**TM352 notes**

The underpinning technologies of the web are:

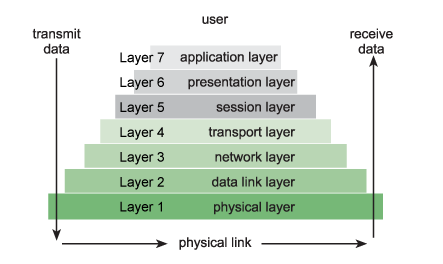
* Protocols
* Open standards
* Architectures

*Protocols*

A protocol is a set of rules that need to be followed for successful connection, communication and data exchange between any 2 points on a network. Protocols define the format, semantics, start and end, how errors are handled and how messages are received when they are sent between client and server.

*The Open Systems Interconnection (OSI) model*

Describes communication functions of a telecommunications or computing system without regard for the underlying technology in the system.



Each layer receives information from the layer above, processes it and passes to the next layer. Information such as headers is also added with each transportation of the data. Some of the protocols associated with the layers are

|  |  |
| --- | --- |
| Ethernet | Associated with layers 1 & 2. Describes how devices can format data to transfer to other devices in the same network. |
| TCP/IP | Transmission Control Protocol and Internet Protocol. TCP manages the assembly of messages or files as packets for transmission over the internet to be received by another TCP layer that then assembles the packets into the original message. IP handles the addressing of the packets so they go to the right destination. Works at levels 3 and 4. |
| Telnet | Enables users to access remote computers. Level 7 |
| HTTP | Hypertext Transfer Protocol is foundation of the web, and is a request-response protocol that functions at level 7. |

Most operating systems span layers 1-4

|  |  |
| --- | --- |
| 7. Application layer | High-level Application Programme Interfaces eg email and browsers. |
| 6. Presentation layer | Translation of data between a networking service and an application. |
| 5. Session layer | Management of communication sessions between any 2 given nodes |
| 4. Transport layer | Transmission of data segments between points on a network |
| 3. Network layer | Structuring and managing a multi-node network including addressing, routing and traffic control |
| 2. Data layer | Reliable transmission of data frames between 2 nodes connected by a physical layer |
| 1. Physical layer | Transmitting and receiving raw bit streams over a physical medium |

***HTTP protocol***

Makes no assumptions about hardware and network configuration, stateless application layer protocol. Request-response protocol.

A successful response contains the contents requested, along with a header containing a 200 OK.

When a web site is requested, a HTTP request header is sent. The first line tells the server what sort of request it is, in this case GET. There is also details about the browser that sent the request and the character set the user agent is expecting. Once the server receives the request it interprets it and then sends a response.

Once the server checks that it can locate the requested file, and that the client has permissions to read the file, then it creates a 200 OK response to send back to the browser along with the requested page. The response includes the content size on bytes which is key for the browser to prepare to display the content. It also contains information about whether to keep the TCP/IP connection open for further messages (keep-alive) or whether to close the connection (close). It also tells the browser the content type (text/html).

*HTTP status codes*

HTTP status codes allow the client to respond appropriately if there has been an error with a request. They are also supplemented with human readable error messages that are returned as line one of the response header, and also can be returned as a more detailed HTML page to display more information, which is returned in the body of the response.

|  |  |  |
| --- | --- | --- |
| 1xx | Informational | Request received, expect a further response |
| 2xx | Success | Request received, understood and acted upon |
| 3xx | Redirection | Client should try again, modifying request as indicated and without user interaction |
| 4xx | Client error | Request cannot be fulfilled due to issue at client end; user should be informed |
| 5xx | Server error | Request cannot be fulfilled due to issue at server end |

Message exchange - HTTP message is passed from application layer down to transport(host-to-host) layer, which uses the TCP. This then passes the message down to the network layer, where IP protocol is used, and then down the other layers to reach the HTTP application.

TCP operates at the host-to-host layer. HTTP operates at the application-to-application layer. TCP creates a guaranteed delivery serviced from one computer to another, also known as an end-to-end guarantee. The amount of packets that the message is broken into depends on the transmission medium. TCP breaks messages into packets, and ensures that received packets are put back together correctly. It acknowledges every packet received and can retransmit missing ones. TCP is a connection-orientated protocol.

The internet layer is responsible for directing message across the internet. This is done using IP addresses. The DNS (Dynamic Name System) servers and protocols are used to look up the human readable name of the computer associated with an IP address. IP is a connectionless protocol.

Each protocol defines a set of functions and these are available for use in the layers above and below it.

HTTP requires the delivery of request and response messages to their correct destinations to be guaranteed. This is achieved with TCP/IP. A handshake is used to initiate a connection.

Packet loss occurs when one or more packets of data are lost when travelling across a network. Attributed to network congestion, poor cabling or faulty hardware. Checks are made at endpoints to determine whether packets have been lost, and whether TCP should retransmit. Packet loss is more troublesome when communicating in real-time, like streaming or skyping.

As HTTP messages travel down the stack, each protocol layer adds information to them. TCP adds sequence numbers so the message can be reconstructed in the correct order. IP layer includes the IP address as a header. Then as a response is received the headers are used to check for errors, and if none are found are stripped and the enclosed data sent to the next layer. This aids encapsulation of the data as data for one layer need not be looked at by another layer. Each layer treats information it receives from other layers as a black box.

The TCP protocol connection consists of 2 socket endpoints. 1 socket host one socket a port. Multiple protocols can send messages through the stack at the same time, and these are messages are kept separate by assigning port numbers to each conversation. TCP port numbers distinguish between different applications running on any one computer (as IP distinguishes computers on a network). The combination of an IP address and port number is represented in software by a socket. Port numbers are reserved for common internet services. Port 80 is HTTP. A firewall inspects TCP/IP packets and rejects any that might be unauthorised.

HTTP methods:

GET – requests a resource. Safe as no side effects  
HEAD – asks for headers but no body that would be returned by identical GET method  
POST – Submits data, updates server.  
PUT – uploads a file or other resource  
DELETE – Deletes a file or other resource.

Basic HTTP protocol isn’t very secure. Secure Socket Layer(SSL) was 1st secure HTTP protocol (1994) Transport Layer Security (TLS) protocol also similar, and provide an encrypted connection. https prefix in web address tells the server whether to open a secure or normal connect. Secured connection usually opened in port 443.

