COMP1690 (2015/16)	**Programming Distributed Components**	**Header ID 227796**	**Contribution 100% of course**
Course Leader Dr Markus Wolf	**Release Date Monday 18/01/2016**		**Deadline Date Sunday 10/04/2016**
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: A, B, C			

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. Details are also on the Student Intranet.

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 electronic 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 by midnight on the Deadline Date of **Sunday 10/04/2016** using the link on the coursework Teachmat page for COMP1690.
- For this coursework you must submit a single Acrobat PDF document.
  In general, any text in the document must not be an image (ie must not be scanned) and would normally be generated from other documents (eg MS Office using "Save As .. PDF"). More details are on the IT Support pages . 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 (current values are on TeachMat and the Student Intranet).
- 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 be marked online and comments on your work and a provisional grade will be available from the Coursework page on Teachmat. A news item will be posted when the comments are available, and also when the grade is available in BannerWeb.
- You must NOT submit a paper copy of this coursework, or include the Banner header sheet.
- 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.

This assignment consists of two parts:

- **Part A:** an online quiz, which will take place in the lab on 05/04/2016
- **Part B:** implementation of an online restaurant advertisement system and accompanying report

Part A (10%) – Online Quiz

You are required to complete an online quiz during the lab session on 05/04/2016. The quiz is timed and consists of a number of multiple choice questions on material covered on the course.

At the end of the quiz, your result is automatically calculated and displayed on the screen. You MUST print screen the page showing the score, save it and include it in the report for Part A.

Part B (90%) - Online Restaurant Advertisement System

You have been approached by a company called *Food2U*. They are planning to develop a web system that can help customers find a restaurant close to a particular postcode in England. They have enlisted you to develop a prototype of a system which will enable company employees to add restaurants to the system and customers to search for restaurants.

The system will be used by two types of users:

- Company employees
- Customers

The system needs to provide the following functionality to each type of user:

- Company employee
  - Manage restaurants – adding new restaurants, as well as editing or deleting existing ones
- Customers
  - Register an account
  - View restaurants close to home address
  - Search for a restaurant by postcode and view results
    - as a list (ordered by proximity - closest first)
    - as a visually arranged ads page (closest restaurants at the centre and arranged outwards based on proximity)

In order that only actual restaurants are added to the system, the company decided that it would be the company employees who manage the restaurant data. At a minimum, the following data is recorded for each restaurant: restaurant name, postcode, address, type of restaurant (e.g. Vegetarian, Italian, Indian, etc.), telephone number, email address (optional field), website address (optional field), average price of main dish, image (logo or business card), ad type (single, double vertical or double horizontal).

A customer needs to register an account in order to search for restaurants. The system should capture at least the following details during the registration process: first name, surname, email address (also used as username), password, gender, date of birth, phone number, postcode and address.

A registered user should be able to view a list of restaurants close to the customer's home address. It should also be possible to enter a postcode and see a list of restaurants within close proximity. A simple search algorithm would just match the first part of the postcode, but a more refined version measures actual distances between the postcode given and the postcodes of restaurants, and lists them in order of proximity. Restaurants should be presented in a list and it should be possible to view full details by clicking on a particular restaurant.

In addition to viewing restaurants in a list, it should be possible to view them arranged as a virtual notice board where the logos/business cards are positioned in a grid, based on their proximity to the postcode provided. If a user searches for postcode, for instance SE10 9LS, then the closest restaurant should be displayed at the centre of the page, the eight next closest ones would be arranged around it, then the next 16 closest, etc. Please see figure 1 below for an example of how spaces would be arranged based on the proximity of the restaurant's postcode and the one entered by the user. In this case, the closest restaurant to SE10 9LS is 1.2 kilometres away (shown here as a green field), the 8 spaces surrounding the green field (shown here in yellow) contain the next 8 closest ones and surrounding that are the following closest 16 (shown here in red).

2.9km	3km	3.2km	3.5km	3.5km
3.5km	2km	2km	2.5km	3.8km
3.9km	2.6km	1.2km	2.6km	3.9km
4km	2.7km	2.8km	2.9km	4.1km
4.2km	4.2km	4.3km	4.5km	4.8km

Figure 1 - Example notice board layout

Instead of just showing the distance as in figure 1, each space should also display the restaurant logo or business card. A user should be able to get the full details for a restaurant by clicking on it. It should also be possible for the user to set how many fields are displayed on the notice board (e.g. the example shown here has 25).

A more advanced implementation of the notice board includes restaurants that have a logo or business card which spans two horizontal or vertical spaces. This means that they would occupy two spaces next to each other. See figure 2 for an example where the double ads are shown in bold, italics and underlined.

2.9km	2.9km	3km	***3.2km***	
3.5km	***2km***	2km	2.5km	3.5km
3.5km		1.2km	2.6km	3.8km
	2.6km	2.7km	***2.8km***	
3.9km	3.9km	4km	4.1km	4.2km

Figure 2 - Example notice board layout including double spaces

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

Deliverables

Implementation (80%)

1) **Database (10%):** The database needs to be used for persistent storage of restaurants and their details. For more information on what data should be stored in the database, please see the description above. 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.

2) **Windows Forms Application for Staff (10%):** Create a Windows Forms application using C# that allows a member of staff to add, edit and delete restaurants, including an image of the logo or business card. It should also be possible to view restaurants in a list and apply some search filters. The application should use the database outlined in requirement 1.

