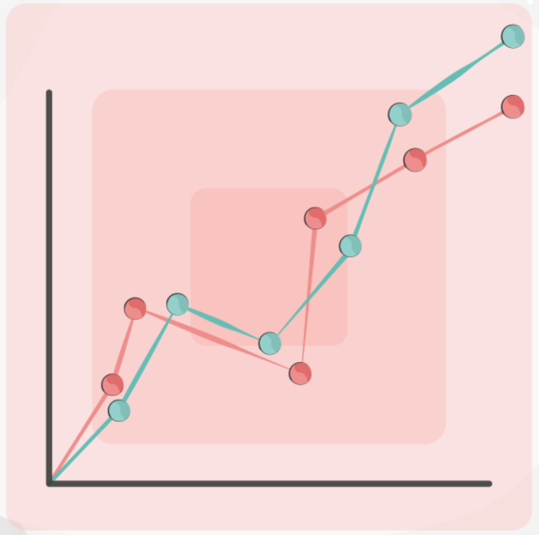
Level 5 Data Engineer Module 4 Topic 3

Designing secure architectures

Welcome to today's webinar.



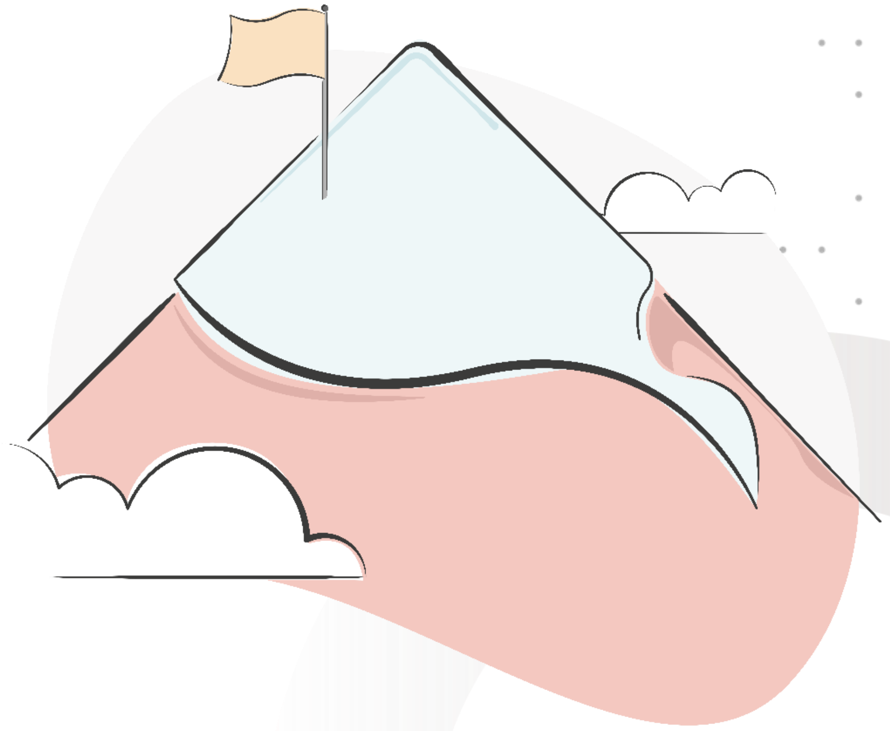
Session aim and objectives

Building Careers Through Education

Everybody Matters.
Stronger Together.
Student, Learner and Client Centric.
Trust and Respect.
Embrace Change.

This webinar supports the following learning outcomes:

- Implement functionality of data products and data pipelines in a way that protects data at rest and data in transit using a zero-trust approach
- Make use of advanced security techniques to safeguard data products and data pipelines against unauthorised access and security breaches.
- Practise visual methods of designing secure architectures through diagrams and flowcharts to improve data product and data pipeline communications



BPP

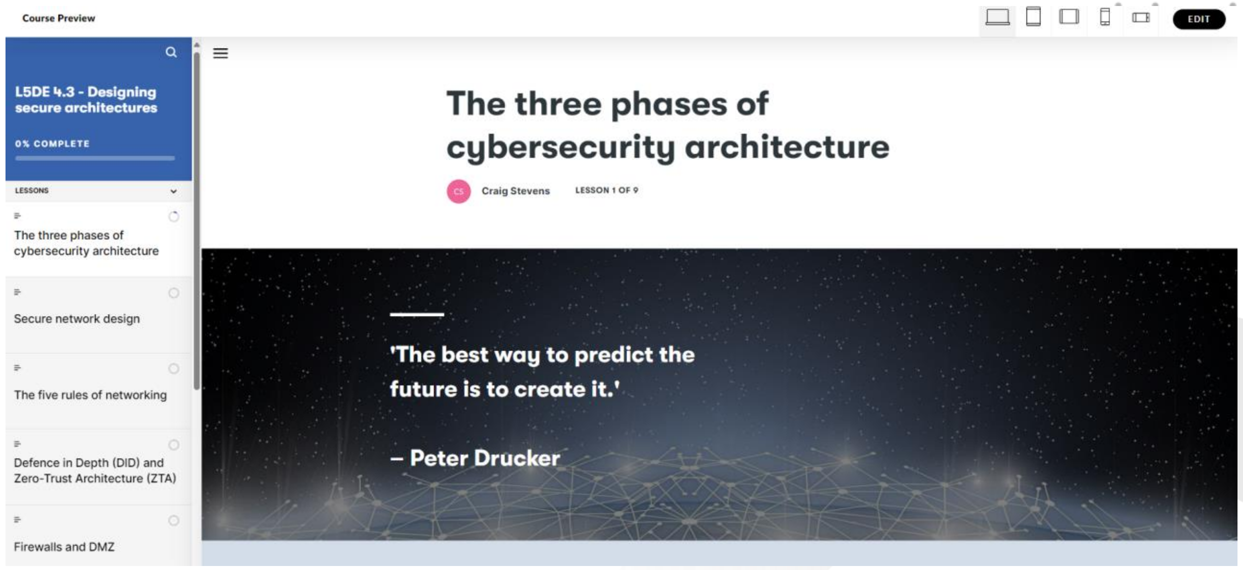
Recap of e-learning

Building Careers Through Education

Everybody Matters. Stronger Together. Student, Learner and Client Centric. Trust and Respect. Embrace Change.

Are you happy with your learning?

- What are the three phases of cybersecurity architecture?
- What are the five rules of networking?
- **"Implicit Trust Is Never Assumed, Always Verified" – *what do we call that?***



Course Preview

EDIT

L5DE 4.3 - Designing secure architectures

0% COMPLETE

LESSONS

The three phases of cybersecurity architecture

Secure network design

The five rules of networking

Defence in Depth (DID) and Zero-Trust Architecture (ZTA)

Firewalls and DMZ

The three phases of cybersecurity architecture

CS Craig Stevens LESSON 1 OF 9

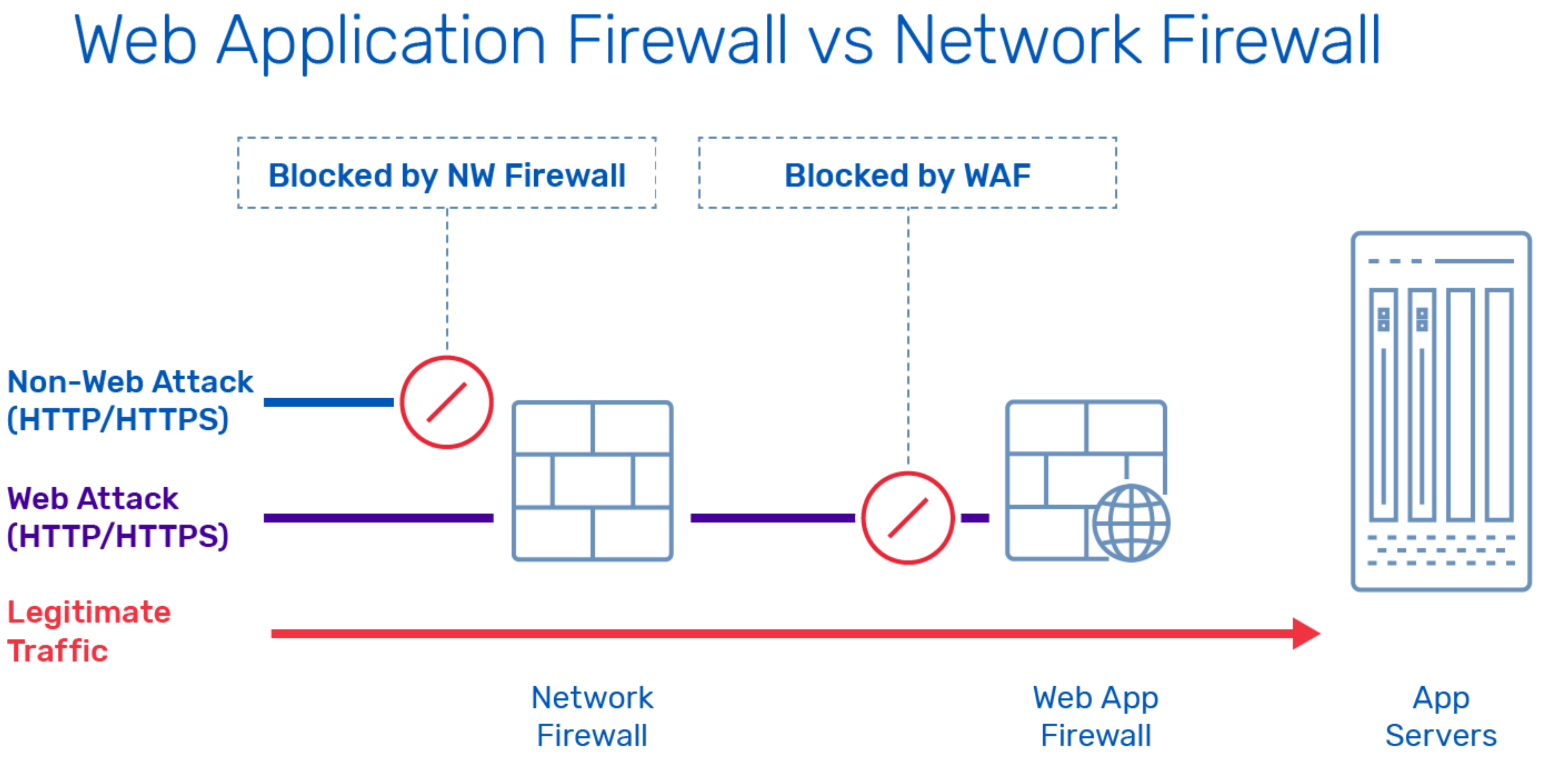
'The best way to predict the future is to create it.'

– Peter Drucker

A screenshot of topic 3 e-learning

Recap of e-learning

Are you happy with your learning?

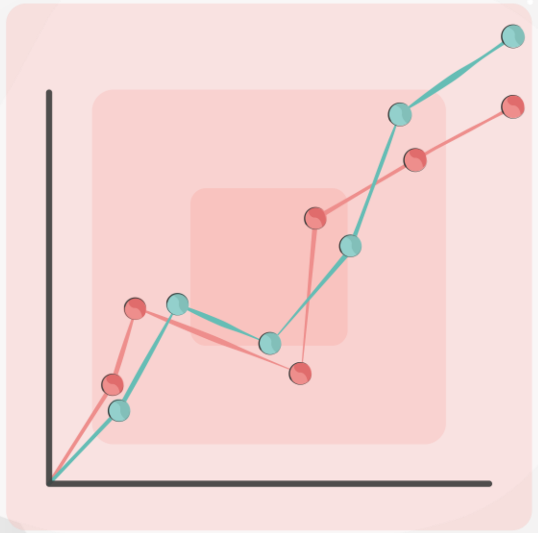


Recap

can you explain the difference between authentication and authorisation?

Authentication
user name
****
LOGIN
Authorization

Secure by design



Useful definitions

Slide 1 of 2…

Service

- A special software interface that enables access to a particular functionality. Usually runs automated tasks.