Open standards - a standard is an agreed, repeatable way of doing something, as a published document that contains a technical specification or other precise criteria for its use. Web standards are a set of recommendations that standardize web technology. Open standards are published for public use and available for 3rd parties to read and implement without royalties. Provide interoperability irrespective of OS and hardware. Ensure access to a resource is not dependent on any single application, or particular hardware platforms.

W3C process states that to be considered an open standard a technical specification must have:

* Transparency
* Relevance (analyse market needs)
* Openness (anyone can participate)
* Impartiality and consensus
* Availability
* Maintenance

Recommendations and Requests for Comments (RFCs) – 2 main community driven organisations (W3C(Recommendations) and IETF(Requests for Comments)) publish RFCs.

W3C publishes documents that define web technologies, and provides support for emerging technologies. Recommendations follow process of peer review before they are finalised. Then the Recommendation becomes a standard.

IEFT publishes internet standards, the term RFC is used to describe its documents that are published for discussion which may or may not become internet standards. TCP, IP, SMTP, FTP and Telnet all IETF internet standards.

ECMA International is an industry association that produces standards for ICT and consumer electronics.

Before web standards it was hard to write code for multiple browsers. Must keep up to date as standards are being updated all the time.

Architecture – conceptual structure and functional behaviour of a computer system. Arrangement and interactions between software components.

* Client-server and two-tier – client is requester of services and server is provider of services. Transaction between client and server called request and response. Any number of clients can connect to and request from server. Typical client is a web browser that accesses web applications using HTTP. Thin client is one with very little functionality of its own and thick client is one which is more intelligent.
* Multi-tier architecture – client and server tiers can be divided further. Each tier can be changed more easily as it’s less dependent on details of the other components.

Applications vs services – an application is something that is used to perform a complete task. A web service is something that performs a smaller task than a piece of software is designed to, and is made to be inoperable between different software applications.

Simple Object Access Protocol (SOAP) is based on Extensible Markup Language (XML) which is used by web applications to exchange information. SOAP is a standard which defines the format of the messages. W3C Recommendation. SOAP messages usually exchanged using HTTP but not limited to that transport mechanism.

A web service is a service that a client, browser or application can call remotely across the internet, utilising web protocols and standards to exchange data irrespective of programming language and platform. Interoperability (platform independence) is made possible by the adoption of standards such as XML, HTTP, TCP/IP, HTML etc).

Three different kinds of web service are:

* Read-only – transmits data back
* Write-only – data is transferred to a web service at regular intervals
* Read/write – data is sent to web service and amended data is received back.

In order to communicate client sends a request message and service sends a response message.

Service Orientated Architecture (SOA) – messages are sent between a client and service provider.

A SOAP message consists of 3 parts:

* An envelope – what is in the message, who should deal with it and whether it’s optional or mandatory.
* A set of encoding rules – defines a serialization mechanism that can be used to exchange instances of application-defined datatypes.
* RPC representation - A convention for representing remote procedure calls and responses.

SOAP specifies a structure and encoding for information to be communicated.

A soap message is made up of a single SOAP envelope, consisting of XML nodes for optional SOAP header and mandatory SOAP body which contains the payload of the request. The envelope node is the root element of the SOAP message.

<envelope>  
<header>  
</header>  
<body>  
</body>  
</envelope>

Features can be added to the header as it moves across the network, and the header contains child elements called header blocks.

Class name, method name and parameters are encoded in XML eg

<Convert>

<convert>

<fahr>41.0</fahr>

</convert>

</Convert>

The service then formulates a response in XML

<Convert>

<convertResponse>

<celsius>5.0</celsius>

</convertResponse>

</Convert>

The client needs to know about the conventions of the response so it can interpret it.

A SOAP message for this might look like

<?xml version="1.0" encoding="UTF-8"?>

<soapenv:Envelope

xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope/

xmlns:xsd=http://www.w3.org/2001/XMLSchema

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<soapenv:Body>

<convert xmlns="http://www.open.ac.uk/tm352/convertNameSpace">

<fahr>41.0</fahr>

</convert>

</soapenv:Body>

</soapenv:Envelope>

Any sender or receiver of SOAP messages is a SOAP node. The node that performs the original transmission is called the original sender. The final node that processes and consumes the message is called the ultimate receiver. Any node in between is an intermediary, and these nodes make up what is called the message path. The body of the message is known as the payload.

There are 3 SOAP attributes that determine what happens to header blocks as they pass through intermediary nodes to ultimate receiver. This attributes are used to process the message until it reaches end node.   
role – identifies which node the header block is targeted at.   
relay – indicates whether that node should forward unrecognised optional headers or discard them  
mustUnderstand – determines whether node discards or forwards the Header block. mustUnderstand=”true”. If no attribute or mustUnderstand=”false” the header is optional.

In Java a web service is implemented by operations which are implemented by Java classes and methods which are annotated as @WebService and @WebMethod. Parameters for the WebMethods are annotated with @WebParam.

XML is a common format for data interchange across the web. Standard published by W3C. The tags delimit the characters that make up the parts of the data.

XML focusses on meaning of data rather than presentation, HTML focussed on presentation. XML is self-describing as it can be understood by humans. Even numbers are represented by strings so it’s very portable across platforms as it’s easy to process. XML namespaces ensure that all element and attribute names are unique and unambiguous.

An XML document is made up of elements, which include the start and end tags and everything in between. The content of an element is everything not including the tags. Empty elements can be written as self-closing e.g. <houseName/>

Tag names must start with a letter, not start with the characters xml, cannot contain spaces and are case-sensitive. Text that occurs between tags can contain nearly any character from any language and script other than English. Certain characters need to be encoded, such as the & and < > as &amp;, &lt; and &gt;

XML is hierarchical and elements can have child elements and parent elements. Elements with the same parent are siblings. When represented as a tree view there are root nodes and leaf nodes. Leaf nodes don’t have any further branches. The root node has no parent node.

Without a root node an XML document is not well-formed.

Every XML document should begin with the XML declaration. <?xml version="1.0" ?>   
The declaration must be at the beginning of the first line of the document.

Any software that processes XML must be able to read it, convert it to a stream of characters, and detect and deal consistently with any error it detects. The component of the software that carries this out is the parser. As soon as the first error is detected the parser stops and the error is reported in some way. XML is strict and no errors are tolerated to ensure identical behaviour between different parsers. A well-formed document is defined, delimited using beginning and end tags and is correctly nested.

