



**Complete, fold and seal**  
(see note 2 below)

FORENAME(S): Tom

## 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 either 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 answer 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.**

[illegible]

i) a) Web browser process:-

First need the IP of "https://google.com"

- DNS

Notes - DNS

- Build packet

> TCP/IP, Ethernet

- ARP table to store IP and corresponding MAC

First, IP address of "https://google.com" must be found

- browser checks its DNS cache to see if it is there

- if not, asks DNS server, starting with the 'com' domain server

- if not there, checks Net BIOS

1/2

1 1/2

Assuming that the IP is now known, the packet can be built

start with TCP

Protocol set to match URL

Now, PC sends its GET request across

Now, build packet

- add TCP containing the

Next, send request to router and desired port

- router enters the IP and MAC of sending machine into its ARP table

HTTP request is now sent encapsulating TCP/IP packet (it's a GET, so just a header)

Request jumps from router to router, with the packet communicating:

- its origin
- its destination and target port
- the next hop

Once it reaches the target IP the target router first checks its ARP table for the MAC of the destination machine - if it is not there it will make an ARP request to all machines in its ethernet segment asking the machine with the matching IP to provide its MAC address. Once the MAC is obtained the request reaches the target machine. It must now respond.

To send a response and body to the client, it builds a TCP/IP packet, which is then wrapped in HTTP, then encapsulated in Ethernet frame to reach the router. The router will then use its ARP table to work out which port and MAC corresponds to which IP to send the response to.

The response packet travels much like the request does, and will either contain an HTTP response code (hopefully 200 OK) and content namely Google's home page.

Once it reaches the client's network the packet is unwrapped at Ethernet, HTTP and TCP/IP as it returns to the sender.

(This space left blank for additions)

b) URI: Uniform Resource Identifier. Generic term for anything that can be used to locate or identify a resource. The term encapsulates URLs and URNs. 3

URL: <sup>Uniform</sup> Universal Resource Locator <sup>in</sup> (?) (can't remember) 1/2  
Refers to the specific code name (acronym RN) used to reach most sites. It is composed of domain, subdomain, root, top level domain and more.

e.g.  
https:// google.co.uk :80 /index.html  
protocol domain top-level Port file path  
licensed by domain  
Microsoft

URN: Universal Resource Name 1/2  
is an absolute known path to a specific resource. Where a URL takes the client to sections of file structure, URNs always give one resource.

e.g.  
http://isbn/1809715 (This is my student number don't think it leads anywhere)

2)  
a)

HTTP verbs for CRUD

CRUD: Create, read, update, delete

GET: request with typically no body, is a request to retrieve and view info data / resources. Safe and idempotent

POST: creates a new resource with parameters detailed in the request body. Not safe, as it alters the state, and not idempotent defined in (b)

PUT: ~~can~~ requests an update to a resource, or creation of a new one if there is no resource on the target path. Unlike POST, PUT is idempotent but is not safe.

DELETE: removes the target resource. is not safe but is idempotent

HEAD: returns only the response header. Safe and idempotent.

OPTIONS: returns all allowed HTTP verbs. Safe and idempotent  
(space left for additions)

b) idempotence? The property that one iteration gives the same result as many iterations. As an example, consider PUT. Run once, PUT will make updates to a resource. Run again, the updates have already been made and no changes are needed - therefore future PUTs with the same content perform no action. Idempotency is a useful tool in HTTP as it offers a degree of security against accidental flooding of requests. The verbs that are idempotent are listed in 2(a)

d) 200 OK - the operation was successful with body 2 1/2  
 401 Unauthorised - a more specialised form of 403, is a failure to continue usually associated with login issues 2  
 403 Forbidden - more generically, client does not have permissions to perform this operation. Common in Linux ownership problems 2  
 404 Not found - the server could not locate the target resource. The client asked for errors serverside where other 5xx codes don't fit. Done of developers everywhere. 2

d) As explained above (oops) 403 is generic permission problems, where 401 is specifically the user is not allowed to perform an operation usually due to login issues or account not having high enough access 3

3)  
a)

## REST: Representational state transfer

Client - Server is composed of resources  
 - Clients interact with the ~~res~~ resources by acting upon them (usually nouns) with HTTP verbs (see (2))  
 - REST is stateless. Client Two forms of state in REST: Application state and resource state. Clients are sent resource states by the server and clients send the server application state to alter the resources.

