COMP1690 (2019/20)	**Component Programming**	**Faculty Header ID: 300926**	**Contribution: 100% of course**
Course Leader: Dr Markus Wolf	**Practical**		**Deadline Date: Monday 16/03/2020**
This coursework should take an average student who is up-to-date with tutorial work approximately 50 hours Feedback and grades are normally made available within 15 working days of the coursework deadline			
Learning Outcomes: 1, 2, 3			

Plagiarism is presenting somebody else's work as your own. It includes: copying information directly from the Web or books without referencing the material; submitting joint coursework as an individual effort; copying another student's coursework; stealing coursework from another student and submitting it as your own work. Suspected plagiarism will be investigated and if found to have occurred will be dealt with according to the procedures set down by the University. Please see your student handbook for further details of what is / isn't plagiarism.

All material copied or amended from any source (e.g. internet, books) must be referenced correctly according to the reference style you are using.

Your work will be submitted for plagiarism checking. Any attempt to bypass our plagiarism detection systems will be treated as a severe Assessment Offence.

Coursework Submission Requirements

- An electronic copy of your work for this coursework must be fully uploaded on the Deadline Date of **Monday 16/03/2020** using the link on the coursework Moodle page for COMP1690.
- For this coursework you must submit a single PDF document. In general, any text in the document must not be an image (i.e. must not be scanned) and would normally be generated from other documents (e.g. MS Office using "Save As .. PDF"). An exception to this is hand written mathematical notation, but when scanning do ensure the file size is not excessive.
- For this coursework you must also upload a single **ZIP** file containing supporting evidence.

- There are limits on the file size (see the relevant course Moodle page).
- Make sure that any files you upload are virus-free and not protected by a password or corrupted otherwise they will be treated as null submissions.
- Your work will not be printed in colour. Please ensure that any pages with colour are acceptable when printed in Black and White.
- You must NOT submit a paper copy of this coursework.
- All courseworks must be submitted as above. Under no circumstances can they be accepted by academic staff

The University website has details of the current Coursework Regulations, including details of penalties for late submission, procedures for Extenuating Circumstances, and penalties for Assessment Offences. See http://www2.gre.ac.uk/current-students/regs

Coursework Specification

This assignment consists of two parts:

- **Part A will be completed in a group**
- **Part B must be completed individually**

Please read the entire coursework specification before starting your work.

Holiday Booking System

You have been approached by a construction company called *Straight Walls Ltd* to develop a holiday booking system. The company gives all of its employees a number of days of holiday entitlement per year, but the company has found it difficult to continue functioning properly at times when staff are on leave. The company relies on a variety of skills in their daily operations. This is aggravated by the fact that staff often go on leave at similar times of the year and the company is wondering how it can ensure that it can still operate well at these times.

To address this problem the company has asked for an automated holiday booking system which can capture constraints such as:

- Holiday entitlement
- Minimum number of staff levels with particular skills
- Peak time balancing

The company envisages that the system could automatically enforce constraints when approving holiday requests. When it isn't possible to approve a holiday request because this would break a constraint, the system should suggest an alternative.

The company has a number of departments:

- Engineering
- Plumbing
- Roofing
- Carpentry
- Bricklaying
- Office

An employee works for a particular department, but in addition an employee will hold one of the following roles within their department:

- Head
- Deputy Head
- Manager
- Apprentice
- Junior member
- Senior member

Every department has only one head and deputy head but has multiple members of all other roles.

The system needs to record details of all employees, which includes an employee's personal details, as well as the date when an employee joined the company.

The holiday entitlement for an employee is based on how long he/she has worked at the company. All employees get 30 days of holiday per year. plus one extra day for every 5 years an employee has been with the company.

The following constraints have been identified by the company which apply at all times:

- No employee can exceed the number of days of holiday entitlement
- Either the head or the deputy head of the department must be on duty
- At least one manager and one senior staff member must be on duty
- At least 60% of a department must be on duty

There are some exceptions which are that none of these constraints apply between the 23rd of December to the 3rd January of every year.

In the month of August, the constraints are relaxed such that only 40% of a department must be on duty.

The following periods are considered peak times:

- 15th of July to 31st of August
- 15th of December to 22nd of December
- The week before and after Easter, which changes each year as Easter falls on a different day each year

When staff apply for holiday the system should prioritise staff who have had a lower number of holidays in the current year, followed by those who have fewer days in the peak time periods.

In the long-term *Straight Walls Ltd* wants to develop an app which its employees can install on a mobile phone or tablet. They are envisaging that employees could also use the app to request their holidays. The company is not sure yet what technology to use for the app, so for the prototype of the system, they want the app to simply be implemented as a desktop application, which would later be replaced with a mobile app. At this stage they are interested in getting sample data submitted via a web service and ensuring that the data is processed correctly. The desktop application could then be replaced by the app at a later stage without affecting the system.