**REST and JSON**

Web services are a set of technologies designed to facilitate machine-to-machine communication. We have seen that we can describe data in a structured manner using XML, but this is not enough to serve as a stand-alone resource that can be called by other applications. There has to be an interface to this data that knows what inputs it might expect, what functions to perform and what outputs to produce. This interface also has to be able to handle and report errors in a graceful manner and enforce security policies using existing network protocols such as HTTP. This is the role of a web service.

REST – Representational State Transfer  
JSON – Javascript Object Notation

RESTful services use web standards (such as HTTP, URL) to make data available to machines and humans. Does not require any additional protocols over and above existing internet protocols. Uses HTTP and URIs. RESTful web services can be created without need for any additional mechanisms above existing web technologies. REST itself is not a standard or actual technology, it’s just a style of implementing web services.

The aim of REST in terms of delivering web services, is to ensure that RESTful web services (those that use URIs and HTTP) retain all the benefits of the ‘design’ of the web in terms of loosely-coupled, hugely scalable, open-standards-based and vendor-independent technologies without introducing further complexity, constraints or weaknesses.

REST is based on the concept that there is a resource accessible over a network at any time, and each resource is defined by its own unique address.

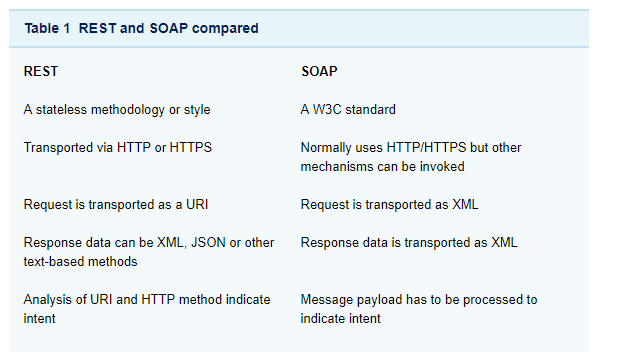
Main concepts in REST are state and transition, and a transport protocol that allows transitions to be navigated. A state is an HTTP response (page and content), a transition is a state that is accessible from the current state (a URI available in the current state) and the protocol is HTTP.

REST services are stateless, each request has to carry all the information required to process the request and no information can be stored on the server.

Each HTTP operation performs an action on the resources making up the web service.   
GET – gets a representation of a resource  
POST – The data in the body of the request modifies the resource  
PUT – The body of the request creates or sets the current state of the resource  
DELETE – resource is deleted from serve  
HEAD – Retrieve information about the state of a resource. Avoids an extra GET if the resource has not been changed.

REST vs SOAP  
SOAP messaging cannot be recognised and understood by firewalls, so cannot be monitored by them. REST operates on HTTP so firewalls can operate on them.

Requests to SOAP resources cannot be logged accurately as the specific service being called is hidden in the message itself. Calls to all REST services can be logged as each service has a specific URI



SOAP has a single endpoint which then exposes a set of operations, whereas RESTful services are resource orientated, with every resource having its own URI.

JSON is more compact than XML, easier to read and write. No need for a parser as can use Javascript functions to convert JSON data into Javascript objects.

JSON is a standard maintained by json.org

Advantages of JSON:  
 Compact  
 Easy for humans and computers to understand  
 Maps easily into data structures used by most programming languages  
 Many languages have libraries for reading and writing JSON

Data representation as name/value pairs, separated by commas. {} hold objects and each name is followed by a :. [] hold arrays and are separated by , .

JSON values can be a number, string, bool, array, object or null. JSON object always surrounded by set of {}

**HTML5**

HTML tags are made up of elements and their attributes.

HTML is important because it delivers the structure of the delivered content

The meta tag in the head contains information about the character set used to parse the document. Simplified doctype over other HTML versions.

New features in HTML include

* Native audio and video support with <video> and <audio>, which means there’s now no reliance on plugins for this functionality
* <canvas> element for drawing 2D graphics.
* Support for SVG
* More descriptive markup <header> <section> <nav> etc
* Cross document messaging
* Support for CSS3
* Support for device detection, which is determined from the header received from a web request
* Support for geo-location
* New input types for forms

**Forms**

Form has a method attribute which would normally either be GET or POST. If not specified the default used is GET, but this means the form data will be appended to the URL. Form also has an action attribute which is the URL of the resource that will handle the request once the form is submitted.

HTML5 allows for email and URL validation on client side without much coding.

A type=”number” input has a minimum and maximum attribute value, and also a step attribute which allows only a stepped value between the min and max values. Browser will display validation messages if rules are broken.

Validation of url type input checks that the input value contains a protocol to allow the value to pass or not.

The legend element can be added within a fieldset to give the fieldset a name.

A web container environment is set up on the web server where the website is hosted, and the container implements the security features such as authentication.

When a user is authenticated, the server creates a **principal** that is assigned to the user, which also includes information about their **role**. (In Java this is implemented using an object of class Principal.)

**Security**

Security protects the system from its users, as well as protecting users from untrusted code.  Security measures should be implemented throughout a variety of application layers for robust protection.

A security policy defines what people can and can’t do with network components and resources.  Attacks on web servers across the internet include trying to gain access to content, files, programs and scripts that they do not have permission to access.  Poor coding standards, improperly configured servers and inadequate use of encryption techniques can lead to successful attacks.

Open Web Application Project (OWASP) is not for profit organisation focused on improving security of software.  Their top 10 of web application security risks are:

         Injection

         Broken authentication and session management

         Cross Site Scripting (XSS)

         Insecure direct object references

         Security misconfiguration

         Sensitive data exposure

         Missing function level access control

         Cross-site request forgery (CSRF)

         Using components with known vulnerabilities

         Un-validated redirects and forwards

A vector is a route by which a computer may be subject to a threat of some kind, such as by email or via portable media storage. The most common route of attack is via the network. Knowledge of the main threats is important to prevention of attack.

Hackers – enjoy the challenge of defeating security systems, with activities including keystroke logging, password harvesting, identity theft or taking over a PC  
Unethical businesses – competitors attempting to gain access to confidential information, click fraud.  
Bulk-mailers – attempt to misuse trust, access information or steal identity with spam messages  
System users and members of the public – are threat through unintended misuse or deliberate effort.