Service account

- A special account with escalated privileges that is used to run services (e.g. for back-up or monitoring). Not supposed to be used by humans (except for maintenance). A service account can also be created to store data or configuration.

- Kind of like a “backstage pass” at a music festival.

Useful definitions

Slide 2 of 2…

Externally-exposed enterprise assets

- Enterprise assets (end-user devices, networked devices) that can be discovered using a Domain Name System scan or a Network scan from the public-facing Internet (outside of the organisation’s private Intranet)

Secure by design

What does this mean…?

- Building systems with security as a foundational and integral component from the outset, rather than as an afterthought.
- Anticipate threats and vulnerabilities during the design phase.
- Security isn’t just a phase; it's a continuous mindset.
- Proactive approach.
- Reduces long-term costs (fixing vulnerabilities later is more expensive).
- Increases trust with stakeholders and users.
- Protects user data and system integrity.

Secure by design

An example…

- A house built on a robust foundation with integrated alarm systems, locks, and secure windows, rather than adding locks and alarms after the first burglary has already happened.

More useful definitions

- **Threat Modelling**:

This involves identifying potential threats and vulnerabilities specific to an organisation or system. By understanding the threat landscape, appropriate measures can be put in place.

- **Redundancy and Resilience**:

Building systems that can withstand attacks and continue to operate, or recover quickly after a breach, is crucial for maintaining business continuity.

- **Security Controls:**

Mechanisms implemented to reduce or eliminate identified risks and threats.

Security control frameworks

Why…?

Think of them as building regulations

- Such as skyscrapers have rules for fire exits, building materials, safe electrical wiring, etc., computer systems including networks also need a framework for security controls to prevent accidents.

Computer security controls tackle:

- Securing data
- Managing who has access to certain information
- Guiding the organisation in fighting common threats, etc…

Without them, systems would be prone to being exploited and compromised, just like skyscrapers would be prone to collapsing.

Secure architectures

- **Holistic Approach:** Secure architectures refer to designing and implementing big systems (like web applications, databases, networks) that inherently resist malicious attacks and unintentional failures.

- **Emerging Challenge:** As technology evolves, so do threats. Secure architectures need to adapt to new paradigms, such as IoT or Cloud environments.

- You should cultivate a **mindset of continuous learning**, regularly updating their knowledge and staying aware of new threats and mitigation techniques.

Two security architecture principles

Simplicity

- The simpler the system, the less can go wrong
- Minimising inconsistencies
- Easy to grasp

Restrictiveness

- Minimising access is key
- Communication should be inhibited
- Boundaries should be tight

Discussion

Voting Software

Accuvote TS

- 30k+ lines of C++

PRUI (Yee *et al*)

- Less than 300 lines of Python
- http://zesty.ca/voting/

Which one would you trust to handle voting securely?

Submit your responses to the chat!

Patterns – invaluable to architects

Patterns are reusable solutions to common problems faced during system design and development.

- ***Efficiency***: Reduce time to develop by leveraging proven strategies.
- ***Reliability***: Tested solutions lower the risk of new issues.
- ***Consistency***: Ensure uniformity in design and development.

Understanding patterns provides a roadmap to efficient and effective secure system design

Pattern examples

Principle of Least Privilege

- This foundational pattern ensures that a system component (user, application, process) should only be given the minimal levels of access necessary to perform its function.
- **Use Case**: In a company's data access system, employees can only access the specific data necessary for their role, preventing unnecessary exposure of sensitive data.

Defence in Depth

- **Defence in Depth:** Layering multiple security measures to provide redundancy and protection at various levels of a system.

Economy of mechanism

- Keeping mechanisms **simple** (“economical”)

- Simplicity refers to **all dimensions**: design, implementation, operation, interaction with other components, even in specification.

- The **toolkit** philosophy of the UNIX system is an example; each tool is designed and implemented to perform a single task. The tools are then put together. Think **modular** design (compare: OOP with SoC).

- This allows the checking of each component, and then their interfaces. It is conceptually much less complex than examining the unit as a whole. (Think integration vs **unit** testing).

Fail-Safe Defaults

- If the protection system fails, then legitimate access is denied but illegitimate access is also denied
- **Also:** *Base access decisions on permission rather than exclusion*
- **Also:** *… set everything to **DENY** and add exceptions*
- **Example:** failure of web server shouldn't expose user passwords

Complete Mediation

- Every access to every object must always be checked
- *“Check every time”*
- *“Assume permissions may change in real time”*

Open Designs

- Prevents the “security by obscurity” anti-pattern
- Principles and methods of design are publicly known.
- Implementation details are not secret
- **Hiding methods or data does not secure your system**
- Example – GSM Algorithms designed in secret

Separation of Privilege

- Requires at least 2 actors or components to implement a secure action
- E.g., a proposer and an approver
- You cannot approve your own proposal

Formally: *Where feasible, a protection mechanism that requires two keys to unlock it is more robust and flexible than one that allows access to the presenter of only a single key.*

Example: root / admin account

- Most operating systems use admin account
- Any privileged action requires admin privileges
  - ➢ All-or-nothing access

Least Privilege

Every program and every user of the system should operate using the least set of privileges necessary to complete the job.

A subject should be given only those privileges necessary to complete its task:

- Function more important than identity
- Rights added as needed, removed if not needed

Least Common Mechanism

Minimise the amount of mechanism common to more than one user and depended on by all users.

- *Every shared mechanism (especially one involving shared variables) represents a potential information path between users*

Is this a good principle?

- Isolation prevents communication, and communication with something—another process or a resource—is necessary for a breach of security.

Psychological acceptability

It is essential that the human interface be designed for ease of use so that users routinely and automatically accept the protection mechanisms correctly.

Security mechanisms should not add to difficulty of accessing resource:

- Ease of installation, configuration, use
- Avoid frustration and non-compliance

Some practical security principles

Building Careers Through Education
Everybody Matters.
Stronger Together.
Student, Learner and Client Centric.
Trust and Respect.
Embrace Change.

- **Secure Communication:** Ensuring data transmitted between components is encrypted and authenticated.
- **Secure Configuration:** Setting up system components and software securely, following best practices and minimising unnecessary features.
- **Access Control:** Implementing mechanisms to control and manage user access, authentication, and authorization.
- **Monitoring and Logging:** Continuously monitoring for security events and maintaining logs for auditing and incident response.
- **Patch Management**: Keeping software and hardware up-to-date with security patches to address known vulnerabilities.

Anti-patterns

Anti-patterns are Common responses to recurring problems that are usually ineffective and risk being counterproductive.

- ***Awareness*****:** Identifying anti-patterns helps avoid pitfalls in system design.
- ***Course Correction*****:** Provides an opportunity to rectify design flaws early.
- ***Education*****:** Acts as a teaching tool, enabling learning from past mistakes.

Anti-pattern examples

“Hardcoded secrets”

- Embedding secret information, such as passwords or API keys, directly into the code. This creates a vulnerability since the code can be inspected or leaked, exposing the secrets.

Activity discussion

In breakout rooms, pick one pattern that you have seen implemented, or could implement in one of the systems that you work with.

- Discuss the pros and cons of implementing that pattern. Could there be any unintended consequences of applying it?

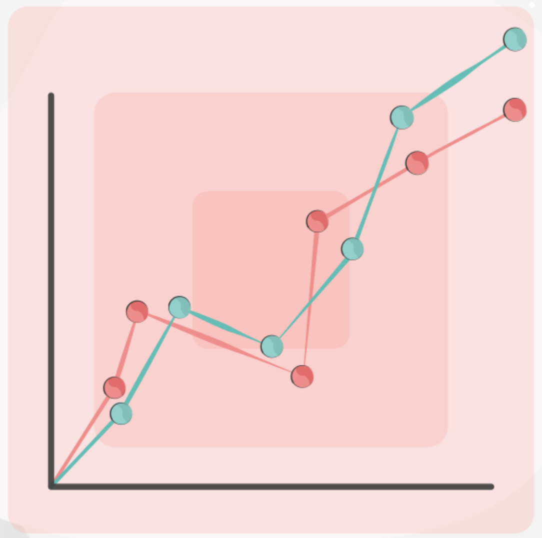
‘Security by obscurity’ is a common anti-pattern.

- **Discuss:** What do you think is meant by that?

- Using your own research, find examples of that anti-pattern in action.

Submit your responses to the chat!

Secure by design



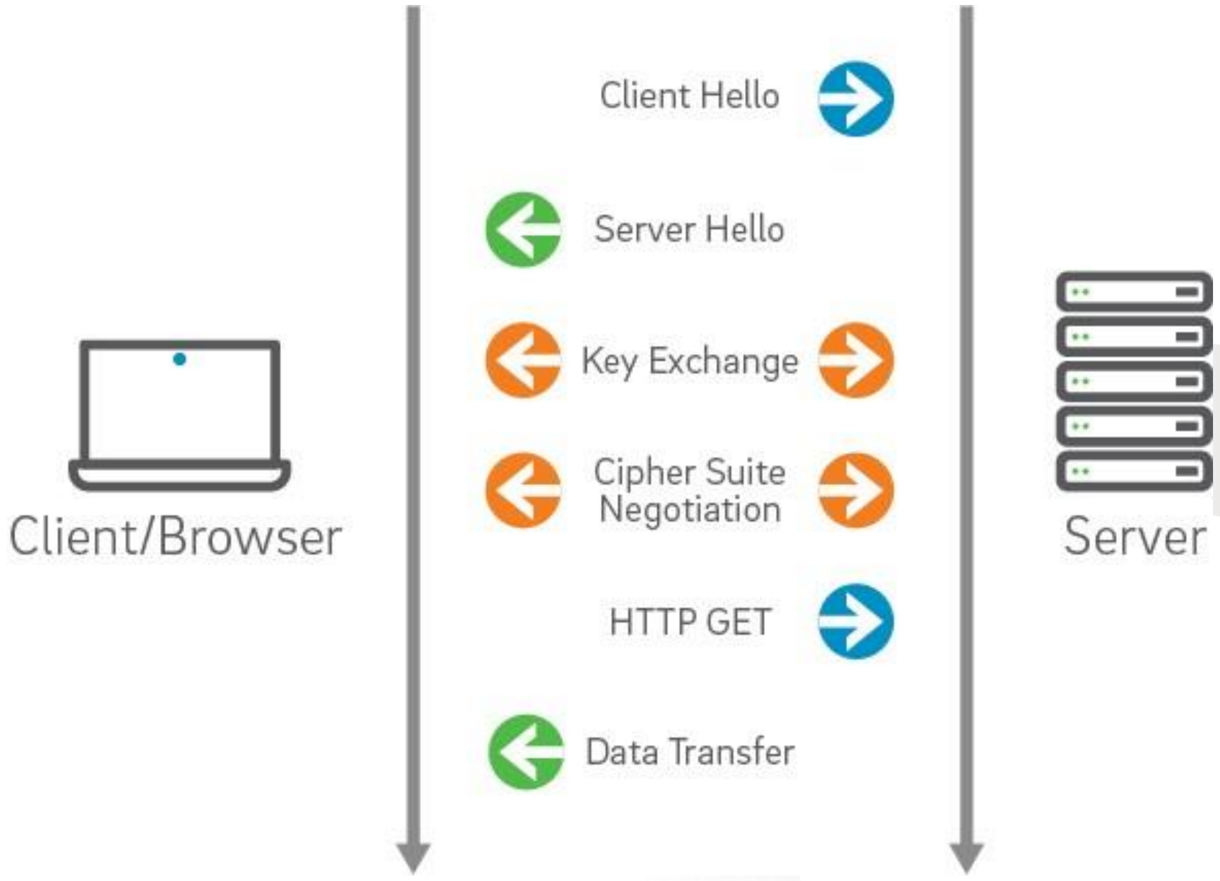
Confidentiality

- SSL/TLS (Secure Sockets Layer/Transport Layer Security)
- VPN Protocols (e.g., PPTP, L2TP, OpenVPN, IPsec)
- RDP (Remote Desktop Protocol)
- SNMP (Simple Network Management Protocol)
- SSH (Secure Shell)

HTTPS

Asymmetric encryption – confidentiality

HTTPS (Hypertext Transfer Protocol Secure) is the protocol used in the communication, which combines HTTP with SSL/TLS to provide encrypted transmission of data.



