Basic Constructs

Check Digit
Redundant digit or letter calculated from digits of code number, then added to code number, ensure accuracy of other digits in code to be checked

1. only ensures data conform to rules or format, cannot ensure accuracy of content data

Stored as string:

- 10 which needs to be replaced with X.
- Leading zero would be discarded.

Recursive
A subroutine is defined in terms of itself and is able to call itself with a smaller or simpler version of the original problem one or more times in its body and terminates when stopping condition reached

1. Function contains a call to itself
2. At least 1 **terminal case** where no further calls are made so it will not continue indefinitely
3. Original task can be reduced to **simpler version of itself**

Nested function
A function defined inside another function.

Iterative
A performing sequence of instructions repeatedly with loops and repetition continues until condition fulfilled

n Too large / No terminating condition
n is too large and results in crash and stack memory overflow (exceeds memory buffer capacity)

No terminating condition
Index out of range error which results in a runtime error.
Eg. Array X of size 4 → X [5]

Modular programming
Break down a program into separate sub-programs.

Recursive

Advantage	**Disadvantage**
Complex problems written with less code	Missing base case leads to infinite loops

Recursively defined problems **easily solved**	Execution speed **slower**, more memory used
Sequence generation is easier in recursion than iteration	Additional **memory** space for local variable

Recursive	**Iterative**
Calls itself to repeat instructions, more recursive, more memory	Uses loops to repeat instructions
Variable used within recursive subprogram stored in stack frame when calls itself	Variables have values overwritten when loops restarts
Memory to be allocated for new set of variable, more memory, slower	No additional memory when loop repeats
Solves by redefining problem as smaller sub-problems	Solves by executing sequence of instructions finite no. of times.
	constant space requirement regardless of how long the list is.

Global	**Local**
Enables **scope** to be accessed anywhere, anytime during execution	Scope limited to subroutine where declared
Declared **outside** subroutines at main program	Declared **within** subroutine

---

Linked List, Array and Hash Tables

Array

A fixed-size collection of elements, of the same data type, stored at contiguous memory locations.

(Binary search requires a sorted array. If the array is sorted, binary search is more appropriate as it has a time efficiency of O(log n), compared to linear search O(n).)

Describe operations in call stack / Why recursion uses stack

1. The call stack manages recursive function calls by using a stack frame for each call. When a function is invoked, a stack frame is created and pushed onto the top of the stack for each call They hold the function's local state, and returns to the correct point after the recursive call finishes.

2. The stack operates on a Last In, First Out (LIFO) to manage and track the sequence of active function calls, local variables, and return addresses during recursive execution, so the **most recent function call is at the top**. When a function completes, its **stack frame is popped from the stack, memory is deallocated, and control is returned to the previous function**, **which then becomes the new top of the stack.**

3. This process continues until there are no more stack frames, the recursion terminates.

How to remove the top item of the stack (and replace with x)
Stack will check if the stack is empty by checking if the stack pointer is 0.
If empty, display output to the user that it is empty and terminate recursion.
If not empty, assign the item to variable x, and decrement the stack pointer by 1.

Linear and Circular Queue
A queue is a First in First Out (FIFO) data structure.
Enqueue: Items are inserted at the end of queue
Dequeue: the first item (head) in the queue will be removed first
Front Pointer: denotes start of queue, **Rear pointer** denotes end of queue
For Circular queue, Data is arranged circularly with front and rear ends are connected to each other. Eg. Song Playlist.

Linear Queue	Circular Queue
The end pointer of a linear queue stops incrementing when it reaches the end index of the array.	The end pointer of a circular queue wraps around to the base index of the array after it reaches the end index of the array.
When dequeued, the front positions become empty and cannot be reused until the entire queue is empty and reset, leads to potential wastage of space.	Circular queues allows the front and rear to wrap around which reuses of the space without the need to reset the entire queue so it is **more memory-efficient.**
Enqueue and dequeue operations can only be done at the rear and front ends respectively.	Enqueue and dequeue operations can be done at any position.
Linear queue is time inefficient.	Circular queue is time efficient.
Linear queue is space inefficient.	Circular queue is space efficient.

Linked list

A linear collection of data elements where their order is the physical placement in the memory but makes use of pointers to connect one element to another, joining all nodes to form a sequence does not always reflect how the data is stored.