Computer security means (CIA triad)

1. Confidentiality – data confidentiality (ensuring confidential info is not disclosed to unauthorized individuals) and privacy (individuals control what info related to them may be collected and stored, and by whom)
2. Integrity – data integrity (info and programs only changed in an authorized manner) and system integrity (system performs its intended function free from unauthorized manipulation of the system.
3. Availability – system works promptly and service is not denied to authorized users.

Classes of security threat are:

Normal operation – secure distributed or local communications  
Interruption – asset is unusable or unavailable when required. Denial of service.  
Interception – unauthorized party has gained access to an asset.  
Modification – interception with alteration to an asset.   
Fabrication – creation of unauthorized new object in the system

**Encryption**

Encryption is the process of encoding messages in such a way that only authorised parties can read it. Stops message being read should it be intercepted. A cipher is an algorithm by which text is encrypted or decrypted.

The intended communication or message is known as plaintext, and this is encrypted using an encryption algorithm, which generates a coded message called ciphertext. Cryptography is the study of encryption schemes.

Caeser Cipher – simple encryption replacing one alphabet character with another, using a fixed offset along the alphabet. Offset value can be considered the key to the code, the value that allows encryption or decryption. The same algorithm can be deployed again and again with different key values. Easy to decrypt with brute-force approach. The set of possible keys is known as the key space. ROT13 substitution method shifts letters by 13 characters.

To make cipher stronger we need to obscure statistical properties (letter frequency) in the cipher text to make it more difficult to decode.

Block ciphers – obscure statistical properties in ciphertext by enciphering it in units of more than a single character. The plaintext is broken down into blocks, and these blocks are encrypted, rather than individual letters. Most modern encryptions ciphers are block ciphers. The strength of the scheme depends on the block size (bigger blocks are more secure). First widely used block ciphers were Digital Encryption Standard (DES), which was superseded by Advanced Encryption Standard (AES) and triple DES.

Stream ciphers – implement symmetric encryption. They encrypt one bit or byte at are time, and are used when the amount of data is unknown, or continuous. Also known as a state cipher as the encryption of each digit in the plaintext depends on the current state of the cipher. Uses a random unique key so should be immune too brute force attacks, but creating the keys can be computationally taxing due to the length and randomness required to keep them secure. RC4 (also known as ARC4 and ARCFOUR) most widely used stream cipher, and is used in wireless security protocols WEP and WPA.

Symmetric key methods – same key is used to encrypt and decrypt messages. Less secure as the key needs to be transported to the receiver somehow that is different to the way that the message has been sent, and the key exchange must happen in advance so can’t be used casually. Involves trust that key won’t be shared with third party by either side. Achieve good level of security against brute force attacks and require lower level of computation.

Asymmetric – also known as public key encryption. Involve 2 related keys that are produced in pairs, either of which can be used to encrypt or decrypt, and then the other to perform the opposite encryption or decryption. The key pair is generated from extremely large prime numbers. Usually there is a key that is widely known, the public key, and one that is not, and could not be computationally guessed, which is the private key. Decryption can only happen with the private key. Benefit is that there has to be no exchange of secret keys beforehand, and they don’t have to trust anyone else not to divulge the key. Disadvantage is that they are computationally heavy. RSA is an example.

Symmetric and asymmetric methods can be employed together to form a hybrid cryptosystem.

When transporting sensitive data between client/server we must consider:

* Privacy – data is intelligible to rightful recipients
* Authenticity – data received has been created by purported author and has not been manipulated by another party
* Integrity – infeasible for contents of data to be changed in transit without changes being instantly obvious to recipient.

Digital signatures – messaged encrypted with private key and decrypted with public key, to provide proof that the sender is who they say they are. These act as digital signatures as they cannot be forged b anybody else. Replay or playback attack is where someone intercepts a message with a digital signature and impersonates the original sender, manipulating information from the receiver. These can be avoided by checking the ‘freshness’ of messages to check they haven’t been intercepted, or exchanging a nonce value with the message. Man – in – the – middle attacks could also occur, with a user pretending to be the other user to each other user.

A method of ensuring integrity is using the hashing method. A hash calculation is performed on the message, with the result given as a single value which is usually a large number or fixed length string of bits, which is referred to as the message digest. Any change to the value will result in a different hash message. Most common hash functions are MD5 and SHA-1. Hashing techniques prove integrity and authenticity of a message.

For two parties to exchange encrypted data cryptographic keys must be shared and exchanged between parties.

Diffie-Hellman key exchange – asymmetric key system avoids exchanging secret keys over insecure connections. The Diffie-Hellman key exchange allows a secret key to be agreed in a secure manner over an insecure network. A shared secret is generated between two parties, but anyone observing the communication won’t be able to see the secret.

Digital certificates – website owners can purchase a digital certificate, public key certificate, SSL certificate or X509 certificate, which is a digitally signed document that binds together the identity of the individual or organisation, and a public key. Certificates are issued by certificate authorities (CA’s), who verify the identity of applicants before signing and issuing a certificate. Certificates are added to the websites server configuration, and this offers user’s assurance that the communications with the website are legitimate and secured by encryption. Browsers come pre-loaded with digital certificates of many certificate authorities, which are called root certificates. Any certificate signed by one of these authorities is automatically accepted by the browser.

Authentication – identifying a user as being who they claim to be. Can rely on one of three things:   
 Something you know (password, username)  
 Something you are (biometric identification  
 Something you have (token)

Modern web servers support authentication with:

* Basic authentication – based on something you know. Simplest of HTTP authentication methods. Unauthorised requests are rejected by the server and get an unauthorised response. The client then sends log in credentials in an encoded message back to the server, where it’s is decoded, and the credentials checked against a local list of users. Access is granted if they match, or denied if not. Exchange is not secure as credentials are not encrypted, so must be used with a secure protocol.
* Digest authentication- introduced to protect credentials using MD5 hash function. The unauthorised response from the server includes a nonce value that is hashed along with the credentials by the client. When the server then decrypts this message and then nonce is the same as that which it sent originally, then access is granted, as l long as the credentials are correct.
* Client certificates – certificates can be used to authenticate an individual by binding a public key to their email address or network credentials.

Authorisation – do you have permission to do what you’re trying to do?

Secure data transmission is transferring sensitive and confidential information over a secure channel, entailing some type of encryption and some form of key exchange.

Secure sockets layer (SSL) – https means HTTP is being layered on top of SSL. Uses TCP to provide authentication and privacy through encryption. A handshake between parties enables negotiations regarding communication. Client and server must be able to communicate using an encryption algorithm that they both understand. Request starting https causes the browser to request a secure encrypted channel, usually on port 443.

