ICT321

Architecture and System Integration

Task 2

Semester 2, 2019

Assessment and Submission Details

Marks: 40% of the Total Assessment for the Course

Due Date: 11:59pm Friday, Week 13

Task 2 consists of a demo code package and a technical report for the integration plan. Submit your work as a **.zip** file, which contains the **code files (another .zip file)** and your **report (a .docx file)**, to Blackboard Task 2. Please follow the submission instructions described in this document.

The task is an individual assignment and will be marked out of a total of 100 marks and forms 40% of the total assessment for the course. **ALL** assignments will be checked for plagiarism by SafeAssign system provided by Blackboard automatically.

Refer to your Course Outline or the Course Web Site for a copy of the "Student Misconduct, Plagiarism and Collusion" guidelines. Academic Integrity Information.

Assignment submission extensions will only be made using the official USC Guidelines.

Requests for an extension to an assignment **MUST** be made to the course coordinator prior to the date of submission and requests made on the day of submission or after the submission date will only be considered in exceptional circumstances.

Case Study: Ryanair – Technology Strategy

After evaluating various platforms, Ryanair finally decided to adopt a Service Oriented Architecture (SOA) for its future IT infrastructure. Ryanair executives were impressed with your Template based, Reference Architecture approach set out in your Task 1 Report and have accepted your recommendations regarding IT Governance and Systems and Data Sustainability. However, Ryanair Executives would like to gain a deeper understanding of:

- Computing and storage infrastructure design,
- Information integration,
- Application and Service Integration, and
- Technologies behind Application and Service Integration including a small specific demonstration of these technologies along with a brief explanation of the concepts and principles of how it works.

There are 2 parts to this Assignment:

Part 1 - Integration Demo (40%)

Ryanair understand planning European aircraft flight paths. Your Integration Demo will demonstrate a basic flight planning microservice using service-oriented architecture. You will use a Google Maps Mashup to plot an aircraft route (including a number of 'stop-overs'). Your Google Maps Mashup will capture the endpoint coordinates (along with the name for the aircraft route) and send them to your microservice for storage. You will also be able to recall a stored aircraft route from your microservice and replot it on your Google Map Mashup. **Please follow the instructions carefully.** There are two major components in the demo system:

1. **RESTful Web service server demo**. In this demo, you are required to build a RESTful Web service which supports two services:
    - 'saveRoute' query from the client browser similar to '/saveRoute?route=[xml object]'. The server Web Service will accept an XML object with similar format to Figure 1. As part of the python service you are required to:
        i. accept the XML object and extract the route name which will become an xml file name (e.g. 'Route 1.xml') and stored on the server,
        ii. before saving the .xml file you are required to extract the 'latitude' and 'longitude' coordinates from the 'latlng' element and create two new elements alongside as siblings to each 'latlng' element to store the individual latitude and longitude coordinates,
        iii. Store the adjusted .xml file using the Route name as the file name.
    - 'getRoute' query from the client browser similar to '/getRoute?route=[route name]' will retrieve the aircraft route information from the server:
        i. The route name will be accepted, and the corresponding xml file will be retrieved and returned as an XML object to the Google Maps Mashup.

The Python Bottle framework is required for this implementation. A Python scrip file with the name '***microservice.py***' will contain this RESTful Microservice Demo.

```
<?xml version="1.0"?>
- <routes>
  - <route name="Route 1">
    - <routeleg>
        <pos>1</pos>
        <lnglat>(-26.577049545101413, 152.91140996704098)</lnglat>
      </routeleg>
    - <routeleg>
        <pos>2</pos>
        <lnglat>(-26.65992168231015, 152.86128484497067)</lnglat>
      </routeleg>
    - <routeleg>
        <pos>3</pos>
        <lnglat>(-26.690599812151657, 152.9666849304199)</lnglat>
      </routeleg>
    - <routeleg>
        <pos>4</pos>
        <lnglat>(-26.61910698885558, 153.01406347045895)</lnglat>
      </routeleg>
    - <routeleg>
        <pos>5</pos>
        <lnglat>(-26.570908479540783, 152.98488103637692)</lnglat>
      </routeleg>
  </route>
</routes>
```

Figure 1 - Suggested XML object format received from the Google Maps Mashup

2. **Mashup demo**. Ryanair operates around Europe so your initial Google Map will be centred on Europe. In this demo, you are required to build a html Europe focussed Google Maps Mashup application which accepts a route name as the input. There three functions (buttons on the html page) are required after the html page is initiated with a raw Europe centred map:
    - **Function 1** - Clicking on the Map will create markers and polylines between markers as you continue selecting locations. These polylines between markers will simulate an aircraft route for which you will input a name, collect the coordinates and their order into an XML object, and send to your microservice via your AJAX web service.
    - **Function 2** – you will need to be able to clear the map back to its raw newly initiated state.
    - **Function 3** – you will be able to type a route name into the input and retrieve a route from your microservice and populate the map with the saved route:
        i. This function will retrieve an XML object from your microservice via another AJAX web service.
        ii. once your AJAX service receives the XML object it will need to extract the individual latitude and Longitude coordinates from the XML object and push them into the MVCArray.