Dynamic Data Structure Eg. Linked List	**Static Data Structure Eg. Array**
Faster insertion and deletion of element than array, only need to update pointers.	**Slow** and inefficient as it requires effortful copying of elements, a linked list update pointers only.
No memory wastage as it is dynamic in size which increases or decreases at runtime by **allocating and deallocating memory**	Memory wastage as it is not dynamic in size so need to give initial size which results in memory wastage
No limit to the capacity.	Less use of memory as elements are stored in a sequential manner in memory locations so no extra memory is allocated and prevent memory leaks.
Inefficient use of memory as extra bytes are needed to store the reference to next node, results in waste of memory.	Fast and efficient data retrieval O(1) as data are stored in contiguous memory blocks **which allows direct access** to the values stored in the array.
Slow and Inefficient data retrieval as LL **does not allow direct random access** and traverses from node at head of linked list, until it reaches that element.	It avoids memory fragmentation as memory blocks are contiguous and aligned.
May result in **memory leak** if memory allocated does not get deallocated and released back to the memory pool. (need to release memory so it is deallocated)	
More time consuming as allocating and deallocating memory at runtime.	

Elevator

This would not result in any run-time error as they may not be able to detect logic errors.

BST vs Hash Table and Linked List

Advantage	**Disadvantage**
Efficient as BST traversal has O(log n) time complexity for all cases but not for LL/HashTable. (BST)	If it is unbalanced and totally skewed left or right due to node insertions, it will become a linked list

	and end up becoming a **linear search** O(n). (BST)
It can get list of sorted items by doing **in-order traversal**, but not for LL/HashTable. (BST)	Hash table has constant search/insert/delete time O(1), while BST has O(lgn), so hash table search is faster
Memory efficient as it does not require more memory than necessary but hash tables/LL require a lot more memory than required to prevent collisions/store pointer. (BST)	Collisions might occur in hash tables that increases the search time, and it could be O(n) in the worst case.

BST

1. Implemented using **four 1-D arrays** [Data, key, Left, Right]. The position index of first element of the array is 0 and the null value is -1.

2. Places first data item in root node. **For every node in BST**, starting with the root, compare the current node with search/insertion key.

3. If the value is less than value of current node, traverse nodes in the left subtree of current. If the value is more than value of current node, traverse nodes in the right subtree of current.

4. Repeat steps until correct position is found when the value of search key is equal to the value of current node, and return True.
   OR current node has reached a leaf node then return False.
   OR current node has reached a leaf node then **add data** as a child node at the empty branch until all data items are added into the BST for insertion.

5. The shape of the binary search tree may differ when the order of how the data items are added are changed.

How to transform unbalanced to balanced BST

Traverse the unbalanced BST in inorder, store each element into an array.
Select the middle element of the array and make it the root node and arrange all elements preceding the middle element as the left subtree, and those succeeding it as the right subtree
Repeat this recursively for each subtree, arranging smaller subtrees to left of root and bigger on right, until each subtree is either null or consists of a single leaf node.

Postfix more efficient than infix

In postfix notation, the order of operations is performed by the position of the operators and operands (left to right). No need for parentheses to indicate grouping, which simplifies both the expression and the evaluation process.

Hash table O(1) has constant search/insert/delete time O(1) faster than BST O(lg n)

Hash table
A data structure that stores the keys to their associated values.

1. Always produce the same hash value for the same key
2. Provide uniform distribution of hash values (every value has an equal probability)
3. Minimise clustering when different keys produce the same hash value. When two or more different keys produce the same hash value, a collision has occurred
4. Efficient to compute.

Load factor
A higher load factor leads to an increase in collisions and clustering, which in turn increases the time complexity of operations.

Collision
A collision happens when a record is hashed by the hashing algorithm to a memory location that is already full.

Deletion without reorganising memory block
The data of the deleted record will be marked by a tombstone value to indicate that it has been deleted. The record deleted is intentionally left in position within the memory location to allow future search to be performed.

Collision resolution

Open addressing (linear probing): Keys with the same hash value are stored in **alternative** locations in the hash table.	If a key creates a hash value that references a position that is already occupied, then the **data will be referenced to the next free position**, by probing sequentially **until an empty slot is found**. If the key is not found at the index position, it does not mean that it is not in the hash table.
Chaining: Keys with the same hash value are stored in alternative locations in the hash table.	In chaining, records are stored in a node of one of the many singly-linked lists. Each array item of the hash table will store the address of one singly-linked list. Whenever a key of a record gets hashed to a position that is already occupied, **the new record will be added to the end of the list that its hashed value** points to.

Advantage of Hash Table Search Over

Hash Table Search/Delete etc	The search will use the hash function to obtain the hash value and go directly to the location to look for the item O(1). hash table only need to get hash address to directly address
Linear Search	Linear search requires the search to loop all over from the beginning of the data set O(n).
Binary Search	Repeatedly compare data and if not found, compare by requires reading data from memory and **calculating** the next index to check, time consuming. O(log2N) pre-condition that requires the data set to be sorted in some order, sorting the records of a media company that possesses a huge quantity of data may be too costly.