There are strict requirements about the technologies and architecture to be implemented. These are detailed in the deliverables section

Deliverables

Part A (30%)

With the help of the tutor, organise yourselves into groups consisting of 2 students to engage in pair programming.

Implementation (10%)
As a team you are required to:

- Design/implement a database which meets the requirements of the case-study described above.

  You can use any relational database management system you wish (e.g. JavaDB, SQLServer, MySQL, Oracle, Access, etc.). The database should be normalised and use primary key and foreign key constraints.

- Create a set of entity/model classes which reflect entities from the database. Feel free to use LINQ or Entity Framework, if you wish.

- Implement a desktop application using C#, which offers the following functionality to admin users:
  - User log in
  - User management enabling an admin user to create/edit/delete employees. It must be possible to allocate an employee to a department and assign him/her a role.

As a group you should decide when you consider this part of the system to be completed, at which time it must be demonstrated to your tutor. ***This must be no later than week 7 of the module.***

Group report section (10%)
The group report section should contain the design documentation that you created as a group, which should contain an Entity Relationship Diagram outlining the database structure and an architecture diagram which shows the overall setup of the system. This should be prepared by the group and each member then includes the same diagrams in their final report (unless you make further modifications during the individual phase of the assessment).

Individual report section (10%)
Please include a reflection of your role within the team and discuss lessons learnt. What you think went well and what you think could have been improved and how. This section should be included in the final report.

Part B (70%)

For Part B the teams will be disbanded, and you are expected to work individually.

You can take a copy of the work completed as a group and build on it to complete the remaining implementation. This may involve exporting the database you developed as a team and importing it to your individual database account.

It is envisaged that the system will be developed in three stages, to include basic, intermediate and advanced functionality.

Implementation (60%)

Functionality A (10%)

Extend the functionality implemented as a group to add the following functionality for an admin user:

- View a list of outstanding holiday requests
- Accept/reject a request
- View a list of all holiday bookings and filter them by employee
- Select a date and show all employees working that day and those on leave that day.

When a holiday request is accepted or rejected it should be removed from the list of outstanding requests.

Functionality B (10%)

To facilitate the data entry of phone numbers, you should implement a Phone Textbox. Implement a component which inherits from the Textbox control. It should only allow numbers to be entered. Further, it should display the numbers that are entered into it in red if there are more than 11. The Phone Textbox component should be integrated into the application so it is used instead of a normal textbox when entering a phone number for a driver.

Create a web application using ASP.NET which allows employees to:

- Log in
- Submit a holiday request
- View a list of existing holiday requests, which also shows whether they were approved/rejected

Functionality C (10%)

You should extend the system by including a component which applies constraint checking.

When an employee submits a holiday request, the system should use the component to automatically check whether all constraints are satisfied. The list of outstanding requests shown to the admin user should be split into two, those which don't break any of the constraints and those which do and can't be accepted.

For requests which break constraints, the system should specify which constraint is being broken. There could be more than one, in which case they should all be listed.

Functionality D (10%)

Create a Web Service (SOAP or REST) which exposes functionality for employee log in and make holiday request submissions. This web service will be called from the employee app.

The prototype of the employee app should also be created, as a desktop application (or mobile app) and it must be possible for an employee to log in and to submit a holiday booking request.

The employee app should not connect directly to the database, but all communication must be via the Web Service (remember that the company is planning to replace the desktop application with a mobile app at a later stage).

Functionality E (10%)

Add more functionality to the web application.

Create a prioritisation component which prioritises requests based on the number of days already approved and the number of days requested during the peak times. The prioritisation should determine the order in which outstanding requests are listed.

Implement a component which suggests alternative dates (where possible) for requests which break constraints.

Functionality F (10%)

Create a visualisation component which makes it possible to visually depict a calendar which highlights the days on which an employee has booked leave.

Technical and User Documentation (10%)

Prepare a final report which should contain the design documentation and reflection from Part A as well as the following sections:

- **Screen shots:** Screen shots demonstrating each of the features that you have implemented. Give captions or annotations to explain which features are being demonstrated.

- **Evaluation (approx. 600 words):** An evaluation of the evolution of your application. You should discuss any problems you had during implementation. You should be critical (both positive and negative) of your implementation. Be prepared to suggest alternatives. Discuss how your final implementation could be improved.

- **Algorithm (approx. 600 words):** A complete implementation of the advanced functionality requires you to devise algorithms for predicting the likelihood of an incident. Outline how you implemented this algorithm and wherever possible support your discussion with pseudo code, equations and/or diagrams. If you didn't implement the component or you think your implemented algorithm could be improved, then you can critically discuss how you would have implemented/improved the algorithm.