- A HTML file '***plan_route_map.html***' should be implemented. In this file, a text field is provided at the page top to accept user's route name input. In addition, three buttons are required to implement the three functions.

Very Important Note:

For ease of demonstration to Ryanair executives, your code must be self-contained and run easily in PyCharm. In addition to using Standards based HTML, JavaScript and Python, the Python 'Bottle' module is the only additional framework you should use. ALL required files must be provided with your submission and should not require any extra installation.

HINT: Most of the code for this assignment is demonstrated in the workshops.

Code Package Format

The code implementation should be with Python 3 and PyCharm. You are required to compress all your Python code, data files and all supporting packages/modules into **one .zip file** and upload it to Blackboard. Make sure your demo code is runnable in PyCharm after being unzipped.

Part 2 – Integration Report (60%)

In addition to the above Demo code (40%) you are required to include a Report (60%) Based on the above case study explaining the key concepts for:

- The demo design and implementation (including instructions on running your code).
- Information Management and Integration
- Application and Data Storage Infrastructure design, and
- Application and Service Integration

Please use the following outline in your report:

	Title	**Major Outcomes**
1	Introduction	
2	Information Management and Integration	Discuss and design strategies to integrate the multiple data sources developed and used by different teams for various business units and departments.
3	Application and Data Storage Infrastructure Design	Discuss various cloud options and their suitability. Design a cloud infrastructure strategy.
4	Application and Service Integration	There are many different types of applications and services coexisting in Ryanair, including legacy applications, internal Web services, and external Web services. Some external services are provided by business partners and others are from public service providers such as Google Map and Google Search. Discuss and design service and application integration strategies.
5	RESTful Web services	Explain RESTful Web services and how they were applied in your Demo
6	Mashups	Explain Mashup principles and how it was applied in your Demo
7	Conclusion	

Consult Rubric in Appendix A for more details on how this question will be marked.

Report Format

Your report should be no less than 1500 words and it would be best to be no longer than 2000 words.

The report MUST be formatted using the following guidelines:

- Paragraph text – 12 point Calibri single line spacing
- Headings – Arial in an appropriate type size
- Margins – 2.5cm on all margins
- Header – Report title
- Footer – page number (including the word “Page”)
- Page numbering – roman numerals (i, ii, iii, iv) up to and including the Table of Contents, restart numbering using conventional numerals (1, 2, 3, 4) from the first page after the Table of Contents.

- Title Page – Must not contain headers or footers. Include your name as the report's author.
- The report is to be created as a single Microsoft Word document (version 2007, 2010, 2013 or Office 365). No other format is acceptable and doing so will result in the deduction of marks.

Please follow the conventions detailed in:

Summers, J. & Smith, B., 2014, *Communication Skills Handbook*, 4th Ed, Wiley, Australia.

Submission

Submit your work as a zip file to Blackboard Task 2 by the due date of **11:59pm Friday, Week 13**. Inside the zip file, a zip package of code and a Word file of report should be included.

The assignment will be assessed according to the marking sheet. Late submission will be penalised according to the policy in the course outline. Please note Saturday and Sunday are included in the count of days late.

Assignment Return and Release of Grades

Assignment grades will be available on the course website in two weeks after the submission. An electronic assignment marking sheet will be available at this time.

Where an assignment is undergoing investigation for alleged plagiarism or collusion the grade for the assignment and the assignment will be withheld until the investigation has concluded.

Assignment Guidelines

This assignment will take a number of weeks to complete and will require a good understanding of application and information integration technologies for successful completion. It is imperative that students take heed of the following points in relation to doing this assignment:

1. Ensure that you clearly understand the requirements for the assignment – what has to be done and what are the deliverables.
2. If you do not understand any of the assignment requirements – Please **ASK** the lecturer or your tutor.
3. Each time you work on any aspect of the assignment reread the assignment requirements to ensure that what is required is clearly understood.

