COMP1690 (2017/18)	**Programming Distributed Components**	**Faculty Header ID: 300386**	**Contribution: 80% of course**
Course Leader: Dr Markus Wolf	**Practical Assignment**		**Deadline Date: Tuesday 27/03/2018**
This coursework should take an average student who is up-to-date with tutorial work approximately 40 hours Feedback and grades are normally made available within 15 working days of the coursework deadline			
Learning Outcomes: 1. Critically evaluate the notion of a component and the forces requiring component design 2. Critically compare distributed programming technologies and assess their applicability at large and small scale 3. Demonstrate, design, implement and deploy software components using one or more of the studied technologies			

Coursework Submission Requirements

- An electronic copy of your work for this coursework must be fully uploaded on the Deadline Date of **Tuesday 27/03/2018** 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 coursework must be completed individually.

Please read the entire coursework specification before starting your work.

Journey Planning System

You have been approached by a company called *Live Journeys*. They are planning to develop a journey planning system which enables a user to plan a train journey. They have enlisted you to develop a prototype of a system which will enable an admin user to set up a layout of the train network, including names of train lines, stations on each line, distance between a station and its adjoining station on the line and also stations where two or more lines cross.

For the prototype the company is not requesting that the actual train network is accurately captured, but it is more interested in getting the algorithms and journey planning functionality correctly implemented. So you can make up lines, stations and distances.

The intention is that the journey planning system will be accessible from a range of devices and *Live Journeys* therefore wants you to create a Web Service so the journey planning functionality can be exposed to a variety of clients using different devices and operating systems.

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 desktop application which enables admin users to set up the train network. The information has to be saved in a database. You can use any relational database management system you wish (e.g. SQLServer, MySQL, Oracle, Access, etc.). The database should be normalised and use primary key and foreign key constraints. The desktop application should be created as a Windows Forms application using C#.

The application needs to provide the following functionality to a user:

- Log in
- Add/edit/delete train lines
- Add/edit/deleted train stations

- Add/remove train stations to a line
- Set the distance between two stations

When setting up the train network it must be possible to add a station to two separate lines. In effect this creates a junction where a customer could move from one line to another.

To facilitate the data entry of distances, you should implement a Distance 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 they exceed 50. The Distance Textbox component should be integrated into the application so it is used instead of a normal textbox when entering the distance between two stations

It must be possible for an admin user to view a list train lines and for a particular train line show what stations are on the line, in what order and show the distance between a station and the next.

Intermediate Functionality (30%)

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

Develop a component which calculates the shortest route (in distance) between a given start and destination. The most basic implementation would only allow travel on the same line, so no changing of trains. However a more complete implementation would allow travel to anywhere in the train network. Please note that this component doesn’t take into consideration the times at which trains run, it only calculates the shortest distance.

A SOAP Web Service must be created which supports searching for a journey, providing the start and destination stations. The search should return the distance between the two stations in miles or kilometres, as well as details of required changes, if there are any. The web service should use the distance calculation component you developed.

To allow a user to query the journey planner system, implement a ASP.NET web application using C# which should connect to the SOAP web service to perform a search.

Advanced Functionality (20%)

Amend the distance calculation component you implemented for the intermediate functionality so that it also includes the time it takes to make a journey, not only the distance.

To simplify the calculation, you can make an assumption that trains on all lines go every 10 minutes and that changing a train just adds 15 minutes to a journey. A user should be able to see an estimated time of arrival and a breakdown of when one arrives and departs when changing trains.

Create a visualisation component which makes it possible for a user to visualise the train network. You can decide whether you want to develop the visualisation component to be integrated into basic desktop application or the intermediate web application.

Finally, make the implementation of the distance calculation component more robust by adding delays to particular sections of the network. A delay should be added to the link between two stations and the component should take delays into consideration when calculating the fastest route.

Technical and User Documentation (20%)

The document should consist of the following sections:

- **Section 1 (5%):** Design documentation of the system that you built. You should use graphical notation appropriately (e.g. ERD/UML diagrams).

- **Section 2 (5%):** Screen shots demonstrating each of the features that you have implemented. Give captions or annotations to explain which features are being demonstrated.

- **Section 3 (5%):** 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 (5%):** A complete implementation of the advanced functionality requires you to devise algorithms for calculating the shortest and fastest routes. Outline how you implemented these algorithms 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 *Live Journeys* which is represented by your tutor who will sign off the system. Failure to demonstrate to the representative will result in a failed assessment. You will be allocated a time slot closer to the submission deadline.

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).

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

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 (40%) you must have made a serious attempt to implement at least the basic functionality. It 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.

To achieve a 2:2 mark (above 50%) you must have made a serious attempt to implement at least the basic and some of the intermediate functionality. They must show some signs of attempting to focus on the assessment criteria given in this document. Good user and technical documentation must also be submitted.

To achieve a 2:1 mark (above 60%) 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. Very good user and technical 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 user and technical 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 user and technical documentation. Successfully meet all assessment criteria outlined below.

Assessment Criteria

The implementation (80%)

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 (20%)

- 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.