SSL/TLS

Building Careers Through Education

Everybody Matters. Stronger Together. Student, Learner and Client Centric. Trust and Respect. Embrace Change.

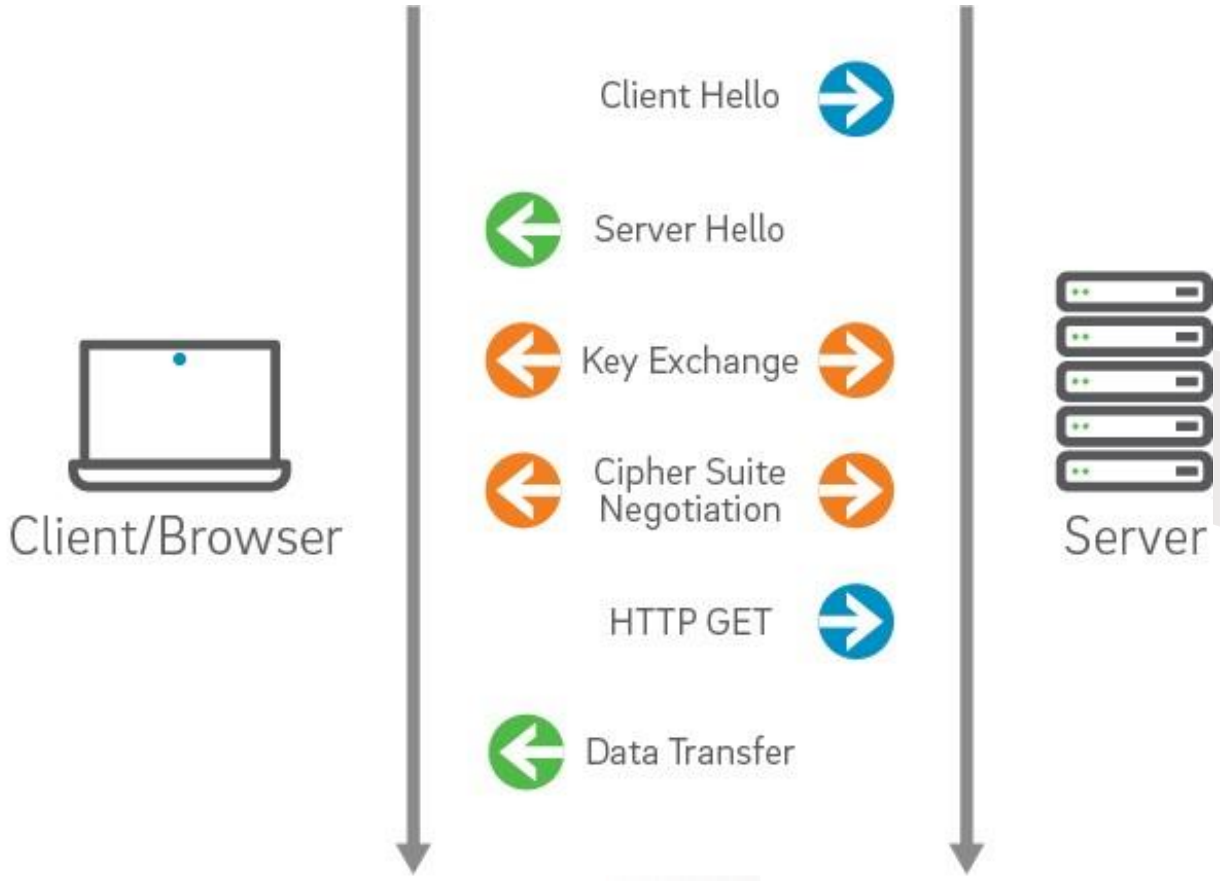
- **What is it?** A cryptographic protocol designed to provide secure communication over a computer network.
- **Why is it used?** To protect data integrity and confidentiality during data transmission, especially on the web.
- **Vulnerabilities:** Heartbleed, POODLE, and downgrade attacks.
- **Biggest Threat:** Man-in-the-middle attacks, where unauthorized parties can intercept and potentially alter the communication.
- **Mitigation:** Regularly update to the latest version of TLS, disable outdated versions, and implement perfect forward secrecy.

BPP

SSL

Does this use symmetric or asymmetric encryption?

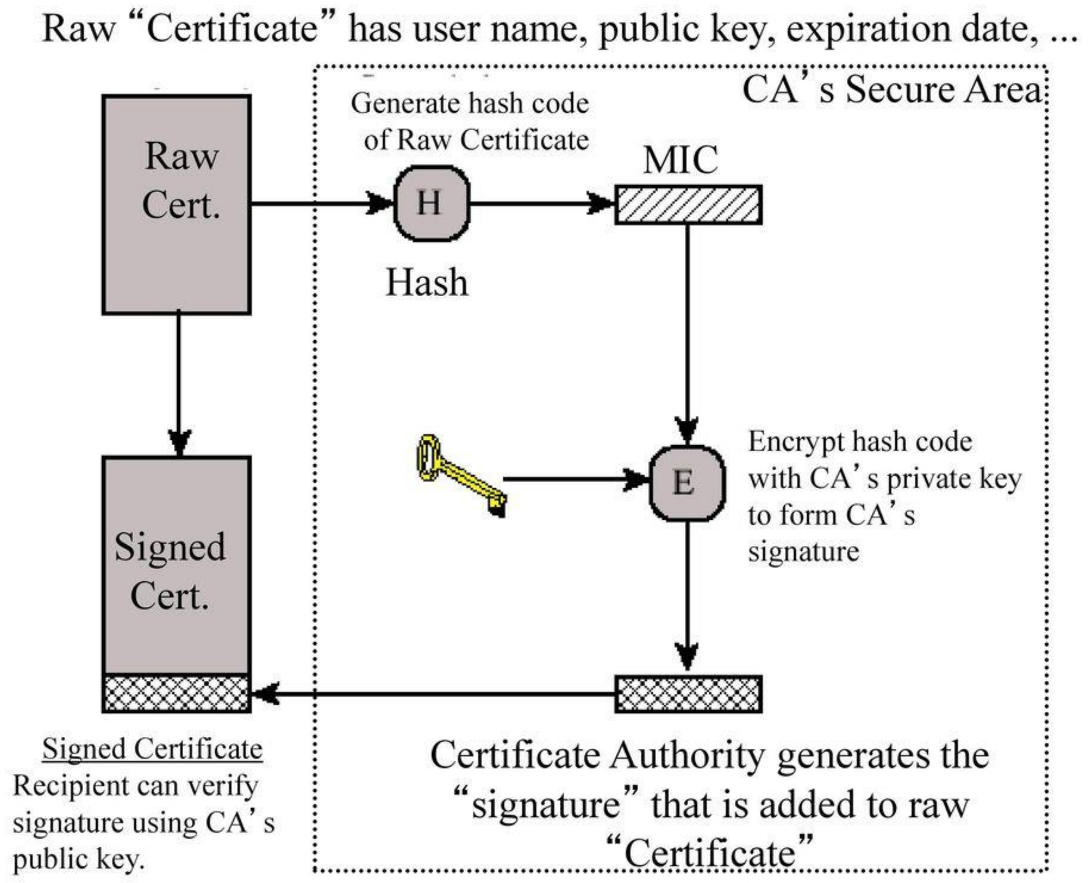
- Both
- Asymmetric encryption is necessary to verify the others identity
- and then symmetric encryption gets used because it's faster.



X.509 Certificates

What you need to know…

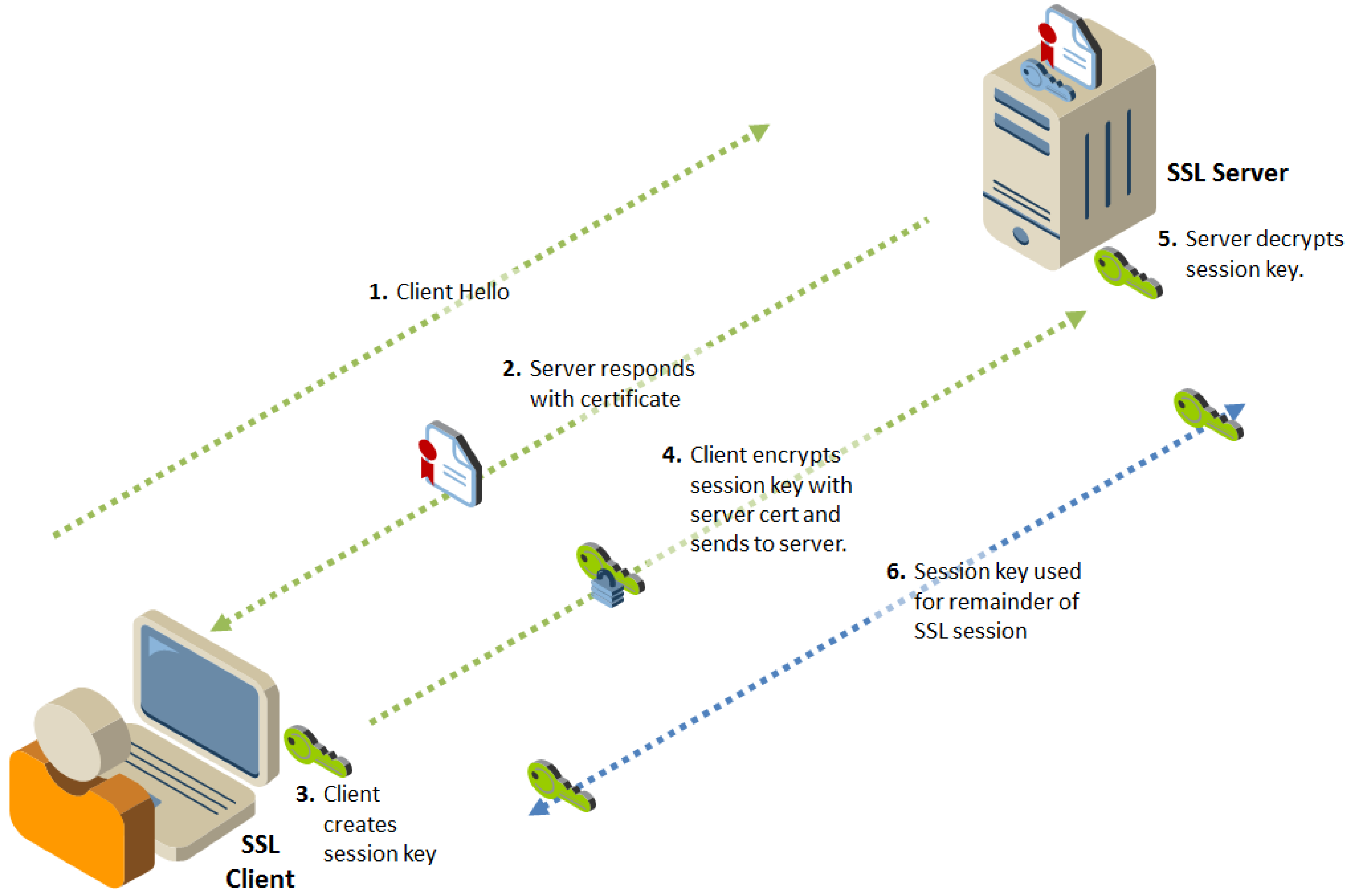
- BSSL is by far the largest use of X.509 certificates, many people use the terms interchangeably.
- They're not the same however
- A "SSL Certificate" is a X.509 Certificate with Extended Key Usage: Server Authentication



Raw "Certificate" has user name, public key, expiration date, ...
CA's Secure Area
Raw Cert.
Generate hash code of Raw Certificate
H
Hash
MIC
Encrypt hash code with CA's private key to form CA's signature
E
Signed Cert.
Signed Certificate
Recipient can verify signature using CA's public key.
Certificate Authority generates the "signature" that is added to raw "Certificate"

Summary of SSL handshake

Portions relevant to key establishment


SSL Server
5. Server decrypts session key.
1. Client Hello
2. Server responds with certificate
4. Client encrypts session key with server cert and sends to server.
6. Session key used for remainder of SSL session
3. Client creates session key
SSL Client

SSL/TLS

***Description*:** A bug in OpenSSL allowing attackers to read memory of servers, potentially accessing sensitive information.