- States are presented to the client via representations. These are sets of data that ~~sample~~ show the contents of the state, and the legal operations that can be performed on them. This is an advantage. Also contained in a representation are hyperlinks to visit related states and to run these operations. This is known as Hyperlinks as the Engine of State (HATEOAS).

This is a powerful tool for clients. HATEOAS gives the "levels of state": the set of changes that the client is able to make. This helps avoid authorization issues as clients can only see methods that they can use.

REST's operation is primarily stateless as the client can send all pertinent data for an operation via a representation. This is useful to allow multiple users to connect at once as they all make

The usage of verbs and building of URIs in REST is made to conform to strict standards. This e.g. to access the drivers of car 4:

This is extensible, a large part of why REST was developed. It is also human-readable, a benefit of REST.

7

## Rhif y Cwestiwn

Hibernate must be configured for as Java to be able to read the database tables and convert them into a format it can use.

One method is ORM (Object relational mapping). ORM pulls or disassembles objects into their constituent parts and stores them in a database. Then facilitates rebuilding of those objects later. Hibernate has ORM tools built into it and most IDEs allow for auto generation of POJOs (plain old Java objects) which share the properties of database entities.

(Object-relational impedance mismatch?)  
(leaving space)

7

4)  
a)

### Cloud:

- Resilient. Cloud companies have access to databases all over the world, allowing functionality to be maintained ~~or over~~ ~~for~~ through even natural disasters
- Scalable. For function-as-a-service (FaaS) and Software-as-a-service (SaaS), your code can be made to run only when it is needed, and can be spun up to handle high traffic. This is a more natural fit for ~~cons~~ Consumers - services should expect dead periods and spikes (e.g. amazon at Christmas time)
- Cheap. When your service is not needed it can be spun down to zero instances. This means the developers are not being charged for this downtime. This eliminates maintenance costs
- Maintenance. Spoking of cloud companies are not required to obtain new hardware, licences etc to meet demands of new tech - they also don't need to repair broken servers or handle configuration - this is the ~~respons~~ responsibility of the cloud service provider

### Drawbacks:

- Cold start. If a service has been inactive and a connection is made to their service, a connection must be speeded up to meet the demand. This O-1 time is significant and may cause errors or customer dissatisfaction

- unpredictable billing. There is no way to know how many much traffic a service will receive. This means it is uncertain how much a company will need to pay for their servers. This is a problem for budgeting.
- Security Fundamentally, your application service is being run remotely. There may be private data that cannot exist off-site.

b) Vulnerabilities of web services:

- Sanitise inputs. A user may be able to inject SQL in strange places or other code.
- secure endpoints. Unsecured, any user can perform any operation on any resource. A solid authentication system is required.
- Password Hashing (or similar). If user details can be sent in a request to authenticate them but the request itself is merely encrypted - sniffers can pick up the requests and steal user details. Instead, hash private info before transfer to protect it.
- Traffic / Overload. A service should be able to handle its expected user base all at once, or at least a lot of them. Measures should be taken to protect have access to more processing power when in need (cloud is one way).

- Browser issues. A service should work on most common web browsers. It should not be dependent on Javascript as some users have it disabled.
- Backup systems. If the main server has a physical fault, there should be some way to keep the service alive, possibly with a secondary host.
- Safe storage. Data necessary to the service's operation should also be backed up in case of accidents.

c) JMeter: a useful tool for testing simulating usage of a web service. Requests and actions can be programmed into JMeter to test a path through the service as if a user were performing the task. These programmed paths may then be called simultaneously and repeatedly. This later allows for stress testing. This is also a useful way to catch bugs that require multiple actions to be performed at once.

Swagger is an implementation of Automation Tools are helpful for this property of simulating live usage but also consistent usage. A query can be run repeatedly overnight to check for memory leaks.

S

L