Programming constructs

Transcription Error
Mistakes due to incorrect character typed/copied in (Eg. 7382 -> 7482)

Transposition Error
Mistakes due to two adjacent digits interchanged accidentally (Eg. 7382 -> 7832)

Data Validation
Ensures data entered is sensible, reasonable and accurate before data is stored or processed further
Eg. **Presence** (not left blank)
Eg. Check Digit - checks Transcription and Transposition error
Eg. **Type** / Length / Range (Date) / **Format** check (correct format, DDMMYY)

Data Verification
Protects integrity of data by ensuring information being entered into system is correct, matches original source
Eg. Double Entry: User can be requested to enter number two times
Eg. Proofreading: User can be requested to proofread, comparing number and hardcopy

Constructs

Sequence: the order in which instructions execute
Selection: determines the path the program takes when it is running
Iteration: execution of codes multiple times at run-time, can be count (for) or condition-controlled (while)

Readability
Comments: to explain useful descriptions of what code does
Proper and correct indentation: Use of white space and line breaks on codes
Meaningful names: given to Variables, Methods and Functions in programming

Errors
Logic: does not result in program to crash, **runs not as intended**
Runtime: happens during execution, not detected when compiled, revealed when code runs
Syntax: does not follow **syntax rules** of language Eg. Misspelling, Parenthesis, Incorrect Format

Eliminate errors
Test Plan: ensure programs **run as intended**, **eliminate errors**, with end in mind
Desk Check: test logic of algorithm run by computer in step by step manner, to understand, protect and predict potential logic error
Trace Table: table with column for each variable that record changing values

---

Time complexity

It only describes how the execution time scales with the amount of data and is not an indication of speed **for a particular data size** so it is possible for certain sorts to be faster than others in small arrays or when the data is in the first n indexes of array.

Insertion Sort

Best-case:O(n) **Average:** O(n2) **Worst-case:**O(n2)

1. Insertion sort partitions the sequence into 2 parts, 1st part is sorted integers contains the first element in sequence is sorted at initialisation, and 2nd part contains remaining unsorted.

2. For every iteration, each element in unsorted part will be removed and compared **sequentially item by item with every element in sorted part**, until correct position for insertion in sorted part.

3. Continues until whole list is sorted or unsorted part is empty.

Quick Sort

Best-case: O(n log 2n) **Average:** O(n log 2n) **Worst-case:** O(n2)

1. Chooses the first item in the array as the pivot. Partitions by moving the items in the data set about the pivot items to smaller than pivot to the left and larger to the right. Pivot will be in a position that partitions(split) the data set into 2 parts.

2. Recursively performs tasks on each sub-array partitioned until data until 1 item remains, and the data set will be sorted.

Selection of Pivot

The ideal pivot is to choose a pivot that will result in two sub-arrays of **equal size after the first pass**. So in an unsorted array of integers **the median value of the integers should be chosen as the pivot done by linear search** has to be done to find the location.

For data sets that are sorted, choosing the **first or last position of the pivot will lead to the worst-case** O(n2) time. If **random selection** is used, the chance of selecting a **worst-case pivot is largely reduced**, which will result in better performance.

Insertion Sort	**Quick Sort**
For already sorted data sets, insertion sort is more efficient, it **reduces the number of comparisons by stopping when it finds an**	Quicksort **depends on the pivot choice**. Its best case is O(n log n), but the worst case is $O(n^2)$ if the pivot is poorly chosen. (Full)

smaller element. It requires only a few O(n) passes to place any out-of-order elements.	
Insertion sort is a **stable** sorting algorithm, it preserves the **order of equal elements**, making it efficient for nearly sorted arrays by **avoiding extra time and memory use**	Quicksort is an **unstable** algorithm, alters the order of equal elements and may be less efficient, especially if the **pivot choice leads to excessive comparisons and memory usage.**

Merge Sort
Best-case:O(n log 2n) **Average:** O(n log 2n) **Worst-case:**O(n log 2n)

1. Consistently divide the array by half recursively until the base case of 1 item remains
2. Take **linear time** to recursively merge the two halves.

Bubble sort
Best-case: O(n) **Average:** O(n) **Worst-case:** O(n2)
Eg. If x = 10 , n2 = 102 = 10 000

1. Repeatedly steps through the list, compares **adjacent** elements, and swaps them if they are in the wrong order.

2. The pass through the list is repeated until the list is sorted. The best case occurs when the list is already sorted.

Binary Search
Best-case: O(1) **Average:** O(log n) **Worst-case:** O(log n)

1. Start by determining the number of data and compute the index of the middle element.

2. Compare the search key to the middle element. If they are equal, the search key is found, and the operation terminates. If the search key is larger, repeat (1) on the upper half of the dataset with values larger than the middle element. If the search key is smaller, repeat (1) on the lower half of the dataset with values smaller than the middle element.

3. Stop when subdividing the data set **can no longer be done when size = 1**, or when search key is found.

Linear Search
Best-case: O(1) **Average:** O(n) **Worst-case:** O(n)

1. Dataset sorted by ascending order and iteratively compares an item of dataset with search key starting from item at index one, item by item without skipping.