***Mitigation*:**

- Update to the latest version of OpenSSL.
- Regenerate SSL certificates.
- Reset passwords and session keys.

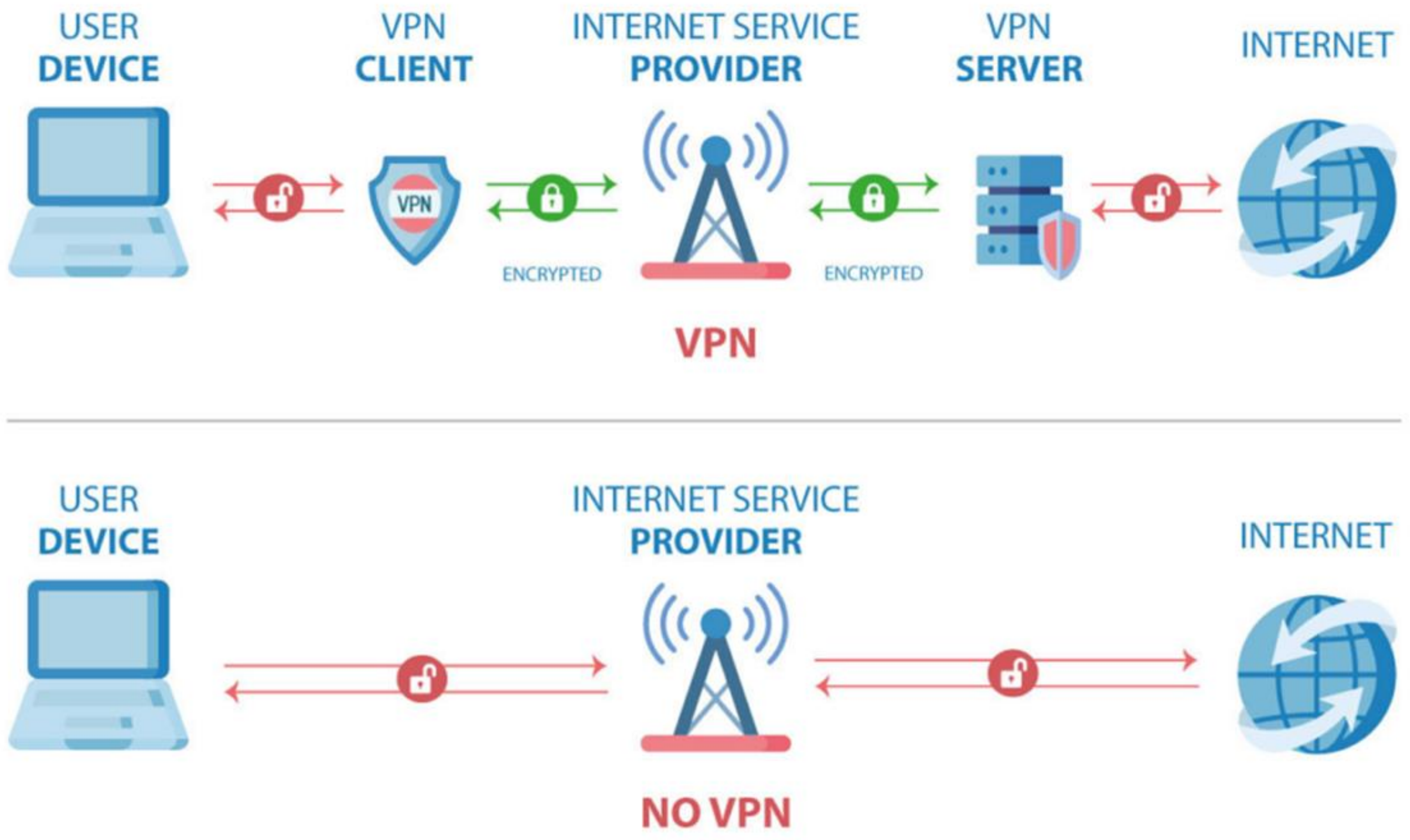
Downgrade attacks

***Description*:** Forces a connection to use older, less secure versions of protocols, making them vulnerable.

***Mitigation*:**

- Use latest versions of TLS.
- Implement TLS_FALLBACK_SCSV to prevent forced downgrades.
- Monitor server logs for repeated connection attempts.

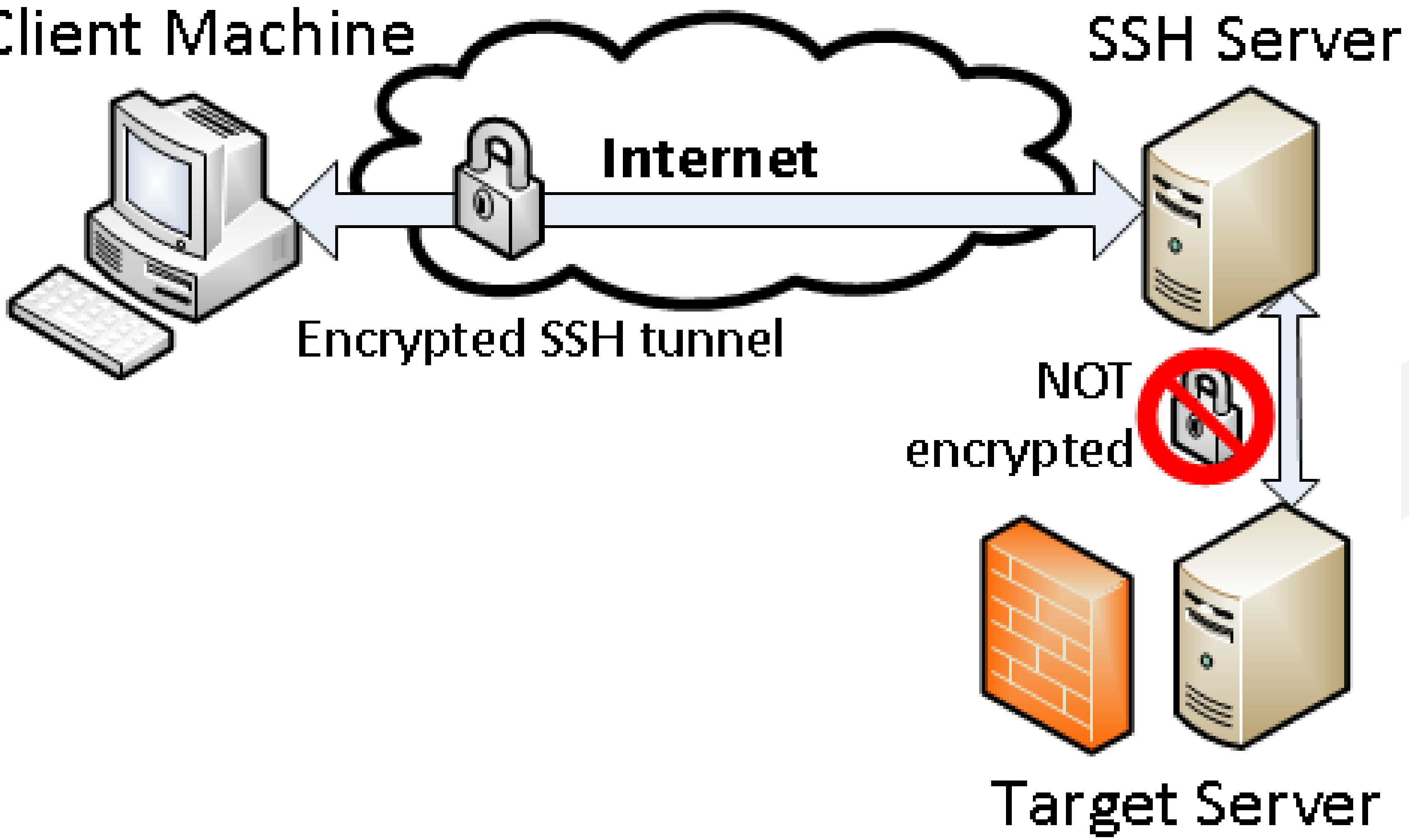
Virtual Protected Networks (VPS)



VPN Protocols (e.g., PPTP, L2TP, OpenVPN, IPsec)

- **What is it?** Protocols used to create encrypted tunnels between devices and servers.
- **Why is it used?** To ensure secure and private communication, especially over untrusted networks like public Wi-Fi.
- **Vulnerabilities:** Weak encryption in older protocols like PPTP, misconfigurations.
- **Biggest Threat:** Unauthorized access due to weak or default credentials.
- **Mitigation:** Use modern protocols like OpenVPN or IPsec, strong authentication methods, and ensure proper configuration.

Client and machine architecture



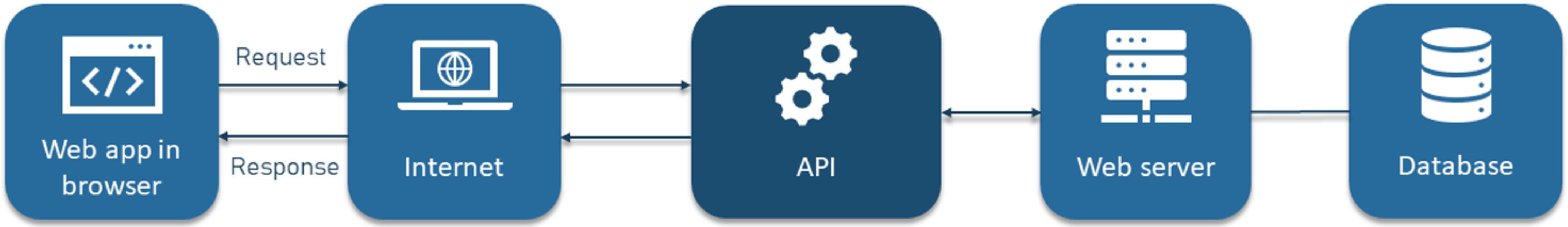
General best practices

•Regularly update software and libraries.

•Use strong, unique certificates and keep private keys secure.

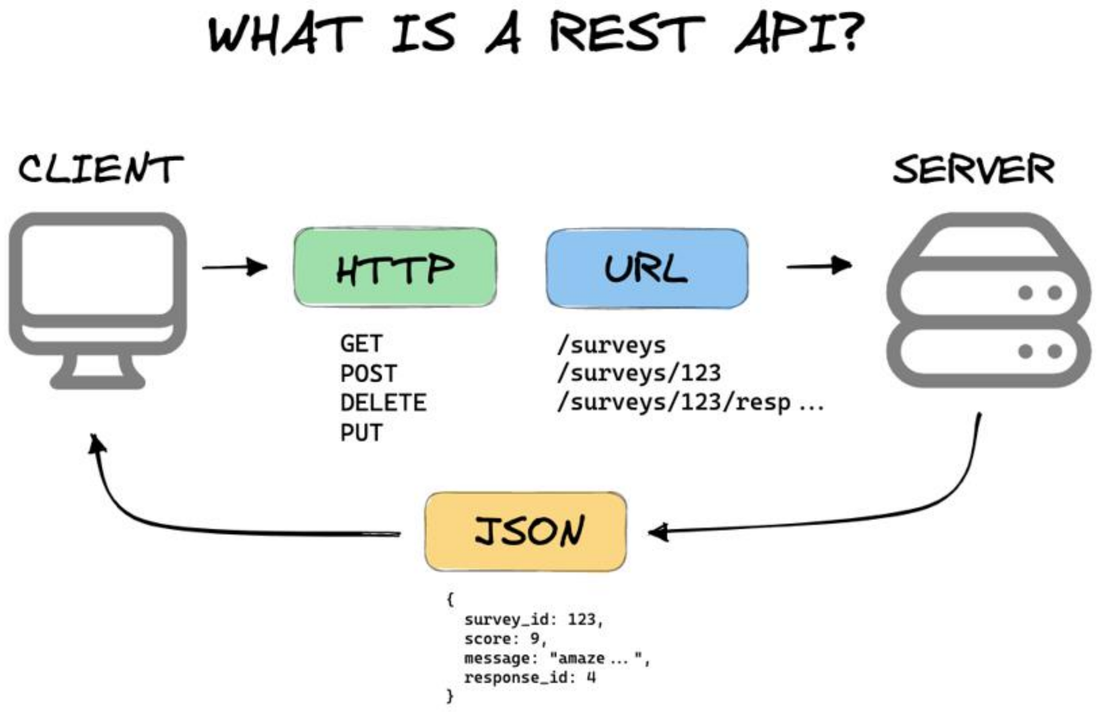
•Monitor for anomalies and suspicious activity.