Appendix A

ICT321 Task 2 - Systems Integration and Report Rubric

Criteria	High Distinction (85-100%)	Distinction (75-84%)	Credit (65-74%)	Pass (50-64%)	Fail (0-49%)
(20%) Create a Python script to implement a working RESTful Web service microservice server	**Skillful and proficient** RESTful Web service server will listen on a TCP port and accept, process and serve XML data with no errors. Input error handling. Very logically set out code with clearly documented code.	**Proficient and effective** RESTful Web service server will listen on a TCP port and accept, process and serve XML data with no errors. Input error handling.	**Effective and competent** RESTful Web service server will listen on a TCP port and accept, process and serve XML data but with minor errors.	**Competent RESTful Web** service server will listen on a TCP port and accept, process and serve XML data with but with some errors.	**Not competent** RESTful Web service server will not listen on a TCP port and / or not accept and / or serve data. Major errors
(20%) Create a HTML script to implement a Mashup from two data sources including the RESTful Web service server.	**Skillful and proficient** HTML script will access, process and show data from two sources no errors. Well set out HTML page with intuitive functionality Clearly documented code.	**Proficient and effective** HTML script will access, process and show data from two sources no errors. Sufficient error handling and easy functionality	**Effective and competent** HTML script will access, process and show data from two sources but with minor errors. Effective functionality.	**Competent** HTML script will access, process and show data from two sources with errors. limited but sound functionality.	**Not competent** HTML script has major errors when accessing and processing data.

Criteria	High Distinction (85-100%)	Distinction (75-84%)	Credit (65-74%)	Pass (50-64%)	Fail (0-49%)
(15%) Information Integration and Management - Discuss and design strategies to integrate the multiple data sources developed and used by different teams for various business units and departments.	**Skillful and proficient** Clear definition of Enterprise Information Integration. clear elaboration of the three data integration methods with explanations (including advantages and disadvantages of each) and examples of how each method could be applied in the case study.	**Proficient and effective** Definition of Enterprise Information Integration. Clear elaboration of the three data integration methods with clear explanations of how each method could be applied in the case study.	**Clear and coherent** Elaboration of the three data integration methods with some discussion and reference to the case study.	**Coherent** Sound explanation of more than one information integration strategy and how they relate to the case study.	**Imprecise and vague** Discussed one or two data integration methods with basic or no application to the case study.
(15%) Application and Data storage infrastructure design - Discuss various cloud options and their suitability. Design a cloud infrastructure strategy.	**Skillful and proficient** Clear explanation of cloud infrastructure. Clear identification and elaboration on the types of cloud infrastructure (including advantages and disadvantages of each) and how each cloud type could be applied to the case study. Clear cloud strategy recommendation.	**Proficient and effective** Explanation of cloud infrastructure. Identification and elaboration on the types of cloud infrastructure and how each cloud type could be applied to the case study. Some cloud strategy recommendation.	**Clear and coherent** Identification and elaboration on at least three types of cloud infrastructure and how each type could be applied to case study.	**Coherent** Identification and explanation of at least two types of cloud infrastructure and how they relate to the case study.	**Imprecise and vague**

Criteria	High Distinction (85-100%)	Distinction (75-84%)	Credit (65-74%)	Pass (50-64%)	Fail (0-49%)
(15%) Application and service integration - Discuss and design service and application integration strategies.	**Skillful and proficient** Clear explanations and examples of the three main methods of Enterprise Application Integration (EAI) including the advantages and disadvantages of each. Clear recommendations on how EAI can be applied in the case study.	**Proficient and effective** Explanations of the three main methods of Enterprise Application Integration including the advantages and disadvantages of each Clear application to the case study.	**clear and coherent** Explanations of the three main methods of Enterprise Application Integration and applied to the case study.	**Coherent** Basic explanations of Enterprise Application Integration with reference to the case study.	**Imprecise and vague**
(7.5%) Explain RESTful Web services and how they were applied in your Demo.	**Perceptive** An accurate and logical explanation of RESTful Web services and how they could apply in other applications as well as applied to the demo code. Clear instructions on how the demo code should be executed.	**Specific** An accurate and logical explanation of RESTful Web services and applied to the demo code with instructions on how the demo code should be executed.	**Logical** An accurate and logical explanation of RESTful Web services and applied to the demo code.	**General** A basic but correct explanation of RESTful Web services.	**Ambiguous** Little or no explanation of the principles or application.

Criteria	High Distinction (85-100%)	Distinction (75-84%)	Credit (65-74%)	Pass (50-64%)	Fail (0-49%)
(7.5%) Explain Mashup principles and how it was applied in your Demo.	**Perceptive** An accurate and logical explanation of Mashup principles and how they could apply in other applications as well as applied to the demo code. Clear instructions on how the demo code should be executed.	**Specific** An accurate and logical explanation of Mashup principles and applied to the demo code with instructions on how the demo code should be executed.	**Logical** An accurate and logical explanation of Mashup principles and applied to the demo code.	**General** A basic but correct explanation of Mashup principles.	**Ambiguous** Little or no explanation of the principles or application.