2. Returns position index of item if found when search key equals to the number and returns zero if all items in the dataset has been examined or if the value of search key less than the value of of current item.

Hashtable	**Search**
The search will make use of the hash function to obtain the hash address and go directly to the location to search for the item, but linear search need to search from the beginning of the data set.	As binary search has a pre-condition that requires the data set to be sorted in some order, sorting the records of a company that possesses a huge quantity of data may be **too costly**.
Collisions might occur in hash tables that increases the search time, and it could be O(n) in the worst case.	Linear Search requires visiting all the elements sequentially until the search key is found

---

OOP and their benefits

A **class** is a blueprint it defines the attributes and methods of objects in that class. An **object** is an instance of a class. Different objects which belong to the same class can be created. (object) LL = linkedlist( ) (class)

3 types of access modifiers
Public, private, protected

Private Class	**Protected Class**
Cannot be directly accessed by member functions of the derived class.	Can only be directly accessed by the member function of the derived class.

Encapsulation

Encapsulation restricts access to private attributes using public methods, involves the bundling of data and methods that act on the data into an object, encourages information hiding, data security, protection and integrity.

Information hiding **Data Security** **Data Protection** **Data Integrity**	Attributes are private, while methods are public so other subclasses can use methods but cannot see and change attributes as classes **restricts direct access attributes** of other private classes. This helps to maintain data integrity/ensuring modifications are done deliberately/ reduces accidental errors/ protecting the internal state of objects from being corrupted.
Hides complexity	Increases usability as encapsulation allows access to a level without revealing the complex details below that level and allows for easier maintenance of code as code changes can be made independently.

Inheritance Superclass

Derive new classes by inheriting the data and operations of existing classes like common traits which is an inheritance relationship, inherits its attributes and methods to extend it further, from general to more specific classes.

• provides a common structure and behaviour that can be shared by subclasses.
• promotes code reuse, override and maintainability.

Eg.The superclass (VEHICLE) contains attributes that subclasses (CAR) and (VAN) inherits with each having its own additional attributes such as (No_seats for CAR) and (Load_vol for VAN).

Code duplication/ reusability	Enables classes to share methods/code, reducing code duplication, **promotes code reusability** (no need to be redefined) code only needs to be updated in one place, reuse common attributes and methods inherited from superclass in addition to its own attributes and methods.
Method Overriding	A chosen method in the superclass, can be extended and **re-defined with its own code to perform differently**, while keeping the same method name.
Simplification	simplifying the maintenance of codes thus reducing likelihood of bugs

Polymorphism

Polymorphism refers to an object's ability to take different forms so same method behave differently on different classes. It allows for code extensibility, different object types to respond to the same method call, without modifying class or method.

Eg. TakeDamage method for the class MINION will behave differently from the same name TakeDamage method in the class WEAPON. A minion takes damage when it is being attacked, while a weapon takes damage when it is being used for an attack.

Complexity	Reduces the complexity of code
Methods <get/set>	Software objects are have a <name of getter/setter>, which can be invoked regardless of type of software object

OOP benefits:

Data protection through encapsulation.
Code reusability via inheritance.
Code generality via polymorphism
Realistic model of entities, each have state and behaviour so easier to design
Abstraction via Methods/Simplification need not know details of their implementation. This allows developers to focus on how to use objects rather than how they work, making it easier to collaborate and work with different types of objects.

Handling data

Data Integrity
Human error: accidentally deleting a row of data in a spreadsheet
Collection error: data collected is inaccurate and lacking information, incomplete
Cybersecurity or privacy breaches: hacker hacks into your company's database to damage/steal information, or employee damages data purposely
Inconsistencies across format: data that relies on cell referencing may be inaccurate in a different format which doesn't allow those cells to be referenced

Data Integrity (CCAV)
The completeness, consistency, accuracy and the validity of the data maintained over time and across formats, recorded exactly as the user intends, retrieval is in exact state as recorded.

Data Security (AADD)
Preventing data from being accessed, altered, disclosed, or damaged without authorisation.

Serial file
A file that stores information in chronological order.

Backup (Data protection)
Keep additional copies of their data which they can recover if the data is lost or corrupted.
Storing the backup copy on **separate medium** and **regularly backing up** daily, weekly and monthly copies to prevent the backup copies of the data from being lost or corrupted due to **hardware damage, due to threats such natural disasters.**

Archive (Data recovery)
Archiving allows data to be stored for long-term retention, while **helping to manage hardware space limitations** (needed in the future).
Archived data is copied to a suitable safe storage medium. The original data is deleted from the computer system to free up resources for new data.

Consequence
The business will lose all its information due to **hardware failure or data corruption**, need to spend extra time to come up with all the information lead to profit loss and loss of trust of clients.