APIs


Web app in browser
Request
Response
Internet
API
Web server
Database

Activity discussion

What is a Rest API…?



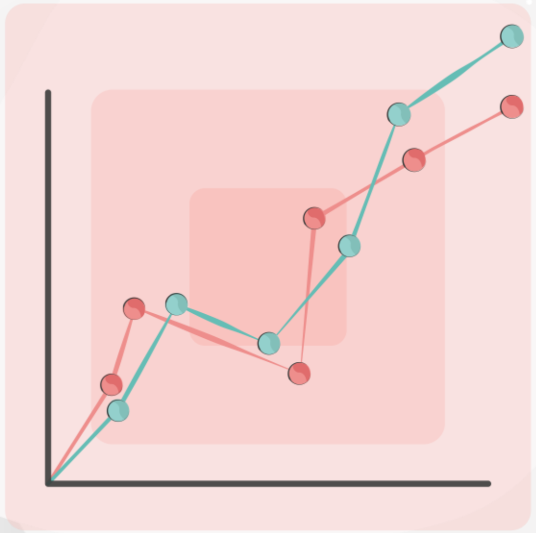
```
{
  survey_id: 123,
  score: 9,
  message: "amaze … ",
  response_id: 4
}
```

Discuss: How would you make an API 'secure by design'?

.

Submit your responses to the chat!

Vulnerabilities



OWASP TOP 5 vulnerabilities

Broken Access Control

- Developers and administrators should follow the Principle of Least Privilege here.

- Access controls should be applied to APIs, and authorization checked for every request.

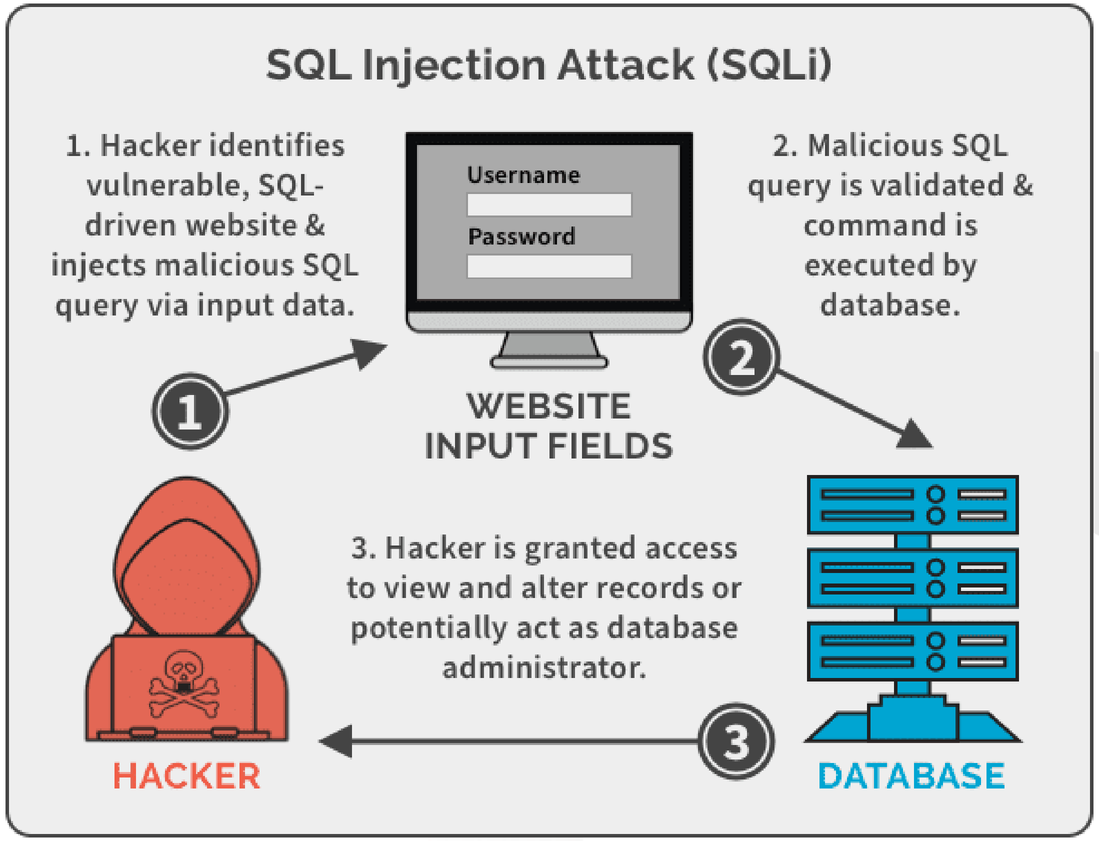
- Regular security audits and code reviews are a must to identify and fix access control issues, and multi-factor authentication should be enforced to limit unauthorised access.

2 - Cryptographic Failures

- Also known as ‘sensitive data exposure’
- Implement regular security testing (including code reviews and vulnerability assessments) to identify and fix cryptographic weaknesses
- Use latest versions of secure cryptographic libraries too.

3 - Injection

SQL but also other types (LDAP, XML, …)


SQL Injection Attack (SQLi)
1. Hacker identifies vulnerable, SQL-driven website & injects malicious SQL query via input data.
Username
Password
WEBSITE INPUT FIELDS
2. Malicious SQL query is validated & command is executed by database.
1
2
3
3. Hacker is granted access to view and alter records or potentially act as database administrator.
HACKER
DATABASE

2 - Insecure Design

- An example of insecure design is an app that produces overly detailed error messages.

- If it reports on error conditions in too much detail and offers diagnostic clues about the application environment, or other associated data, it could be revealing potentially useful information to attackers.

Coursework (discussed next) focuses on secure design.

5 - Security Misconfiguration

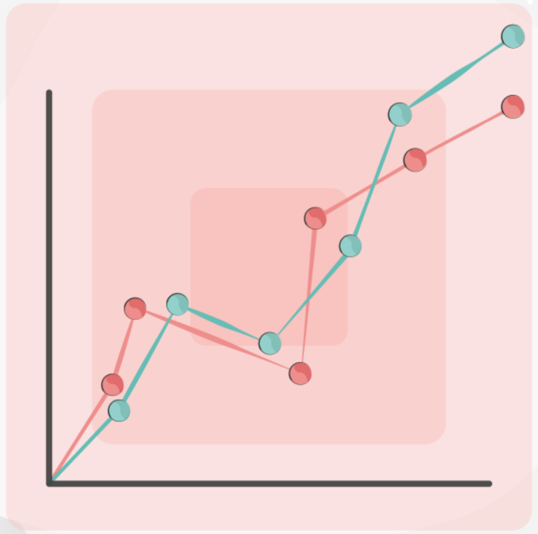
- Unpatched vulnerabilities
- Default configurations
- Unused pages
- Unprotected files and directories
- Unnecessary services
- Use of vulnerable XML files

Activity discussion

Discuss: Which vulnerabilities could be the most critical ones in your line of work?

Submit your responses to the chat!

Diagrams



Design vs architecture

An **architecture** diagram

- Explains what you’re building
- How **stakeholders** will interact with it
- The **constraints** of the system
- Is often drawn as **layers**

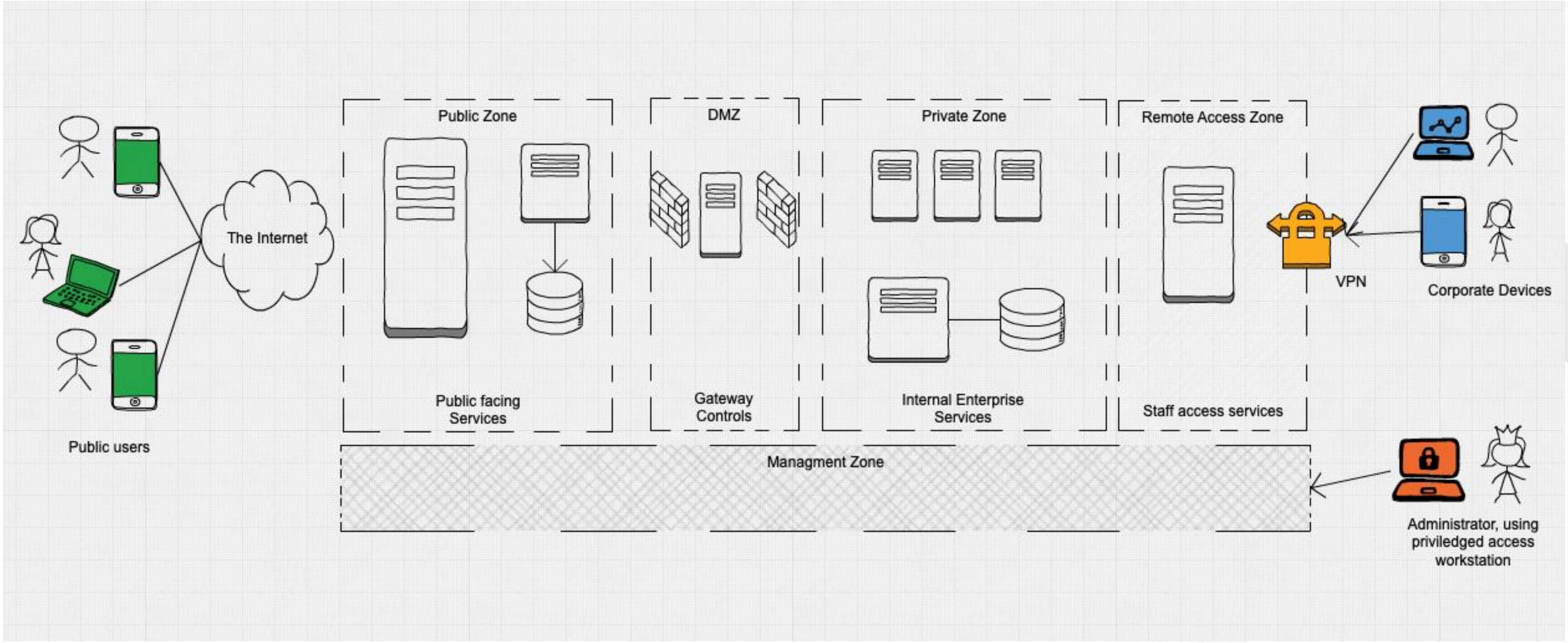
A **design** diagram

- Normally does not have stakeholders, constraints or layers
- Focuses on one part of the system and shows its bulding blocks