**Block 2**

**The Cloud**

Cloud computing – storing, processing and use of data on remotely located computers accessed over the internet. Computing power available everywhere to anyone.

AWS – Amazon Web Services – provides networking, storage and network infrastructure on demand as the need arises. Computing power to host web servers and database servers can be bought as virtual CPUs per hour, storage for data bought as gigabytes per month; and networking bought as gigabytes transmitted and received. Pay for what you use.

Mell and Grance ([2011, p. 2](https://learn2.open.ac.uk/mod/oucontent/view.php?id=1151364&section=__references)) defined cloud computing as:

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models.

**NIST cloud definition** -

Characteristics of the cloud –

* On demand self-service – consumer (owner of hosted application) can create infrastructure to host an application (provision) such as server time and network storage as it’s needed without requiring human interaction. Dynamic assignment of IP addresses and DNS entries for hosts.
* Broad network access – capabilities are available over the network and accessed through standard mechanisms (protocols and standards)
* Resource pooling – the providers computing resources are pooled to serve multiple consumers, with different physical and virtual resources dynamically assigned and reassigned as needed. Location independence.
* Rapid elasticity – able to scale rapidly outwardly and inwardly to match demand. Increase resources as demand increases and dispose of them as it falls.
* Measured service – control and optimize resources by measuring and monitoring usage.

Service models –

* SaaS – software as a service – entire app and hosting infrastructure is responsibility of service provider. A single service is shared by multiple tenants. Eg dropbox a Netflix. Subscription services.
* PaaS – Platform as a service – consumer supplied with the capability to deploy onto the cloud infrastructure. Consumer does not manage the underlying cloud infrastructure, but has control over the application and configuration of settings. Eg AWS and Azure
* IaaS - Infrastructure as a Service – tenant takes responsibility for most of the application, but not computer or networking infrastructure. Renting a dedicated host to run a web server or data server. Choice of OS. Eg Rackspace, Azure.

Deployment models

* Private cloud – cloud infrastructure provisioned for use by single organisation comprising multiple consumers (business units)
* Public cloud – cloud infrastructure is provisioned for open use by the general public.
* Hybrid cloud – composition of 2 or more cloud infrastructures.
* Community cloud – cloud infrastructure provisioned for use by specific community of consumers. Access restricted to members of the community.

Different cloud models: NIST, UCSB+IBM and Hoff.

How to cloud arose:

* Virtualisation – due to increased processing power year on year (Moores Law), a typical single-application server only runs at 10-15% of it’s potential. To solve this a single server can be separated using software into multiple applications (virtualisation)
  + Virtual machines – takes resources of a single physical host and shares them between multiple guest computers. Each guest appears to be an independent, self-contained computer. Virtualisation software manages and protects the virtual servers. Each VM requires it's own OS, a guest OS on the host machine. VMs need to be booted up.
  + Containers – much smaller than a virtual machine as does not have its own OS. Essentially a file containing application and associated libraries. Managed by software that load and unload them. Faster input/output operations as they have direct access to the machines drivers. Docker is a popular application containerisation tool. No need to boot up, so quicker to get started.

|  |  |  |
| --- | --- | --- |
|  | VM | Container |
| Size | Whole OS per VM means they are larger in size | Much smaller, could fit 100's on a machine. |
| Start-up time | Large as whole OS needs to be booted up, which can take many minutes. | Seconds as do not need to be booted up |
| Admin costs | High as they need a lot of configuration |  |
| Licencing costs | Whatever the VM management software costs | Whatever the container management software costs |
| Security |  |  |

* Scaling –
  + scaling-out providing more processing power at times when it is needed (when requests are higher). Workload shared out by a load balancer. Increases the availability of the website. Requests need to be stateless so they can be fulfilled by any server. Creating more VMs to hold additional webservers to handle load
  + Scaling-up is where additional resources are added to a computer to increase its performance, by redefining the allocated processors and memory for a VM.

Elasticity of the cloud achieved with virtualisation and scaling. A new VM is launched when demand grows and removed when it falls. Images of VMS are stored, and machines created from it as and when they are needed.

A flavor is a guest configuration offered by cloud providers. These flavors allow single hosts to be maximised. an XLarge flavor is allocated half the virtual resources. Can also have Large, Medium and Small, what are allocated 1 sixteenth of the resources. When a new VM is launched the scheduler simply looks for a host with enough resources for the given flavor.

Difference between traditional deployment and cloud deployment is most of the infrastructure for the web app is created dynamically as and when requested.

IP addresses are spit into a network Id and a host Id. Using the first 3 bits of the IP address, a router can determine what class the IP address is (A, B or C) and therefore how many bits of the address are the network Id.

Traditional Infrastructure:

* Network Time Service (NTP) – global NTP server sits outside the application. Synchronises the clocks of all the servers in the application using network time protocol (NTP). Local NTP derives clock data from global NTP, and syncs all other servers in the application. If times are synced as if file timestamps differ by more than a few secs content will unnecessarily be sent again.
* Domain Name Service (DNS) – global DNS server enables web browsers to locate websites, by converting web addresses to IP addresses. When new domain names are created an entry is created in the appropriate DNS server. Local DNS server handles queries for IP addresses across the local network. DNS cache holds results of successful queries, and is queried before a new DNS query is started.
* Router/NAT – router converts low level signals and packets of one network type into signals and packets of another network type. It also interconnects different networks. A routeing table is used by the router to make decisions about routes to use between multiple LANs. Network address translation allows internet traffic to be exchanged with local devices.
* Firewall/DMZ – blocks unwanted packets of data travelling between internet and LAN. Only packets for new and existing connections pass through. Also check contents of messages and check they conform to the associated protocol. DMZ (demilitarized zone) provides additional protection to more complex networks, using multiple firewalls. The DMZ is the safe point between 2 firewalls.
* Load balancer
* Servers
* Database

Functionality of some components can be provided by software, rather than all individual pieces of hardware.

**OpenStack –** open source IaaS project, can be downloaded and run on own machines. Cloud utilises same protocols and standards as traditional network (see above list)

DNS responsible for handling new records for websites as they’re created dynamically. Cloud is made up of hardware elements Controller, Compute and Network, and Images and Storage which are parts of disk space.