Protection of data
Implement and regularly update / review data policies and practices
Employ authorised professionals to help store and manage data.

Negligence of data
Data may be used for illegal activities such as identity theft or fraud and used to unlock large amounts of personal information of an individual as data (Eg. NRIC) is also a permanent and irreplaceable identifier specific to an individual.

Keys

Keys
Candidate key: an attribute or a combination of attributes that can uniquely identify each record.
Primary key: a candidate key that is most **suited** to become the main key. Ensures **unique** row identification which **results in faster** sorting, searching, and querying operations.

Secondary key: a candidate key that is not chosen as the primary key.
Composite key: a combination of two or more attributes that can uniquely identify each record. (ER Table)
Foreign key: an attribute in one table that references the primary key in another table. Creates a link between two tables and maintains referential **integrity** between the referencing column and the referenced column.

Relationships

Foreign key `TenantID` in table `Rental` references to primary key `TenantID` in the table `Tenant` .

Foreign key `TenantID` in table `Rental` ensures that every `Rental` record **is associated with a valid** `TenantID` where its record is **stored** in the primary key `TenantID` in the table `Tenant`

New Foreign Table

Add a foreign key `ProjectManager_ID` to `PROJECT` table that references to a new table `PROJECT_MANAGER` that has primary key `ProjectManager_ID` and a foreign key `Employee_ID` that references to the primary key `Employee_ID` of the table `EMPLOYEE`.

Normalisation

The process of organising the tables in a database to reduce **data redundancy and inconsistency.**

1. **First Normal Form (1NF)**
   - All attributes should hold only atomic values (each record has to be unique, no duplicate/repeated values in each row).

2. **Second Normal Form (2NF) → 1 PK only**
   - In 1NF
   - All non-key attributes should be fully dependent on the entire primary key.
     Eg. 2 Primary keys (In employee table, PK is Employee_ID, but values dependent on Skill_ID PK)

3. **Third Normal Form (3NF) /2023**
   - In 2NF <state requirements>
   - No transitive dependencies - all fields must only be determined by the primary
   - no **partial and non-key** key dependency

Consequence / Not managed properly

Disadvantage
There will be a case of **data redundancy** as cars of the same Category will require the same DayRate to be entered **into the table more than once** which can in turn lead to **data inconsistency** if an insert, update or delete **error is made**. (category determines Day Rate, need to enter twice, eg, luxury → $50, enter both luxury and $50, may enter wrongly), compromises **data integrity AlvQn**
Partial update: Since there exists multiple record location with Skill_Name and Cost_Per_Hour **repeated**, update anomaly could be **data inconsistency** may arise which may result in some electricians with rates updated and some not updated, compromises **data integrity**. (ownself go change)
It will be bigger in size which will waste memory space and make less efficient.

Data inconsistency	when the same data element stored in multiple locations within a database is found to contain **different information**
Data redundancy	when the same data element exists in **multiple locations** within the database.

NoSQL	**RDBMS**
Greater **flexibility**, when handling changes like adding new fields or data types can be done dynamically without having to modify an existing schema	**Less flexible**, RDBMS systems **enforce rigid** schemas, requiring significant effort to adjust table structures when changes are needed.
More cost-effective, as open-source and can use more affordable, commodity hardware.	**More expensive**, RDBMS have separate software which needs to be purchased, need hire experts to manage sys.
Scale horizontally, **less expensive** to increase performance (add more machines without the need for high-end hardware)	RDBMS **scale vertically**, upgrade to more powerful servers (exp), **more expensive** to increase performance
support **hierarchical** data storage, where less frequently accessed data can be moved to more cost-effective storage solutions, optimising storage costs and performance.	
NoSQL databases do not promote **data integrity.**	RDBMS promote **data integrity**.

NoSQL databases do not support **complex queries.**	RDBMS support **complex queries** (FK).

American Standard Code for Information Interchange (ASCII)
Character encoding standard for electronic communication. 7-bit character code where each value represents a **unique character**.
Eg.ASCII represents lowercase letters (a-z), uppercase letters (A-Z), digits (0–9) and symbols

Unicode
Computing industry standard for consistent encoding, representation, and handling of text.
Eg. **UTF-8:** represent more symbols, characters and languages when compared to ASCII as it has 8-bit than 7-bit character code for ASCII.

Network Security

Internet Protocol (IP) address
A numerical label assigned to each device connected to a computer network (WiFi) that uses the Internet Protocol for communication.
IPv4 address consists of 32 bits, written in **dotted decimal notation** with 4 decimals separated by dots. XX.XY.YY.YX

Host allocate IP Address
IP Address can be allocated dynamically(own self) when the host joins the network via Dynamic Host Configuration Protocol (DHCP) or by configuration of host hardware or software.