3) **Price TextBox and Optional Component (10%):** 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 price textbox component should be integrated into the application defined in requirement 2 to be used for the average price of a main dish. Also, choose any other control of your choice and extend it to add some custom functionality and apply the new component in your implementation of requirement 2.

4) **Customer Registration and Search Web Application (10%):** This is the public-facing user interface of the system. The web application should be implemented in ASP.NET using C#. It should allow a user to register a new account. An authenticated customer should be able to view a list of restaurants closest to his/her home address. Furthermore, a customer should be able to search for restaurants by postcode and search results should be displayed as a list. Full details of a restaurant should be shown when a user clicks on one of the search results. The web application should also use the database outlined in requirement 1.

   UK postcodes consist of two parts - e.g. SE10 9LS, where the first part identifies the general area and the second part a small section within that area (e.g. a road or part of a road). A simple implementation of the search could compare the first part of the postcode and show results where the first part matches. For a more complex implementation, see requirement 6 below.

5) **Virtual Notice Board (10%):** Expand the search functionality implemented for requirement 4 to include the option to display the search results graphically as a grid of restaurant ads (logos or business cards). This would be an alternative to presenting the results as a list. Clicking on a

logo/business card should display the full details of the restaurant. It should be possible for a user to specify how many restaurants should be displayed on the virtual notice board.

6) **Distance Calculator Component (10%):** Implement a component that calculates the actual distance between the postcode that the user searched for and that of the restaurants. You should use a third-party service to obtain the distance or to aid the calculation of the distance. For instance, you can obtain the latitude and longitude coordinates for any UK postcode by calling the RESTful web service at http://www.uk-postcodes.com/ which returns geographical details of a postcode in XML or JSON format. You can then use the formula described by http://www.movable-type.co.uk/scripts/latlong.html in order to calculate the distance between the latitude/longitude coordinates. You do not have to use this approach, but can decide to research your own way of measuring distance between postcodes.

   The distance calculator component should be integrated into the web application developed for requirements 4 and 5. Search results should be shown ordered by distance (closest first) and the distance to a restaurant should be displayed in the search results.

7) **Ad Spread Component (10%):** Devise an algorithm that arranges restaurants on the virtual notice board according to their distance to the postcode that the user searched for. How this should be implemented is described on page 4. In order to implement this requirement you need to extend your implementation of requirement 5 and use the Distance Calculator Component developed for requirement 6.

8) **Double Ads (10%):** Extend your implementation of requirements 5 and 7 by supporting double-sized ads. These can be horizontal (occupy two spaces one next to the other) or vertical (occupy two spaces one below the other). This should not result in empty spaces, but other ads need to be arranged around the double ads.

Written Report (10%)

The report should provide sufficient evidence that the requirements outlined in the course specification have been met.

The report should consist of the following sections:

- **Section 1 (2.5%):** Design documentation of the system that you built. You should use graphical notation appropriately (e.g. ERD/UML diagrams).

- **Section 2 (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 (2.5%):** 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 (2.5%):** Complete implementations of requirements 6, 7 and 8 require you to devise algorithms for calculating distances and arranging spaces according to proximity. Outline how you implemented these algorithms and wherever possible support your discussion with pseudo code, equations and/or diagrams. If you didn't implement a particular implementation requirement, or you think your implemented algorithm could be improved, then you can critically discuss how you would have implemented/improved the algorithm.

- **Section 5 (0%):** You are required to include a screenshot of your online quiz result in the report. The screenshot doesn’t carry any marks, but if you fail to include it you may not be awarded the marks for Part A – the online quiz.

Demo (0% but see below)

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 – the only exception being the packages folder which can be removed to reduce file size).

You **MUST** clearly reference code borrowed from sources other than the lecture notes and tutorial examples, such as examples a book, the web, a fellow student, etc.

Notes on the Report

The report should be submitted separately as a PDF document.

Notes on the Demo

You will demonstrate to your tutor. **If you miss your demo 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 the University. 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 demonstrations will be made available on the course website closer to the submission deadline.

Grading Criteria

The online quiz accounts for 10%, the report for 10% and the implementation for 80%. There are eight implementation requirements, each of which carries a total weight of 10%.

The mark for each implementation requirement 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 10%. 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 4-5%.

To achieve a pass (40%) you must have made a serious attempt to implement at least four requirements. They must show some signs of attempting to focus on the assessment criteria given in this document. A relevant report must also be submitted and you must have attempted the online quiz.

To achieve a 2:2 mark (above 50%) you must have made a serious attempt to implement at least five of the requirements. They must show some signs of attempting to focus on the assessment criteria given in this document. A good report must also be submitted and you must achieve a passing grade in the online quiz.

To achieve a 2:1 mark (above 60%) you must have made a serious attempt to implement at least six of the requirements. They must show some signs of attempting to focus on the assessment criteria given in this document. A very good report must also be submitted and you must achieve a good grade in the online quiz.

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 an excellent report. Successfully meet the large majority of assessment criteria outlined below. Achieve a very good result in the online quiz.

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 an outstanding report. Successfully meet all assessment criteria outlined below. Achieve an excellent result in the online quiz.

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:

- 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 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 report (10%)

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

The Online Quiz (10%)

- The online quiz is marked automatically and takes into consideration how many questions have been answered correctly. Questions that are partly correct are not awarded any grades (e.g. if you asked to choose two correct answers in the multiple choice question and you only identify one correctly then no partial score is given).

The Demo (0%)

- 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.
- The demonstration will take place after submission. A timetable with demonstration slots will be published on the course website closer to the date.