COMP1610 (2021-2022)	**Programming Enterprise Components**	**Contribution: 100% of course**
Course Leader: Dr Rafael Martinez-Torres	**Practical**	**Deadline Date: 26 April 2022**
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 **26/04/2022** using the link on the coursework Moodle page for COMP1610.
- 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 will be completed in a group.

Please read the entire 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

Holiday requests are manually approved by a Head of Department. However, when the system lists holiday requests, it should identify those when break constraints and shouldn't allow them to be approved. Also, 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, so requests should be listed in order of priority (highest priority first). If the approval of one request means that others now no longer fulfil the constraints, then the system would need to identify them and not allow them to be approved.

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

Deliverables

With the help of the tutor, organise yourselves into groups consisting of 4-6 students.

The successful implementation of an enterprise application requires teams of people with different roles to collaborate.

Within the group assign the following roles:

- Java back-end programmer
- User interface designer
- Database administrator
- Server administrator
- Project manager

Depending on the group size, you may have multiple members sharing the same role or members with multiple roles.

It makes sense to allocate roles based on experience, but as you will engage in work together, this is also an opportunity to learn from other members of the team, so don't worry if you don't yet have expertise related to your role at the start of the assignment.

Implementation (70%)

As a team you are required to implement the following functionalities:

Functionality A (10%)

- 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.

- Set up an Application Server which will allow you to deploy an enterprise application. This includes configuration files which change the session timeout to 15 minutes and setting the database up as a data source in the Application Server

- Implement a web application using Jakarta EE, 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.

The implementation should use entity classes and Session Beans for back-end functionality and JSP or JSF for the user interfaces.
You should secure access to this functionality, so that it is limited to an authenticated user.

Functionality B (10%)

Extend the functionality implemented as a group to add a separate section 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 also 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 D (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 E (10%)

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

The prototype of the employee app should also be created, as a Java desktop application, 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 F (10%)

Add the following functionality to the web application.

When a holiday booking request is created via the web service, this should trigger an alert to the administrator. You should implement a Message-Driven Bean which is called when a holiday request is generated. The web service should publish a message using JMS which is then received by the Message-Driven Bean. At a later stage, the Message-Driven Bean should send an automated email to the administrator, but for the time being, this should record the alert in the database.

Functionality G (10%)

Implement the prioritisation of 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.

Finally, add functionality to the system which suggests alternative dates (where possible) for requests which break constraints.

Technical and User Documentation (30%)

Prepare a report which consists of two parts (one completed as a group and one completed individually).

Part A (completed as a group) (10%):

This part of the report should be prepared by the group and each member then includes the same part A in their final report**.**

- **Design Documentation:** This 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.

- **Screenshots:** 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.

Part B (completed individually) (20%):

- **Research (approx. 600 words):** Jakarta EE makes it possible to create enterprise applications. Carry out some research and critically discuss what other technologies and frameworks are available and how they compare to Jakarta EE.

- **Individual Reflection (approx. 300 words):** 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.

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 parts (A & B) must be included in your PDF.

Notes on the Implementation

You **MUST** upload a ZIP file containing all of your source code (i.e. the folders containing the NetBeans projects).

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 Demonstration

Live Demonstration

A Live Demonstration (see table below) will be held weeks in advance to the final submission. As it takes place prior to final deadline, the questions will be regarding functionality completed to date.

You will demonstrate to your tutors who act as a client. **If you miss your slot you will automatically fail the coursework** – irrespective of the quality of the submission. If you have a legitimate reason for missing a demo then you should wherever possible contact your tutor in advance and arrange an alternative demo slot. It is entirely your responsibility to contact your tutor and arrange a new demo if you miss your demo 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 at Greenwich. 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 **live demonstrations** is to be scheduled prior to the submission deadline.

Deliverable	Date	Type	Group/Individual
Code Q & A (Live Demonstrations)	05/04/2022	On campus/MS Team (for students online)	Group
Report and Code	26/04/2022	Submission on Moodle	Individual
Video Demonstration	28/04/2022	Submission on Panopto	Group

Video submission

You will demonstrate the implemented product as group to your tutor by preparing a 10-minute video.

The video you upload on Panopto should follow this naming convention – "Surname1,Surname2, etc."
You should include the surnames of all group members of all group members in the video name, so we can easily identify the group.

Grading Criteria

The implementation accounts for 70% of the grade. This is further divided into separate functionalities, each carrying a total weight of 10%. The report accounts for 30% and is subdivided into two parts. Part A is completed as a group and accounts for 10% and part B is completed individually and accounts for 20%. The mark for each deliverable 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 for that category.

For the implementation and part A of the report, the same grade is given to all members of the group.

To achieve a pass (50%) you must have made a serious attempt to implement at least five functionality sections. It must show some signs of attempting to focus on the assessment criteria given in this document. A complete and relevant report must also be submitted.

To achieve a merit mark (60% and above) you must implement at least six functionality sections and parts of the advanced functionality to a very good level These must address most of the assessment criteria given in this document. Good technical and user documentation must also be submitted.

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

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 technical and user documentation. Successfully meet all assessment criteria outlined below.

Assessment Criteria

The implementation (70%)

- It is not necessary to completely implement a whole functionality level in order to implement all or parts of later levels.
- If you have incorporated component 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/package structure logical?
    - Have you commented appropriately?

- The implementation of the entity/model classes should reflect the design documentation

Technical and User Documentation (30%)

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

- The database design should be fully normalised and make appropriate use of primary and foreign keys

- The design documentation should be created using standard notation and should clearly convey an overview of the system