Local Area Network (LAN) / Wide Area Network (WAN)
Computer network (WiFi) that connects computers over a small/large geographical area. Devices can be linked using cables or WiFi. Examples include the WiFi and Ethernet. Examples include 4G and the Internet.

Router TCP/IP
At the Network (start) layer, router uses IP address to send data packets to the designated host (server) in the network.

Switch TCP/IP (network)
At the Data Link (end) layer, switch uses MAC address to send data frame to the designated machine (device) in the **local** network.

Trojan
Trojans hide malicious code that is executed when a file is opened which is sent to users as an innocent-looking file attachment. If opened, it could search for sensitive information on the devices and send it to the attacker.

Worm
Worm is a type of malware that replicates itself by inserting its own code into other resources without human activation.

Virus
A virus is added to a program that a device downloads and executes. If executed, it could log keystrokes of the device and leak other kinds of information to the attacker.

How Worm, virus, trojan, ransomware, spyware transferred from internet to computer

1. Source: Downloading from FTP or webpage or Email attachment
2. Defend: Firewall, Antivirus software

Describe P2P network Risks
The absence of a centralised server (moderators) would mean that any device connected to the requesting (exposed) device share some resources to the device so both parties have no control (works both ways) on what content is being transmitted from the **senders and the receiver**, which carry risks such as data integrity, viruses, spyware, adware, and unwanted files.

	Client-Server Architecture	**Peer-to-Peer (P2P) Network**
Definition	Computing model where server provides the service and client requests for the service. Server has higher privileges than the client.	Computing model where all peers provide and request for the service. All peers have equal privileges.
Management	Client-Server Architecture is easy to maintain.	P2P network is difficult to maintain.
Security	Client-Server Architecture is secure.	P2P network is insecure.
Cost	Client-Server Architecture is expensive to maintain.	P2P network is cheap to maintain.
Reliability	Client-Server Architecture is unreliable as the failure of the server leads to failure of the client.	P2P Network is reliable as the failure of one peer does not lead to failure of other peers.

How data stored complies with PDPA for Business (Data security)

Confidentiality	To ensure that private data that is disclosed only to authorised persons and protected from any unauthorised access.
Consent	Ensure that customers have given consent to the collection of data. Notify customers about the data sharing arrangement, including how their data will be used and protected.
Availability	Data, network resources and services, **policies** are **continuously available** to authorised users whenever they **require** it.
Integrity	To make sure that data is accurate, complete and consistent, and not changed by unauthorised personnel.
Purpose	Ensure that the customers are aware of the purpose of collecting, using or disclosing.
Necessity	Cease retention of data if it is no longer necessary.

Describe DoS Attack

Denial-of-Service (DoS): malicious user attack floods an internet server with traffic or sending it information that triggers a crash, overloads memory and bandwidth, making it inaccessible to its intended users.

Prevent

Huge Spike	To notify detection of a huge spike in site requests of malicious intent.
Approaching Limits	To notify that the server is **approaching its limits** in resources to serve incoming requests.

Firewall

1. A barrier between a trusted internal network and incoming traffic from untrusted external network, such as the Internet before LAN

2. A software or hardware device, or both which monitors and controls incoming and outgoing network traffic based on predetermined security rules.

3. To **block malicious traffic** (viruses, hackers), and **allow incoming traffic** as due to requests from internal hosts based on source address, destination address or protocols.

Limitations:

• Hackers can bypass firewall by inserting malicious attacks in legitimate programs. Eg. emails

• Cannot protect against internal attacks. Eg. virus in PC in the network

• May block some legitimate programs

How to ensure it is only accessible to Intended Recipient to protect confidentiality
Using public key cryptography: The sender encrypts data using the **receiver's public key (address)**, and the receiver **can decrypt it with their private key**, data **confidentiality**.

Describe how the sender creates a digital signature to ensure message is from Intended Sender (Sending)

1. Digital signatures prove a message was not modified from the time it was signed, by generating a hash of message (**to obtain a digest**) and encrypting it using the sender's private key unique to the message. The hash generated is unique and changing any part of it will completely change the hash.

2. Sender attaches digital signature together with the original message to send to receiver.

How the recipient verifies authenticity and integrity of a digital signature from receiving messages

1. Since the message is encrypted using sender's private key, the only way to decrypt the message is by use of sender's public key. Recipient will use the public key of the sender to decrypt the digest. If the recipient can successfully decrypt the message it means the sender is authenticated. The recipient would also put the message received to the same hash algorithm to obtain a digest of its own.

2. Recipient will then compare the decrypted digest it receives to the digest it generates. If they match, the message has not been modified and the digital signature is authentic. If the hash value does not match, signature is changed, prevents the message from being decrypted, provides proof of origin and ensures data integrity.

