COMP1610 (2018/19)	**Programming Enterprise Components**	**Faculty Header ID: 300579**	**Contribution: 100% of course**
Course Leader: Dr Markus Wolf	**Practical**		**Deadline Date: Sunday 24/03/2019**

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 - Design, develop and deploy Jakarta EE applications, using technologies such as EJBs, JPA, Servlets, JSP, Web Services.
2 - Understand, implement and evaluate service-oriented architectures.
3 - Critically evaluate and use technologies and architectures for developing enterprise applications.

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 **Sunday 24/03/2019** 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 must be completed individually.

Please read the entire specification before starting your work.

Taxi Driver System

You have been approached by a taxi operator called *Speedy Taxis*. The company has a fleet of taxis which serve all of London and the surrounding area. The company employs a large number of drivers and has experienced a number of incidents and customer complaints which it is now trying address. In order to achieve this, it is planning to offer drivers various training courses and qualifications in an attempt to reduce the number of incidents and customer complaints. A system is planned which would record details of training a driver has attended and what qualification(s) they have obtained.

As some training and qualifications expire, the system should remind users when it is time to renew a training or qualification.

When a driver is involved in an incident, which was caused or aggravated by unprofessional behaviour on part of the driver, then he/she gets a warning and it is added to his/her disciplinary record. In addition to the training, the system should also keep these disciplinary records about the drivers.

In the long-term *Speedy Taxis* wants to develop an app which its taxi drivers can install on a mobile phone or tablet. They are envisaging that drivers would use the app to log the start of their working day, the end of their working day and the start/end of each journey. This would give *Speedy Taxis* a record of the drivers working hours and time actually spent driving. There should be a component in the system which accepts the work logs and processes them. 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, submitting it to component 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

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

Implementation (80%)

Basic Functionality (30%)

The basic functionality should consist of a web application which enables admin users at *Speedy Taxis* to manage taxi drivers and record their training and qualification. The information has to be saved in a database. 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. The web application must be created as a Java EE Web Application and you can implement it using JSP or JSF.

The application needs to provide the following functionality to an admin user:

- Log in
- Add/edit/delete drivers
- Add/edit/delete qualifications
- Add/edit/delete training types
- Schedule sessions for a training type
- Assign drivers to a training session and record the outcome (e.g. training completed successful)
- Search for a driver and see his/her profile including qualifications and training
- See a list of drivers who have qualifications/training expiring within the next 30 days

Currently there are two qualifications which are relevant to the taxi drivers, which are their driving license and geographical tests (there are 5 different geographical tests: Central London, North London, South London, East London, West London).

Training types which the company is currently offering include: advanced driving course, driving at night, cyclist awareness, reduce fuel consumption.

It must be possible for an admin user to view a list of drivers and for a particular driver show his/her qualifications and training. It should also be possible to get a list of drivers who have qualifications/training expiring within the next 30 days.

Intermediate Functionality (30%)

For the intermediate functionality, you should extend the basic system by including additional features.

Create a Log component which can receive the following four inputs:

1. Start of day log
2. End of day log
3. Start of journey log
4. End of journey log

It should calculate journey durations (when receiving an end of journey log) and total hours worked in a day (when receiving an end of day log). It should report an error if it receives a log in an unexpected sequence (e.g. an end of journey log when no start of journey log was received).
The component should be exposed through a Web Service (SOAP or REST), so that it can be called from the driver app. The prototype of the driver app should be created as a Java desktop application and it must be possible to submit the four types of log listed above. A driver must also be able to log into the application on start-up, so the application knows whose driver's logs are being submitted.

The driver app should not connect directly to the database or the Log component, 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).

Advanced Functionality (20%)

Amend the web application created for the basic and intermediate functionalities to make use of Enterprise Java Beans. Make use of Session Beans to encapsulate all of your business logic. Use Entity Classes to handle all interaction with the database.

Add more functionality to the web application created for the basic functionality. It should be possible to create entries in a driver's disciplinary record. This functionality should be added to the desktop application developed for the basic system. Examples of incidents are: accidents, abusive behaviour towards customers, speeding offences, aggressive/dangerous driving, running a red light.

Improve the desktop application, so that a Message Driven Bean is triggered when an incident is recorded about a driver, so that the driver is informed. The Java desktop application should subscribe to this Bean and display a simple message – e.g. "A new incident has been reported. The details are as follows: [details of the incident]". The message could be displayed in a label or status bar.

Finally, a Message Driven Bean should be used to inform drivers when their training/qualification is due to expire (within 30 days).

Technical and User Documentation (up to 20%)

The report should consist of the following sections:

- **Section 1:** Design documentation of the system that you built. You should use graphical notation appropriately (e.g. UML diagrams). If your designs have changed since you uploaded them to the logbook, please explain what you changed and why.

- **Section 2:** Screen shots demonstrating each of the features that you have implemented. Give captions or annotations to explain which features are being demonstrated.

- **Section 3:** 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.

- **Section 4:** Java 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 Java EE.

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 Technical and User Documentation

The report should be submitted separately as a PDF document.

Notes on the Acceptance Test

You will demonstrate to your tutors. **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 acceptance tests will be made available on the course website closer to the submission deadline.

Grading Criteria

The user and technical documentation accounts for 20% and the implementation for 80%. There are three stages to the development of the system. The basic functionality carries a total weight of 30%, the intermediate one also 30% and the advanced one 20%. 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 (50%) you must have made a serious attempt to implement the basic functionality. It must show some signs of attempting to focus on the assessment criteria given in this document. Relevant technical and user documentation must also be submitted.

To achieve a merit mark (60% and above) you must have made a serious attempt to implement at least the basic and intermediate functionality. They must show some signs of attempting to focus on 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 a very good level, or most 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 (80%)

- 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?

Technical and User Documentation (20%)

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