A focus on architecture diagrams

Building Careers Through Education

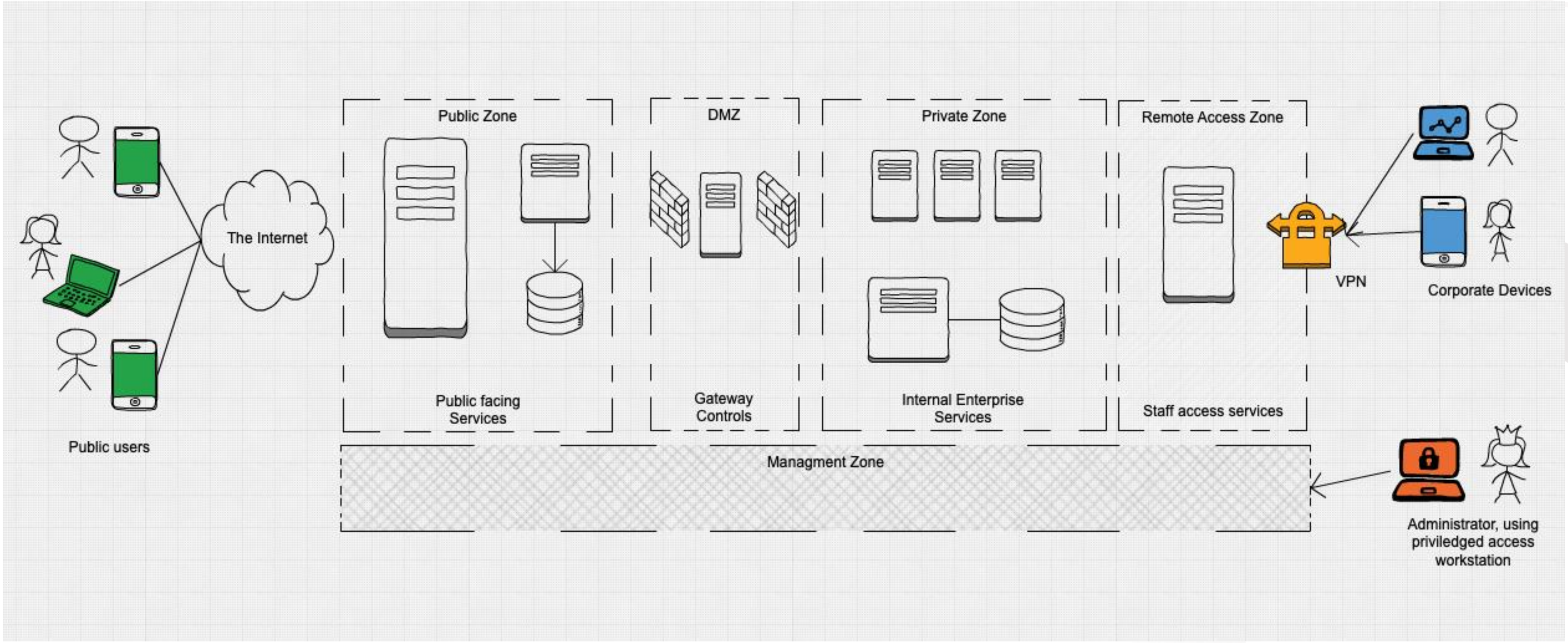
Everybody Matters. Stronger Together. Student, Learner and Client Centric. Trust and Respect. Embrace Change.



From this diagram, you may deduce the following:

- There are a number of different **users** of the system. Public users, corporate users and system administrators. There is also some context as to how these users may interact with the system.
- The system is made up of 5 core **zones**. Public Zone, DMZ, Private Zone, a Remote Access Zone and Management zone.
- We know that there are **functions** that happen in each zone, but the details are not included. These are areas where supplementary diagrams and documentation would help.

Example 2



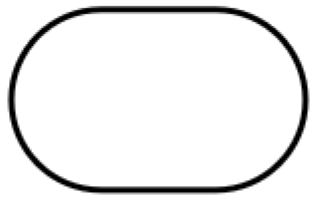
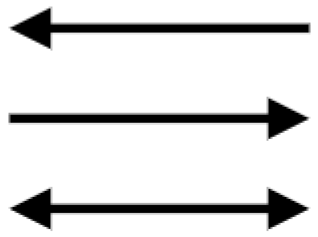
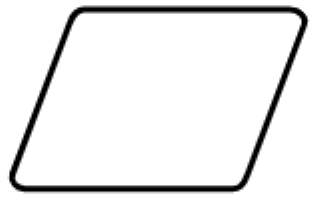
Public Zone
DMZ
Private Zone
Remote Access Zone
The Internet
Public facing Services
Gateway Controls
Internal Enterprise Services
Staff access services
VPN
Corporate Devices
Public users
Managment Zone
Administrator, using priviledged access workstation

Why diagrams

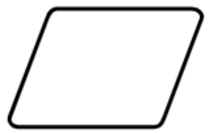
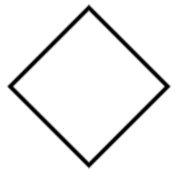
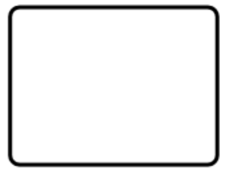
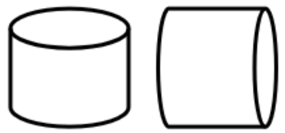
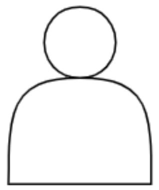
- Diagrams communicate the **most important** aspects of your system
- No diagram is perfect
- You **choose** the most important elements to focus on
- Most diagrams are **starting points** for further discussion
- It's possible to look at the architecture of a system through a number of different lenses. Your 'go to' lens will depend
  - on your job
  - project role
  - area of expertise
  - Stage of project design

Flowcharts

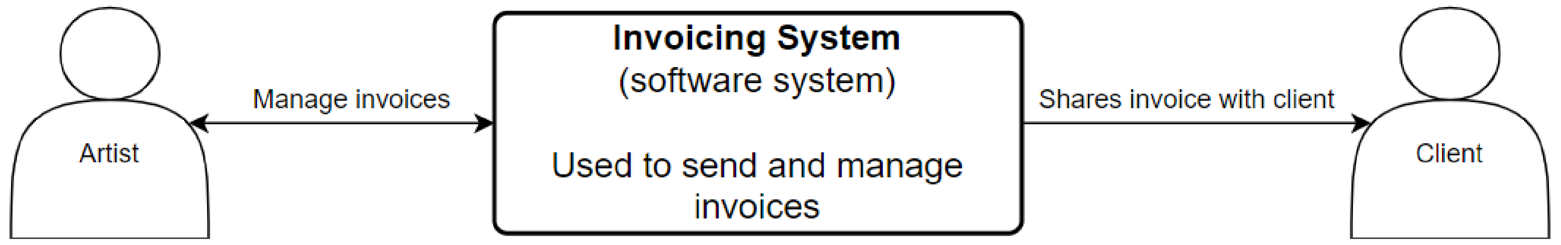
- Flowcharts are one of the most basic types of diagrams you can make

Start / End		Used to represent the starting point or terminal point of a flowchart
Flow lines		Connects components in a flowchart and indicates flow direction
Input / Output		Represents information or data that is transmitted or received

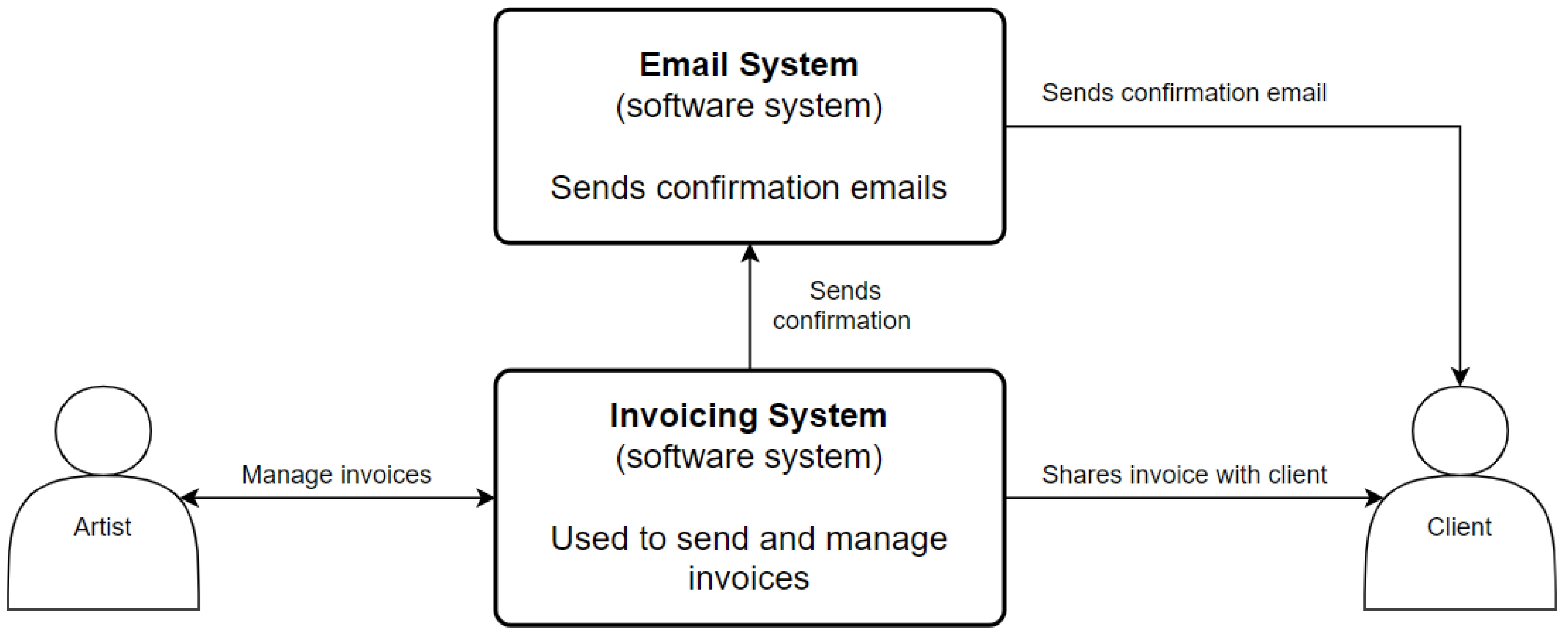
Flowcharts

Input / Output		Represents information or data that is transmitted or received
Decision		Represents checkpoints to evaluate conditions for making decisions
Process		Represents processes (e.g., mathematical operations)
Database		Represents databases
Person		Represents actors or users or a software system