Encryption of Messages **(Lock)**	Ensure it cannot be read without the correct decryption key. **Benefit**: Protects data confidentiality and prevents unauthorised access. **Limitation**: If encryption keys are compromised, the encrypted data becomes vulnerable.
Digital Signature	Apply digital signatures to data using a private key to verify the sender's

(Unopened seal) **(Receive/Seal)**	identity, ensure the data has not been altered and to validate the signature. **Benefit**: Provides authentication and integrity, confirming the origin of the data and verifying it has not been tampered with. **Limitation**: If the private key is compromised, the signature's security is undermined.
Firewall	Firewalls need constant monitoring and updating to remain effective.
Digital Certificate **(Ownership)**	To authenticate the identity of individuals, devices or servers and facilitate secure communication, is used to prove the ownership of a public key. This enables clients/customers to trust the public key to improve security through encryption / identifying phishing sites / <other points of PK>

Extra Protection company/user to ensure account is not hacked/breached
Measures for Network Application Security (unauthorised access)

Authentication	Firstly, ensuring that the client accessing data belonging to an account in the server is authorised to do so. (MFA + PW) Secondly, ensuring that the server is the system that it claims to be. (DC)
Password + MFA (password)	Implement strong authentication methods like MFA, at least two forms of authentication for all user logins. Eg. password, OTP, biometric data **Benefit**: Ensures that only authorised entities can access or initiate data transfers, reduces the risk of unauthorised access even if passwords are compromised verify the identities of users and systems. **Limitation**: Increase complexity and is inconvenient for users
Centralised Password Management System	• Enterprise-grade password manager that generates, stores, and manages complex passwords. • Enforce strong password policies (length, complexity, regular changes) • Implement role-based access control to limit password visibility and sharing. • Automatic password rotation and real-time monitoring for suspicious activities.
Digital Certificates	To authenticate the identity of individuals (company), devices or servers and facilitate secure communication, is used to prove the ownership of a public key with **permissions set so that they are not tampered with**. This enables clients/customers to trust the public key to improve security through encryption / identifying phishing sites / <other points of PK>
Authorization	

User Account Controls	Administrator to **set permissions for different accounts**. Eg. staff accounts would be given more access to information and student accounts have limited permissions to edit information.

How a breach happens, how to prevent it, and how to correct it
Due to human error, misconfiguration of access controls or security settings, granting unauthorised users access to sensitive data or systems.

Preventive measure: Implement a "four-eyes principle" for critical system changes, require two authorised personnel to review and approve modifications before they are applied.

Corrective measure: An automated rollback mechanism that can revert system configurations to a good state due to misconfigurations and unauthorised changes.

Describe Domain Name System (DNS) URL for an IP address. / (manager → direction → boss → employees)

1. The client proposes a domain name resolution request (youtube.com) and sends the request to the local DNS.

2. The local DNS receives the request and queries (search) the local cache. If there is such a record, the local DNS directly returns the result. If not, the local DNS sends the request to the root DNS which returns the primary domain name (subdomain of the root) of the local DNS's domain.

3. The local DNS sends a request to the Top Level DNS returned in the previous step, and the server that accepts the request queries its own cache. If there is no such record, it returns the address of the relevant lower-level Authoritative DNS and repeats until correct record found.

4. The local DNS saves the returned results to the cache for the next use and returns the results (webpage IP address) to the client and store the IP address in local cache for future use.

Native application
Utilises the functions on the customer's personal device which removes the need for the customer to scan the QR code to make payment (China), the customer could easily use their online banking to make payment.

Native application

Touchscreen to capture thumbprint or signature for online payment.
Save the previous orders in their personal devices for easy reference, especially for returning customer.
Customer able to log in with their credential to allow transacting using their identity.

Native Application	**Web Browser**
Safer to use as harder to encounter fake apps.	Unsafe as it is easy to create fake websites.
Allows **Offline usability** as app is fully downloaded	Customer can order from the website using a browser needing to install any software.
Time consuming and Costly for maintenance, testing, updating	Time consuming and Costly for maintenance, testing, updating
Less flexibility as it is platform specific	More flexibility as not platform specific
Enhances usability as there are optimised aspect ratio	Customer can **order remotely** without being in the restaurant (pre-order).

Network Application

Explain the need for communication protocols in a network.
How requests are handled using Communication Protocol (Emailing)

1. Communications protocol **ensures the proper transfer of data between devices** that **specify format of data (FTP), and signals to start, control, end the transfer**. to initiate, facilitate exchange of data based on rules that both senders and receivers agree

2. The web server will differentiate the requests and responds accordingly. If it is a **web request**, server uses HTTP to deliver webpage to web browser. If it is a **file transfer request**, use FTP to upload or download files, for **email** through IMAP for email retrieval. If the requested page, file or email does not exist, the server will respond with an error message.

Describe the roles of switches and routers in a computer network.
Switch uses MAC addresses to forward data frames between devices within the same network. Router uses IP addresses to send data packets between devices in different networks.

