



Complete, fold and seal
(see note 2 below)

Cahill

FORENAME(S):

Leah

Instructions to Candidates

1. Please give the information requested in the boxes below.
2. Enter your name in the space provided to the right, fold over and seal down (using the label provided) to hide your name.
3. The answers must be legibly written. The Examiners can accept or ignore illegible script or have it typed and the expense of typing charged to the candidate.
4. Write your answers in ink.
5. Begin each answer on a fresh page. Do all rough work in the extra book and cross it through.
6. No part of this book is to be torn out; it must be handed in complete.
7. If more books than one are used, fasten them together using the tags available from the invigilator(s).
8. **Every book must be handed in. In no circumstances may answer books, used or unused, be taken out of the examination venue.**

1	Student Number	P140092
2	Title of Examination Paper	Emerging Web Tech.
3	Module Code	*M6X01096
4	Date and Time of Examination	12pm 9th May 2018
5	Examination Venue	Shoreda Gym
6	Seat Number	C060
7	Number of Answer Books handed in	

List the numbers of the questions you have attempted	Office Use
1	13
2	18
3	21.5
Total	70%

$$\begin{array}{r} 52\frac{1}{2} \\ \hline 75 \end{array}$$

70%

GT 1/6/18

1a) "A class is an OOP concept used in the code as follows:
 class Position { A class is where the methods and
 } "properties are defined: Template of
 an object."

There is also an abstract class:

"Abstract Class Shape implements IDrawable { }"

IDrawable is an example of how interfaces are used in typescript. ~~the `draw()` method is not defined in the `IDrawable` interface, but it is defined in the `Shape` class.~~

The `Shape` class implements the `IDrawable` interface, which it then implements the methods defined in `IDrawable` which is `draw(): void`. The key word for implementing the interface is "implements".

Inheritance is a concept in OOP where a child class inherits methods from a parent class. In this code, the key word "extends" is used to allow the class to inherit methods from the parent class (which is `Shape` here) and the method `draw(): void` it also has inherited the properties from `Shape`. The child class is `Rectangle`.

b) class Picture extends Shape {
 constructor (el: JQuery, position: position Z)
 { super(el, position);
 }
~~draw() {~~

~~draw() {~~ draw(item: string) {
~~item = [];~~ item = [];
~~}~~ }

The class would have to use polymorphism to override the `draw` method so that it has a return of an item ~~draw~~ rather than void, so it can draw.
 it would need a new private attribute that is =
 private item: String;

1 c.)

XML =

```

<Employee>
  <name> Fred Blogs </name>
  <department> Engineering </department>
  <roles> Designer </roles>
  <roles> Programmer </roles> <roles> tester </roles>
  <email> fred.blogs@company.com </email>
  <telephone> 01523847983 </telephone>
</employee>

```

JSON =

```

Employee = [{ "name": "Fred Blogs",
  { "department": "engineering",
  { "roles": ["Designer", "programmer", "Tester"],
  { "email": "Fredblogs@company.com",
  { "telephone": "01523847983" } } } } ]

```

2.)
a)

REST is a standardized set of principles or architecture which enables the transfer of data through HTTP Request and Response. REST is stateless. So a response is sent within the same connection as the request and it doesn't store history so it is quick and efficient. REST provides a one-to-one mapping to CRUD operations to create, update, delete and read data. The GET method is used to retrieve and read a specified resource. The POST is used to create a resource. The PUT method is used to update or modify a resource and DELETE is used to delete a resource. REST uses HTTP methods explicitly for the request and responses of data. REST also has directory-like URLs so it is more human-readable than the SOAP architecture.

AJAX enables data to load on a part of a web page without reloading the entire page. AJAX can also send data to the back end.

REST Services have a low overhead and are quick in sending and receiving data. REST is relatively easier to implement compared to the SOAP architecture. 2.5
 AJAX enables interactivity in an efficient way and is faster than reloading a whole page.

2b) Their solution works with a proxy because of the "Same origin Policy" problem. SOP restricts access to another website's methods and properties. To overcome SOP, a server side proxy must be implemented to enable Cross Origin Post. 2.5
 If they want their original design to work then they could ^{merge} ~~incorporate~~ products and sales together so if they both are on the same server ~~webpage~~ they could also use the JSOP approach where they edit the `<script>` tag to be able to access the website methods and properties so it doesn't have to be on the same ~~webpage~~ ~~webpage~~ service. x COR

- 3) a)
- Volume - The amount of data that is going to be processed for a big data application. This could be ^{the size} terabytes, for e.g.
 - Veracity - This relates to the uncertainty of the data collected and whether it will all be valid, reliable (or even usable)
 - Velocity - This relates to the speed at which the data is processed. Some situations require real-time processing to make quick decisions.
 - Variety - This relates to the data and whether it is structured, semi-structured or completely unstructured.
 - Value - Also a V to consider. The value of ^{the} ~~that~~ data is ~~going to~~ and what it is going to be used for is important. There is no point collecting data that has no value.

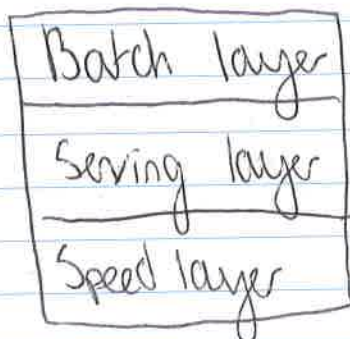
3c.)

The Speed layer can handle errors as it computes queries from recent data and is always going to be updated/recompiled. The batch layer cannot handle errors as it is where all the large data sets are stored and that is what the Serving layer uses to compute batch views; however the Serving layer can handle errors more than the batch layer as its views are automatically updated (with the queried data from the batch layer) (3c. continued below)

3.5

3b.)

Lambda architecture of big data =



The batch layer is where the large datasets are stored. This layer stores the raw immutable data and is ever growing, the datasets are too large to query quickly so subsets of limited data are

compiled and these are called batch views.

The Serving layer is the layer where it displays these batch views so that they can be queried.

This layer is updated automatically by the batch layer and ~~the~~ when it receives newer data.

The Speed layer makes up for the speed of the batch layer as this layer is real-time data processing and is based upon recent data rather than every piece of data stored in the batch layer.

The Speed layer can support random reads and writes.

3d.)

This architecture is known for BASE = This stands for = Basically available - as if one dataset is down, then another datanode is

available to be accessed.

Soft State which is also linked to eventual consistency where the big data application will always find a response to why a server goes down and provides an immediate answer ~~that~~ when that happens.

3b) (continued)

The batch layer is the slowest layer to query, the serving layer is quicker as it is querying limited datasets.

The Speed layer is the quickest to query as it only processes the incoming data.

The data in the batch layer is raw, immutable and ~~is~~ externally true. Data is never overwritten or deleted from here as it is the master dataset. This architecture enables the big data application to be fault tolerant, Scalable, extensible and debuggable.