* Controller node – manages the cloud, schedules new VM’s to start and stop, manages VM images, handles authentication for cloud admins and tenants. All done with services. Decides where a VM will be created based on the flavor chosen and available resources on compute nodes.
* Network – manages network elements used by VM’s to communicate with the internet, such as assigning IP addresses to VM’s and routing traffic. Services running here mimic internetwork and data link layers of OSI stack. Also creates equivalent of firewall for VM. Can also provide load balancing functionality.
* Compute – host computer for the guest VM’s. Controller messages selected compute node which loads VM image from image storage.
* Images – Disk space used for VM images, managed by service running on controller.
* Storage – Disk storage used for persistent storage needed by the VMs. Also controlled by service on controller node.

Tenants can either control applications with web-based interface or by utilising RESTful apis.

To begin setting up a website on OpenStack a web server image and database server image will be created and saved to images node. When launched the web server VM will be connected to the internet, and a private network established between the web server and database server. Controller monitors the web server and if necessary launches another instance to spread the load. At this point a load balancer service would be launched to manage requests to the different servers. For more complex/commercial sites there would be multiple nodes of each kind so there are no single points of failure, which are load balanced in the traditional way.

OpenStack services:

* Neutron – provides connectivity for VM instances to internet.
* Nova – compute service to manage VM instances.
* Glance – manages storage of VM images and disk images. Stores images in Swift.
* Cinder – block storage for compute. For end users to store data in. Associated with a VM via Nova. Storage is virtualised – attached to VM, used, then detached.
* Swift – object storage
* Keystone – identity service
* Horizon – web based portal to interact with underlying OpenStack services.
* Ceilometer – metrics collection for billing and scaling

Services can be accessed via:

* Horizon (OpenStack dashboard)
* CLI
* Python-based APIs
* YAML

Software and services can be added to a VM in 2 different ways, dynamically by running as script as the instance is created, or as part of the image itself and saved as an enhanced image.

In this part you have seen how a small traditional web application is mapped to a cloud environment. What this mapping has shown is that every element in the traditional solution has a counterpart in a cloud solution. The crucial difference is that most of the hardware has been replaced by software equivalents. Servers become virtual machines, firewalls become security groups that control access to ports and users, routers are replaced by routing tables that can be reconfigured as required, and disk storage becomes a software volume delivered by a service.

Orchestration means to create a set of cloud component (networks, VMs, storage) automatically, in one action. Specification is created that creates the components as a sequence.

OpenStack orchestration project is called HEAT, and has created a standard way of orchestrating resources using templates (HOTs) which are written in YAML. The basic layout of a template is

heat\_template\_version: 2015-04-30

description:

# a description of the template – human readable description

parameter\_groups:

# a declaration of input parameter groups and order – allows parameters to be grouped and ordered. Alternatively you can just list parame

parameters:

# declaration of input parameters

resources:

# declaration of template resources

outputs:

# declaration of output parameters

Lines starting with # are comments.

Can be used in conjunction with an environment file that specifies parameters.

A stack is an orchestration YAML file.

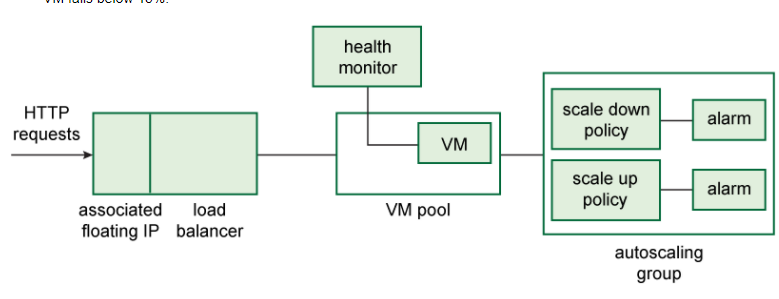
When a web server comes under increasing load, its response time will degrade. More resources would service the demand and maintain quality for users. One approach is horizontal scaling, where another machine is added to share the load. Most organisations would be reluctant to have standby resources just in case, cloud services provide a pay for user model for these circumstances.

A load balancer can be added to the architecture of a network to aid when there are a high number of requests coming in. In OpenStack you can add a load balancer to dynamically add VMs when load increases. Takes the place of the server in receiving requests then sends on to one of the multiple servers that it’s connected to.

OpenStack instances can be monitored by a ‘health monitor’ that will monitor for server failure. If it finds a VM instance isn’t running, then it will remove it. You can define parameters when orchestrating a health monitor, like how often it should try to connect, how many retries, and how long before a time out. VMs are monitored within a pool which can contain it and other VM instances, and the load balancer is associated with this pool.

Load balancers can use the round robin approach, simple sending requests to each server in turn. Another more efficient method is to monitor the load of each server and select the server that is least loaded.

In the orchestration scripts pools are defined and load balancer method defined, load balancer associated with the pool, and the health monitor defined and associated with the pool. Also need to define and alarm and the scale up policy. The alarm monitors the resources, and implements the scale up or scale down policy when a set of criteria are met.



Amazon Web Services (AWS) – proprietary cloud solution (not open-source). Can’t be based in-house and has to be hosted on AWS machines. Resources for AWS are stored in 10 locations globally (regions), which are all isolated from each other to maximise fault tolerance and stability.

An AWS solution begins with a virtual privet cloud (VPC). This exists with AWS public cloud, but is logically separated from VPC’s and is private to the tenant. All VPC’s are allocated the IP range 10.0.0.0/16

Amazon Route-53 is AWS equivalent of a DNS service. Handles registration of domains and directing internet traffic to websites.

Amazon S3 is a storage component that allows assets (files or images) to be stored and then retrieved by a specific URL. AWS automatically maintains a copy of the database in a different availability zone.

Identity and Access Management (IAM) maintains user accounts to access the VPC.

Elastic load balancing (ELB) distributed requests within and across availability zones. Receives internet traffic, then distributes. Monitors load and disposes underutilised resources, or creates new servers as required.

Elastic Compute Cloud (EC2) instances are the equivalent of VMs in OpenStack. Servers are hosted on an EC2 instance. An instance is created within an availability zone and connected to the ELB service. Instances are created from Amazon Machine Images.

Amazon’s Relational Database Service (RDS) provides the database functionality, and can run several popular database engines,

The Amazon CloudFront enables cached content to be served from the edge of the cloud rather than from deep from the VPCs. When a request is made for content from the S3 storage it is moved out to the edge to reduce response times. S3 service is part of the VPC as it is managed by the tenant.