Describe TCP (Transport)
Network protocol at the transport layer of TCP/IP model which **ensures the security and validity** of data by using a **three-way-handshake** to establish a connection for data transmission.
Data is broken into segments with sequential numbers sent from one **node(LL)** to another on the internet for reassembly at the **receiver**.

Layering TCP/IP	**Function**
Network Layer	Find the best route to deliver data package.
Application Layer	Consists of applications that uses the network
Data Link Layer	
Physical Link Layer	Transform data package to electrical signals and transmit to physical devices
Transport layer	Establish connection between applications of sender and receiver.

Layering TCP/IP Advantages
Provides **design modularity** (changes to a single layer without affecting others.)
Enables programmers to **specialise** in a particular layer of the model (efficiency)
Simplifies the network model.
Allows for **standardised interfaces to be produced** by developers.

Explain what is data Packet Switching and how it is resilient to damage in the network.

1. Packet switching is a connectionless network switching technique where data is divided and grouped into units called packets. These packets are independently routed from the source to the destination.

2. Each packet has a **header which has source and destination addressing information** used by routers to **direct** the packet to its destination and a **payload** which carries the data being transmitted.

3. A packet is **transmitted when it is available in a node, separated** and **sent from one node to another on the internet independently**, finding the best

route to the receiver based on its header information. (path can be modified to work around changes (if pc is damaged) in the network.) At the receiver's end, packets are reassembled correctly in the correct order using Sequential numbering.

Packet Switching vs Circuit

Circuit switching has call **setup delay.**	Packet switching is **not synchronous has delays**, less suitable for real-time applications (like voice calls, where **delays** can occur due to packets arriving out of order and needs to be reassembled.)
Different packets may travel in different route and thus more efficient, **saves bandwidth and avoids congestion.**	Packet loss needs to be **retransmitted which lead to delays**
If a packet is corrupted, only that specific packet needs to be retransmitted, makes error correction more manageable and **reduces the time required to resend**.	High traffic can lead to **congestion**
If one node is not operational due to a broken cable, the router will **select an alternative route** for the packet to travel via another node to the destination.	Packets take different paths so they will arrive at the destination out of order, **need time to reorder.**
It is also **more secured** since it became much harder to attack all the routes instead of one route in circuit switching network.	(Circuit switching path is established for the entire conversation, less secured)
Smaller packets can be sent over multiple paths enables **faster transmission**.	(Circuit switching path is established for the entire conversation, slower)
Packet switching has dynamic bandwidth, saves resources and congestion.	Circuit switching has fixed bandwidth, waste of resources and congestion.

Circuit switching	It provides a **dedicated path** for communication, avoiding these delays.
Data Compression	It reduces the size of data to **speed up transmission**.
Quality of Service (QoS)	It **prioritises** certain packets in routers to ensure faster delivery.

Router (Packet)

A router connects different links to transmit an incoming data packet from input link to an output link. It **checks IP packet headers to find the destination address** using a **routing table** of known networks and best routes which **to send the packet on as the next link**. If the packet is not fully received, it will buffer or store the packet's bits and only transmit when all the packet's bits have been received.

Code of ethics
Prioritising on-time delivery over fixing a fault in a nuclear power station's control system risks catastrophic consequences, violating the **principles of safety, public welfare, and legality.** Ethical standards require that critical errors be **resolved before deployment**, especially in high-stakes environments like this, where lives are at risk. The suggestion to issue a patch later is irresponsible and **fails to meet professional and legal obligations.**

Ethical Company Guidelines (Selling, Advertising, Developing)

1. User education: Provide security training for those accessing high security systems

2. Incident response plan: Develop a comprehensive plan with specific procedures for each tier, ensuring quick and appropriate responses to potential breaches

3. Company's responsibility to keep software up to date against latest kind of viruses, and push updates out to users

4. test software thoroughly before release, and resolve bugs that they have discovered

5. Regular security audits: Conduct periodic penetration testing and vulnerability assessments, prioritising higher tiers

6. Make sure software works as advertised, and not make false claims

7. Network segmentation: Implement VLANs to separate different tiers, limiting potential breach impacts

Professionalism	Enhance the prestige of the profession and organisation. Use computing skills to benefit society. Do not harm society for personal benefit.
Responsibility	Follow work standards/guidelines, fulfil assignments before due date. Acknowledge mistakes and correct them when they are made.

	Eg. Carrying out duties in violation of the company's standard operating procedures.
Integrity	Act with true competence, maintain discretion with confidential information, **practice impartiality, declare conflicts of interest,** and give proper credit to others' work. Eg. selling confidential information about personnel or organisation for personal or financial gain.
Competence	Improve skills and stay informed about developments in your field. Increase public knowledge about computing. Support and encourage professional development for employees.