In the form dd/mm/yyyy)
This field is required.

Staff Christmas Holiday End Date
(In the form dd/mm/yyyy)
This field is required.

Easter Holiday Start Date
(In the form dd/mm/yyyy)
This field is required.

Easter Holiday End Date
(In the form dd/mm/yyyy)
This field is required.

Good Friday Date
(In the form dd/mm/yyyy)
This field is required.

Easter Monday Date
(In the form dd/mm/yyyy)
This field is required.

End of Spring Exam Period
(In the form dd/mm/yyyy)
This field is required.

Academic Year
Please do not use "/" in this textbox
This field is required.

Master's Modules

If you do not wish to generate and choose the Master's modules all of them will automatically be included in the timetable.

Assessment CSV File Name(.csv file extension not needed)
This field is required.

Change a module's assessment dates

Please note:
Only the year containing this module will have a new timetable made.

Module code for the assessment you wish to change

Name of the assessment you wish to change

New hand out date
(In same form as the csv, i.e. A4, S6 etc)

New hand in date
(In same form as the csv, i.e. A4, S6 etc)