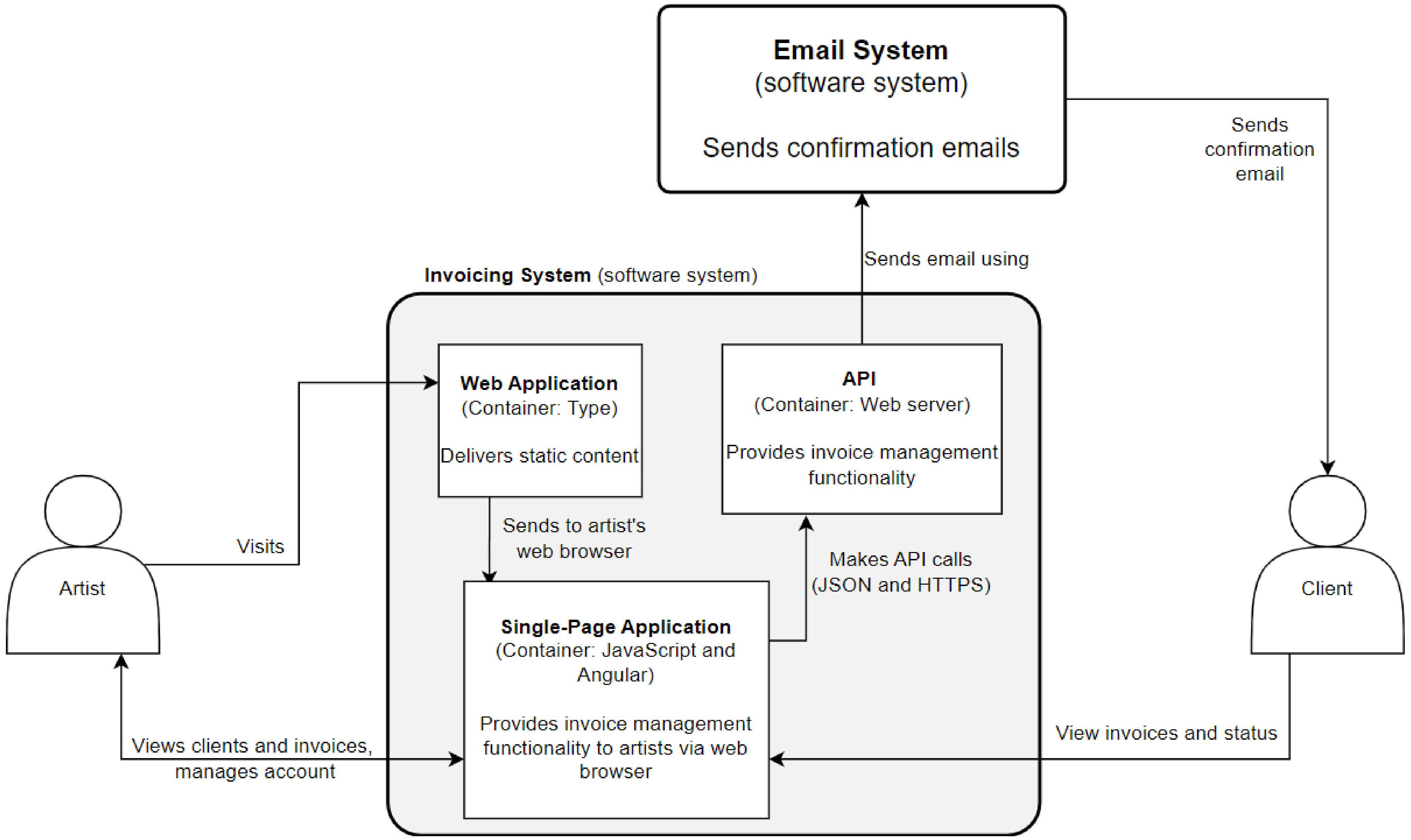
Document the actors



Now that you know who your actors are, document any **external systems** interacting with the software system.

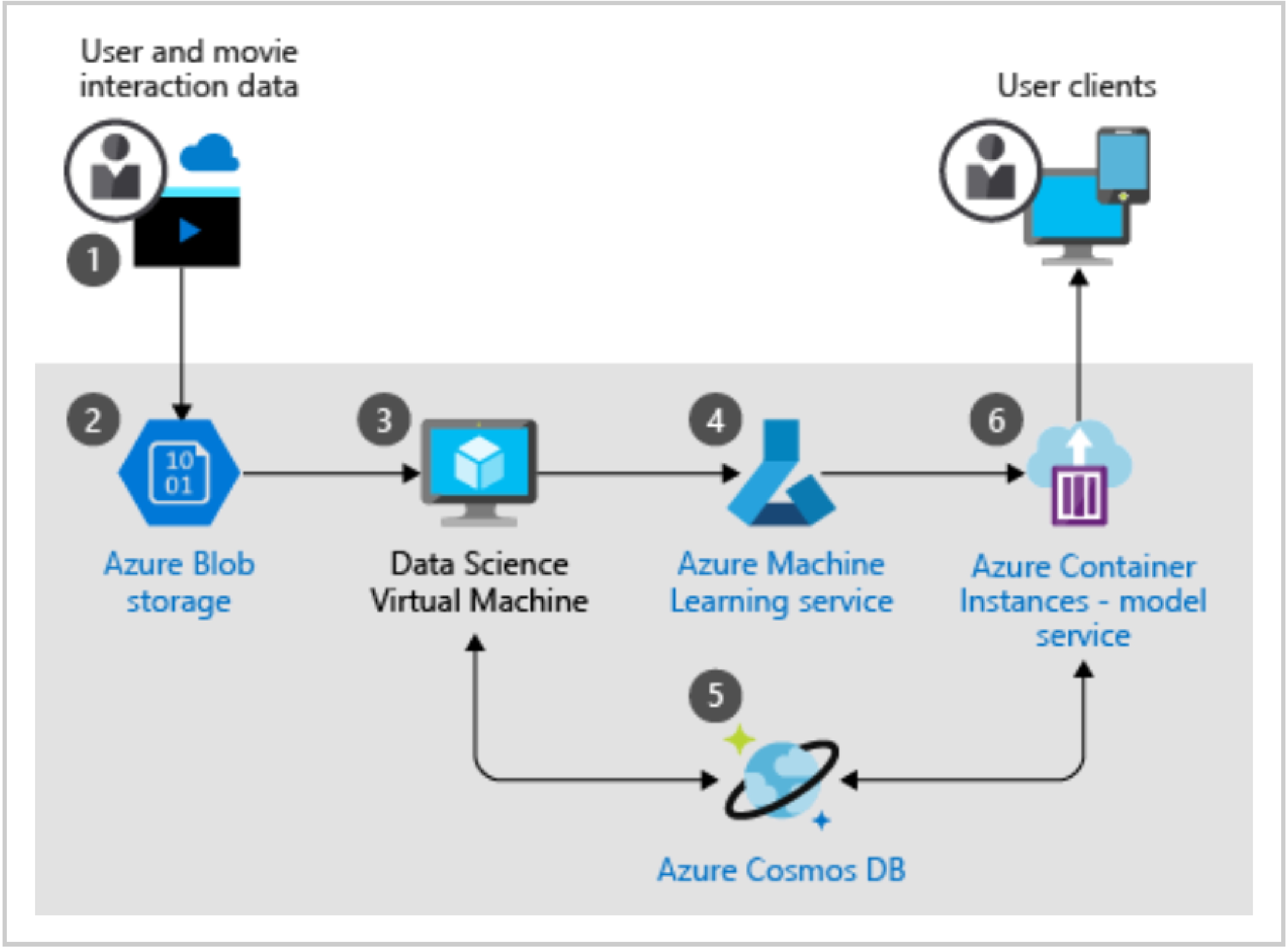


Adding containers



Email System
(software system)
Sends confirmation emails
Sends confirmation email
Sends email using
Invoicing System (software system)
Web Application
(Container: Type)
Delivers static content
API
(Container: Web server)
Provides invoice management functionality
Sends to artist's web browser
Makes API calls (JSON and HTTPS)
Visits
Artist
Client
Single-Page Application
(Container: JavaScript and Angular)
Provides invoice management functionality to artists via web browser
Views clients and invoices, manages account
View invoices and status

Cloud-based architecture diagrams



User and movie interaction data
1
User clients
2
Azure Blob storage
3
Data Science Virtual Machine
4
Azure Machine Learning service
6
Azure Container Instances - model service
5
Azure Cosmos DB

Azure Reference Architecture Diagram

Misuse deployment diagram

(Threat-oriented diagrams)

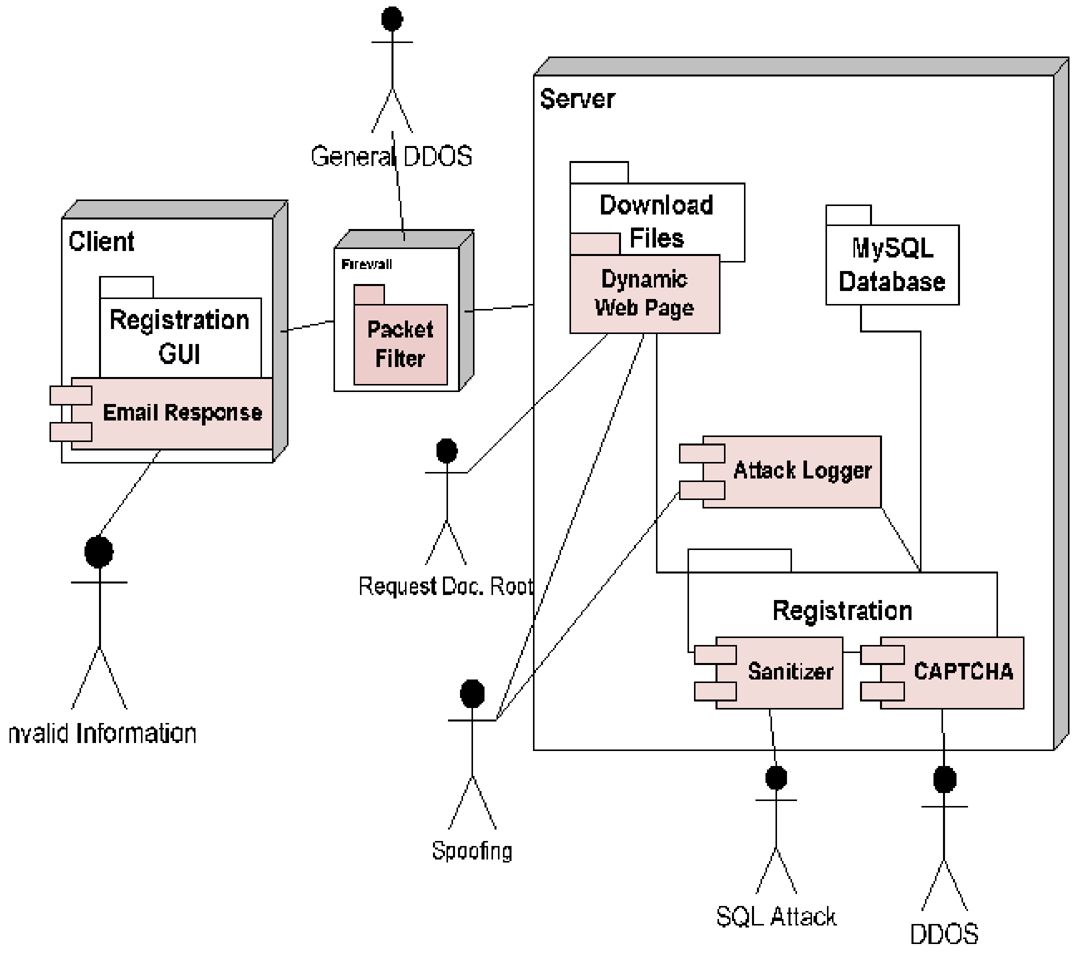
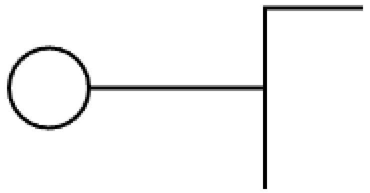


Figure 3. MDD for Web Registration

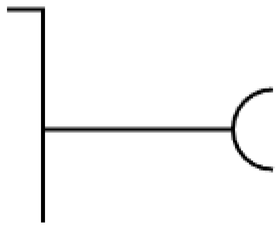
What is a UML Component Diagram

- The purpose of a component diagram is to show the relationship between different components in a system. For the purpose of UML
- 2.0, the term "**component**" refers to a module of classes that represent independent systems or subsystems with the ability to interface with the rest of the system.
- To ideate the system's physical structure.
- To pay attention to the system's components and how they relate.
- Emphasise the service behaviour as it relates to the **interface** (see next slide)