**Benefits of the cloud –**

* Elasticity – ability to automatically expand resources as demand increases, and decrease resources as demand wanes. Running instances are monitored, and if one should fail to respond or exceed performance limits, then a new instance of an image is launched. OpenStack and AWS also allow scheduled scaling to meet expected demand. Reduced costs. Typical aims of cloud providers, in relation to elasticity are:
  + Ability to provision a new VM in less than 5 mins
  + Resize a VM without stopping execution
  + Provide autoscaling based on schedule or monitoring trigger
  + provide sufficient capacity to enable a customer to exceed their normal baseline requirement by a factor of ten in real time and without prior notice.
* Resilience – measure of availability of a VM instance. Reliability of the physical host of the VMs and the ability to move the instances to another host should there be an issue, with minimal loss of service. Critical capabilities are:
  + Detection of physical host failure and automatic restart of VMs on another host.
  + Reduced maintenance downtime by live migration of VM
  + Automated replication across data centres.
* Operation vs Capital expenditure
  + Operational expenditure is expenses of running a business (rent, wages etc.)
  + Capital expenditure is costs for assets for the business (machinery, vehicles, computing equipment) Form part of the value of the business.

It’s easier to buy resources as operational expenditure generally, as it is usually subject to less constraints. They are also able to conserve capital, and don’t need to procure it upfront for projects.

* Costs - would depend on the chosen route of technology.
  + Savings on infrastructure costs, electricity, data centres, equipment and equipment backups.
  + Move from physical to virtual increases server utilisation and reduces costs.
  + Staff costs – increases productivity, deployment code need not be written, and same application behaviour can simply be configured as part of deployment. VM images can be reused and configuration of physical servers need not be repeated.
  + Various providers provide tools to compare the costs of different options, so depending on current hosting set up cost savings would be different.
  + Intangible costs – cost of losing customers if service provided is unreliable

**Risks of the cloud**

Applications and data are hosted on shared resources with a lot less control available to the tenant. VM’s are hosted alongside VM’s of other tenants.

* Data on the move – Using secure protocols for requests to the cloud. AWS also allows for credentials to be passed in the REST requests. Firewalls protect instances, with AWS starting with deny all, with tenant having to open ports required. Data in S3 data store is encrypted, with tenants having their own keys, and managing encryption/decryption.
* Data at rest – encryption of data on disks.

**Privacy**

Data protection principles

1. Personal data shall be processed fairly and lawfully and, in particular, shall not be processed unless –
   * a.at least one of the conditions in Schedule 2 is met, and
   * b.in the case of sensitive personal data, at least one of the conditions in Schedule 3 is also met.
2. Personal data shall be obtained only for one or more specified and lawful purposes, and shall not be further processed in any manner incompatible with that purpose or those purposes.
3. Personal data shall be adequate, relevant and not excessive in relation to the purpose or purposes for which they are processed.
4. Personal data shall be accurate and, where necessary, kept up to date.
5. Personal data processed for any purpose or purposes shall not be kept for longer than is necessary for that purpose or those purposes.
6. Personal data shall be processed in accordance with the rights of data subjects under this Act.
7. Appropriate technical and organisational measures shall be taken against unauthorised or unlawful processing of personal data and against accidental loss or destruction of, or damage to, personal data.
8. Personal data shall not be transferred to a country or territory outside the European Economic Area unless that country or territory ensures an adequate level of protection for the rights and freedoms of data subjects in relation to the processing of personal data.

Personal data –

* Can a living individual be identified from the data, or from the data and other information that you hold, or is likely to come in to your possession?
* Does the data relate to the identifiable living individual?
* Is the data obviously about an individual?
* Is the data linked to an individual so that it provides particular information about that individual?
* Is the data used, or is to be used, to inform or influence actions or decisions affecting the identifiable individual?
* Does the data have any biographical significance in relation to the individual?
* Does the data focus on the individual, rather than some object, transaction or event?
* Does the data impact, or have the potential to impact on the individual in a personal, family, business or professional manner?

Data controller vs data processor – responsibility if there is a breach.

Data controller – carries data protection responsibility and must exercise control over processing. They determine the purpose for which and manner in which personal data are or are to be processed.

Data processor - any person other than an employee of the data controller who processes the data on behalf of the data controller. Processing means obtaining, recording or storing, or carrying out operations on the data

Data processor concerned with technical aspects of operation, such as storage, retrieval and erasure. Activities such as interpretation and decision making in relation to data is job of the controller. Need to thoroughly research whichever solution is chosen and ensure boundaries are clear.

**Trans-border data flows**

DPA says that data should not be transferred out of the European Economic Area unless that country ensures an adequate level of protection for the rights and freedoms of data subjects in relation to the processing of personal data.

**The Mobile Market**

Consumer market is made up of people who purchase goods and services for personal use.

Early adopter vs. Laggard.

Rogers (1962) 5 types of adopters of technological innovations

1. Innovators – 2.5% of total population. First to adopt.
2. Early adopters – 13.5% of population.

🡨Chasm

1. Early majority – 34% of population. Followers rather than leaders, cautious.
2. Late majority – 34% of population, even more skeptical and cautious than early majority
3. Laggard – 16% of population. Last to adopt

Success of an app means the crossing of the chasm to early adopters to the majority. Will never get 100% adoption.

Before iPhone, phones used markup WML(Wireless markup language) which were used for the mobile web that pre-iPhone smart phones used. Text only so mainly checking emails, etc. Desktop web (HTML and HTTP) and mobile web (WML and WAP) were two separate things.

Non-physical factors such as benefits, features and services operate in conjunction with the tangible product to make up the ‘augmented product’.

Post-PC era – use of mobile phones has overtaken use of PC’s.

Consumerisation – innovation for consumers being a driver for business innovation. Consumer products influencing business products and practices.

Bring Your Own Device – empowering employees to utilize their own devices, rather than using company approved devices. Increased agility and productivity, concerns with control and security.

Mobile Device Management – managing devices, users, data and apps.