Acceptance Test (0% but see below)

You must demonstrate your implementation to the company which is represented by your tutor who will sign off the system. Failure to demonstrate will result in a failed assessment. You will be allocated a time slot closer to the submission deadline.

Notes on the Groupwork

If you have problems identifying a colleague to work with, please inform your tutor, who will help you with this.

The two report sections (group and individual) must be included in your PDF with eh user and technical documentation.

Notes on the Implementation

You **MUST** upload a ZIP file containing all of your source code (i.e. the folders containing the Visual Studio projects). If the resulting file is too large, then you can delete compiled code and libraries, but don't remove any source code files.

You **MUST** clearly reference code borrowed from sources other than the lecture notes and tutorial examples (e.g. from a book, the web, a fellow student).

Notes on the User and Technical Documentation

The document should be submitted separately as a PDF document.

Notes on the Acceptance Test

You will demonstrate to your tutors. **If you miss your acceptance test you will automatically fail the coursework**, irrespective of the quality of the submission. If you have a legitimate reason for missing your acceptance test then you should wherever possible contact your tutor in advance and arrange an alternative time slot. It is entirely your responsibility to contact your tutor and arrange a new demo if you miss your slot for a legitimate reason.

You are expected to talk knowledgeably and self-critically about your code. If you develop your program at home it is your responsibility to make sure that it runs in the labs. You should allow yourself enough time to do this.

If you are unable to answer questions about the product you have developed, you will be submitted for plagiarism.

A schedule for coursework acceptance tests will be made available on the course website closer to the submission deadline.

Grading Criteria

Part A, which consists of the group work accounts for 30% and the implementation and user/technical documentation accounts for 70%. There are multiple functionalities which comprise the development of the system. Each functionality section contributes 10% to the overall grade. The mark for each is awarded taking into consideration the quality and completeness of the implementation, as well as the assessment criteria specified below. Just because you implemented a particular requirement doesn't mean that you automatically get the full marks. The full marks are only awarded if the requirement has been implemented to outstanding quality, including software design model, code quality, user interface, error handling, validation, componentisation, etc. A poorly structured, but working implementation of a requirement would attract a pass mark.

To achieve a pass (40%) you must have made a serious attempt to implement at least three functionality sections. The implementation must show some signs of attempting to focus on the assessment criteria given in this document. Relevant user and technical documentation must also be submitted, and the group work must be completed.

To achieve a 2:2 mark (above 50%) you must have made a serious attempt to implement at least four functionality sections. They must attempt to focus on the assessment criteria given in this document. Good user and technical documentation must also be submitted, and the group work must be completed.

To achieve a 2:1 mark (above 60%) you must have made a serious attempt to implement at least five functionality sections. They must address most assessment criteria given in this document. Very good user and technical documentation must also be submitted, and the group work must be completed.

To achieve a distinction (above 70%) you must implement all requirements to a very good level, or most to an excellent level, in accordance with the assessment criteria given in this document. Submit excellent user and technical documentation. Successfully meet the large majority of assessment criteria outlined below and the group work must be completed to a high standard.

To achieve a very high mark (90% and above) you must implement all implementation requirements to an outstanding standard in accordance with the assessment criteria given in this document. Submit outstanding user and technical documentation. Successfully meet all assessment criteria outlined below and the group work must be completed to a very high standard.

Assessment Criteria

The Groupwork (30%)

- The database design should be fully normalised and make appropriate use of primary and foreign keys
- The implementation of the entity/model classes should reflect the design documentation
- The design documentation should be created using standard notation and should clearly convey an overview of the system
- The individual reflection document should be clear, accurate, complete and concise. State any assumptions you have made.
- There should be critical reflection of the group work

The implementation (60%)

The following assessment criteria are used to determine the quality of your implementation and should be addressed in the development process:

- If you have incorporated features that were not explicitly asked for but which add value to the application, they may be taken into account if you draw our attention to them.
- The application should look pleasant and be easy to use.
- Code componentisation – does your code demonstrate low coupling and high cohesion? Have you avoided hard coded URL's and/or file paths (i.e. is your code stateless)? Have you reused external components? Have you minimised code duplication? How much impact would a further change of persistence medium have on your application?
- Quality of Design – how flexible is your application? How easy would it be to add in new functionality, or alter the presentation layer, or change the data source?
- Robustness of the application. Have you properly handled errors and validated input? Is there evidence of testing?
- Quality of code –
    - Is the code clear to read, well laid out and easy to understand?
    - Is the code self documenting? Have you used sensible naming standards?
    - Is your namespace structure logical?
    - Have you commented appropriately?

The user and technical documentation (10%)

- The document should be clear, accurate, complete and concise. State any assumptions you have made.

The acceptance test

- You should be able to demonstrate the implementation level achieved in a clear logical and unambiguous manner without encountering errors. You must be able to show knowledge of your code and design choices.