* Educating users about the potential risks of using personal devices. Policies and training.
* Security policies that control apps with potential to capture and share sensitive information.
* eXtensible Access Control Markup Language

'Invisible computer' - IoT, dedicated devices for certain tasks. E.g. satnav, kindle

Mobile computing:

* Hardware -
  + Screen
    - Size – scrolling allows way to display content that won't fit on the screen.
    - Resolution – measured in pixels along the horizontal and vertical. High resolution means more detail, but require more processing power and memory.
    - Pixel density – expressed in pixels per inch (PPI) combo of screen width in inches and resolution in pixels. To calculate the PPI first find the diagonal width in pixels of the screen (Pythagoras theorem) Then divide this value by the screen size in inches.
    - Device pixel ratio – higher the pixel density, smaller text and images appear. Browsers use property called device pixel ratio to map physical pixels on a device to device-independent pixels to optimize display. The closer you are to the screen, the more resolution and pixel density are important.
  + Central processing units – processors used in mobiles have fewer transistors as the complexity of each instruction is lower, meaning they consume less power.
  + Memory & storage – as each instruction is simpler, more are needed to achieve the same result, so more memory is required. Mobile apps load quickly compared to desktop apps as they’re saved in memory, rather than on hard disk.
  + Energy consumption – dynamic frequency scaling is used to reduce battery consumption.
  + Geolocation – GPS receiver within the phone allows location awareness. If no GPS signal is available the phone can use WiFi signals along with SSID stations to pinpoint the device.
  + Accelerometer – can detect orientation of the user, as well as speed and direction of movement
* Network connectivity – many ways that devices can interact with other devices and their surroundings. Connection to internet via WiFi or across service providers mobile network. Bluetooth between two paired devices over short distances. NFC (near-field communication) communication between 2 touching, or very close, devices.
* Software
  + OS – majority run either iOS or Android. Windows 10 mobile also available
  + Apps – allow user to interact with their device.
* Web browsers
  + Chrome dev tools for mobile –
    - sites detect whether request is from a mobile device using the user agent string that is sent in the User-Agent-HTTP header. Chrome tools simulate mobile agents by replacing this header value.
    - Network throttling – can be used to simulate 1G, 2G, 3G or 4G connections.
* Mobile emulators – a desktop application that can emulate a mobile browser. Cannot reproduce the effect of viewing distance. Put a lot of strain on local client machine resources.

Mobile website design strategies

* Do nothing – do nothing in regards to mobile-specific display considerations. May be able to deduce that most users will be using desktop to view. Also might be too costly.
  + Fixed width design – designers wanted to control how their designs looked, and so created fixed width websites to fit the most commonly used screen size, which was 800 x 600 px. Mid-2000s this increased to 960px width. Advantage of this approach is that the website will always look the same, irrespective of screen size. Default behavior on a mobile is to show the whole scaled down version of the full web page.

Viewport is the part of the browser window where the website contents are displayed, and doesn’t include scrollbar, browser frame or interface components. Viewport meta tags are not supposed to be used on fixed width sites, or any XHTML site, as it’s part of the HTML spec.

* Separate mobile site - requires a way to detect what device is accessing the site, then redirect to appropriate site. Browser sniffing. The browser is identified from user agent string, then client or server side script redirects to correct site. Typically each site still employs a fixed width design, but for each device size.
* Design one site for all devices

Responsive web design – should provide best user experience regardless of device, screen size or resolution. Adjust content according to the width of the browser. 3 principles of responsive design:

* Fluid design – opposite of fixed width design. Allow pag to flow to fit available viewport. % instead of fixed pixel widths.
* Flexible images - no fixed widths
* Media queries – width breakpoints for changes in CSS. Introduced in CSS3.

**Web apps** – application that runs inside a browser

Advantageous as means users don’t have to install software, and only need visit a URL. Users always have the latest version of the software. Web pages with a purpose eg Gmail, maps.

Users must be online, meaning the app is unavailable to the user if there are connectivity issues. They can also offer poor user experience compared to locally installed software, due to limited capabilities of HTML and CSS. They can also be slower, with load time being slow after each interaction with the server.

Some of these weaknesses were solved by AJAX, as data could be sent and received without page reloads. SPA’s.

Appcache feature introduced in HTML5 means application can be accessed offline.

Web app use on mobile device is not as popular as on desktop. Native apps are easy to install, are low cost and are designed for smaller display, making them more appealing.

For web app to appear as native app then shortcut needs to be created on home screen of mobile device. Web apps run in this way launch full screen, so it’s not apparent they’re running in a browser. Metadata in AndroidManifext.xml file in phones source code used to configure web apps for this use.

**Mobile apps**

Either native or web-based. Native are built for and deployed to particular platform and operating system, and are downloaded from app store. Web apps run inside the browser and are created with HTML, CSS and Javascript, cross platform

* Native app –
  + Quality control through ratings and reviews
  + Work efficiently with target OS
  + More time consuming to develop.
  + Integrate tightly with OS and can make us of devices full capabilities

Native apps –

Android apps are generally less expensive, challenging and time consuming to develop than for iOS.

Small fee to sign up for developer account on Google Play, range of IDE’s that can be used.  Can use a number of languages, but mostly Java.  Quick to publish via Google Play Developer Console.  Generally, they’ll appear in the Play store within a couple of hours.

iOS apps are more expensive to develop, involving a monthly fee to Apple Developer Program.  More limited IDE’s, with XCode provided by Apple only capable of running on a Mac.

Developed using Swift programming language.  Apps are reviewed by Apple, based on a lengthy list of guidelines.  Can take between 10 and 14 days to get approved.

Windows app development is comparatively simpler, with a one off small fee to get a Developer account.  Developed in Visual Studio.

Web apps – will work on most mobile operating systems, but work less efficiently than native apps as there need to be components in the middle to translate code instructions to native ones.  Generally built using HTMLS5, CSS3, JavaScript, and run in the mobile devices browser.  Slower to load and less responsive than native apps.  Security implications as view source is available to the user.  HTML5 has number of features meant to facilitate web apps, although still more limited than native applications.  Web apps can access hardware facilities, but it isn’t that straightforward.

Hybrid apps are written in HTML, CSS and Javascript , which is then converted to native code using some kind of tool e.g Cordova.

Requirements – Users needs of a mobile app are much different to those of a desktop app. The needs for a mobile app are typically

* Monitoring the context of use through sensors (accelerometer, GPS, camera)
* Specialising in a single domain and in a unique functionality, so they stand out from other apps.
* Collaborating with specialised apps or online services for less unique functionalities.

A feature gap is a feature of an app that you have identified other apps don’t offer.

Non-functional requirements are the differences in quality attributes, such as usability, customisability, responsiveness etc.

MVC – Model View Controller.

Model – underlying representation or data of program

View – the presentation of the view to the users

Controller – where business logic is carried out