Interfaces

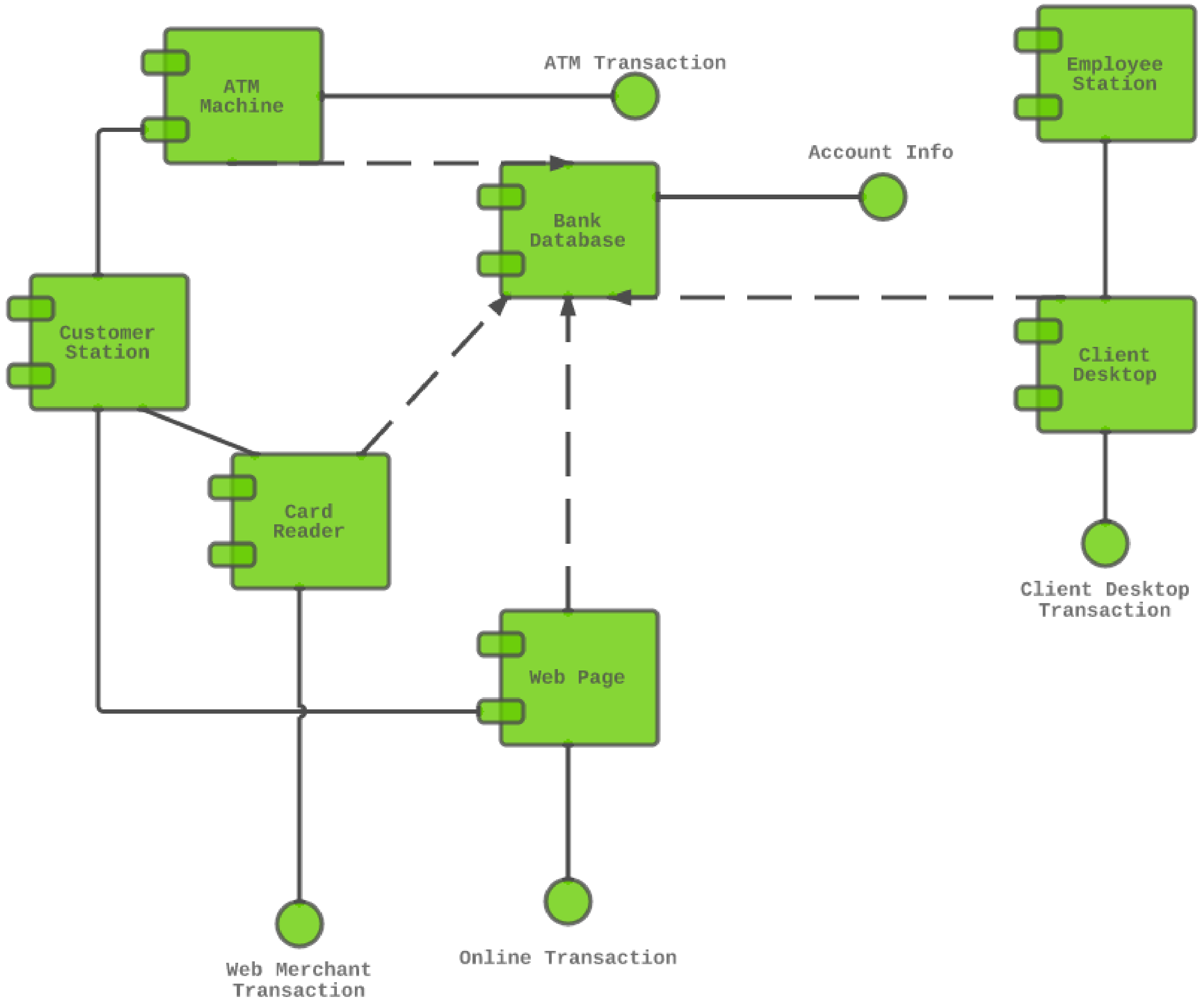


Provided interfaces: A straight line from the component box with an attached circle. These symbols represent the interfaces where a component **produces** information used by the required interface of another component.

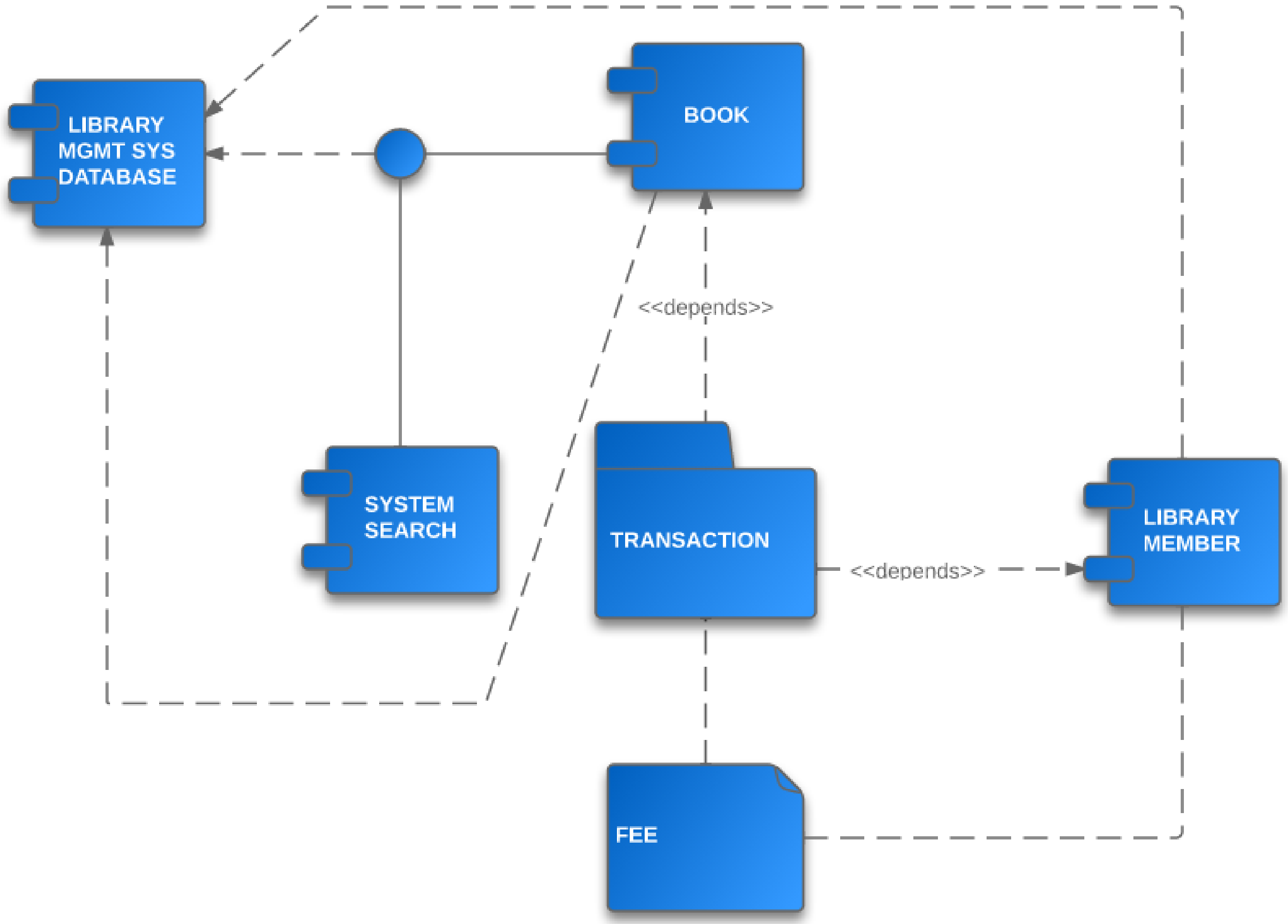


Required interfaces: A straight line from the component box with an attached half circle (also represented as a dashed arrow with an open arrow). These symbols represent the interfaces where a component **requires** information in order to perform its proper function.

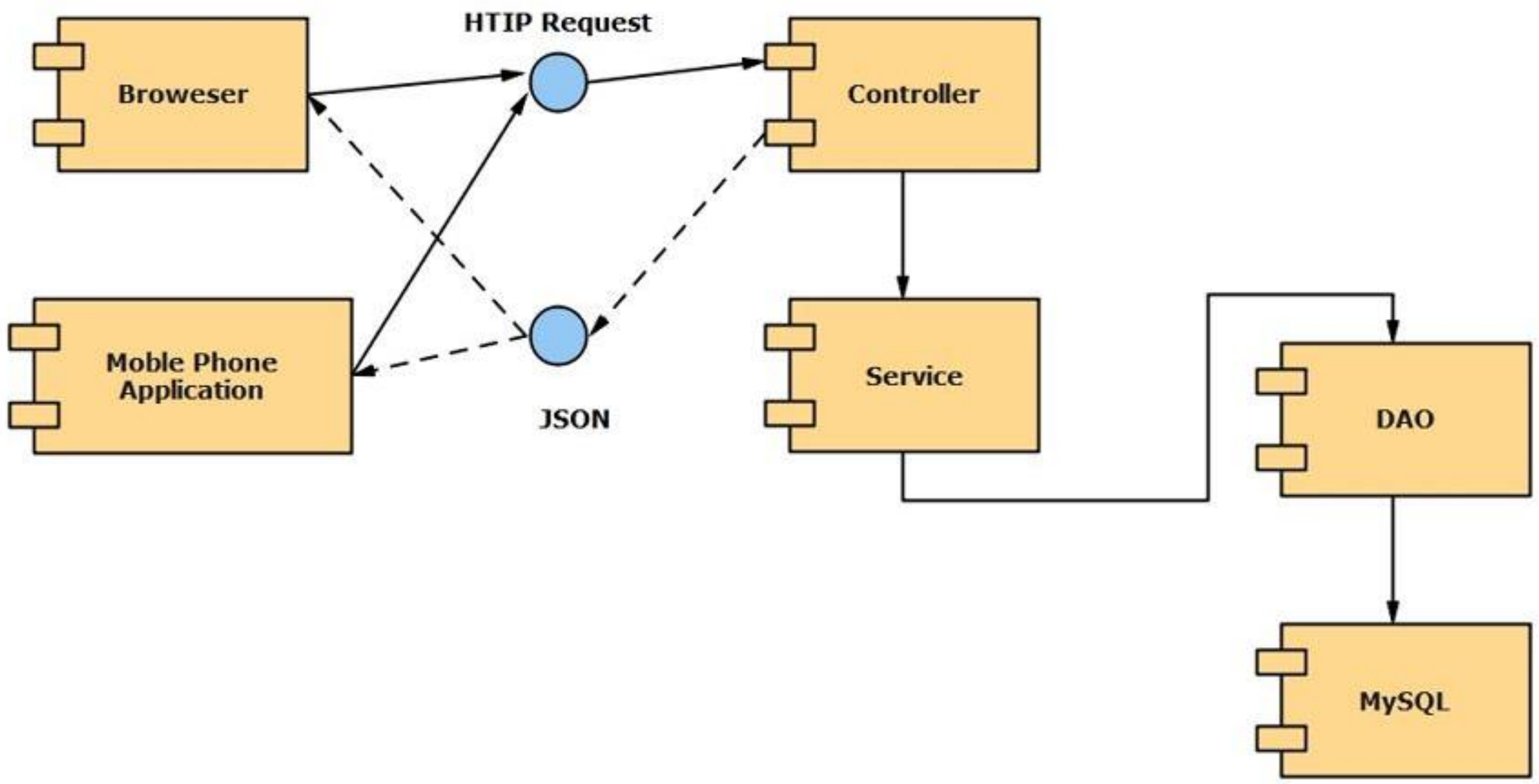
Interface diagram example 1



Interface diagram example 2



System component diagram



How to create diagrams

- LucidChart (online)
- Microsoft Visio (part of MS Office)
- Dia (free software)
- Draw.io (cloud based)
- What do you use?

Building Careers Through Education
Everybody Matters.
Stronger Together.
Student, Learner and Client Centric.
Trust and Respect.
Embrace Change.

Tips for creating architecture diagrams

- Version Control your diagrams – they will evolve
- Simplify when possible, split when necessary
- Create logical groupings using polygons and colours
- Complement diagrams with descriptions
- Avoid too many acronyms
- You will normally have to explain the symbols you use

Applying lenses, layers and chunks

It's possible to look at the architecture of a system through a number of different lenses.

- *A component diagram*
- *A sequence diagram*
- *A threat diagram, etc…*

We also need to think about the different layers of a system. Over time, technology gets built on top of other technology. When you describe a system in a picture, you have to choose which layer to draw.

At one end of the spectrum, we can depict a high level, conceptual representation of a system.

This would help anyone develop a general understanding

At the other end of the spectrum, we could draw out the low-level technical details of an implementation.

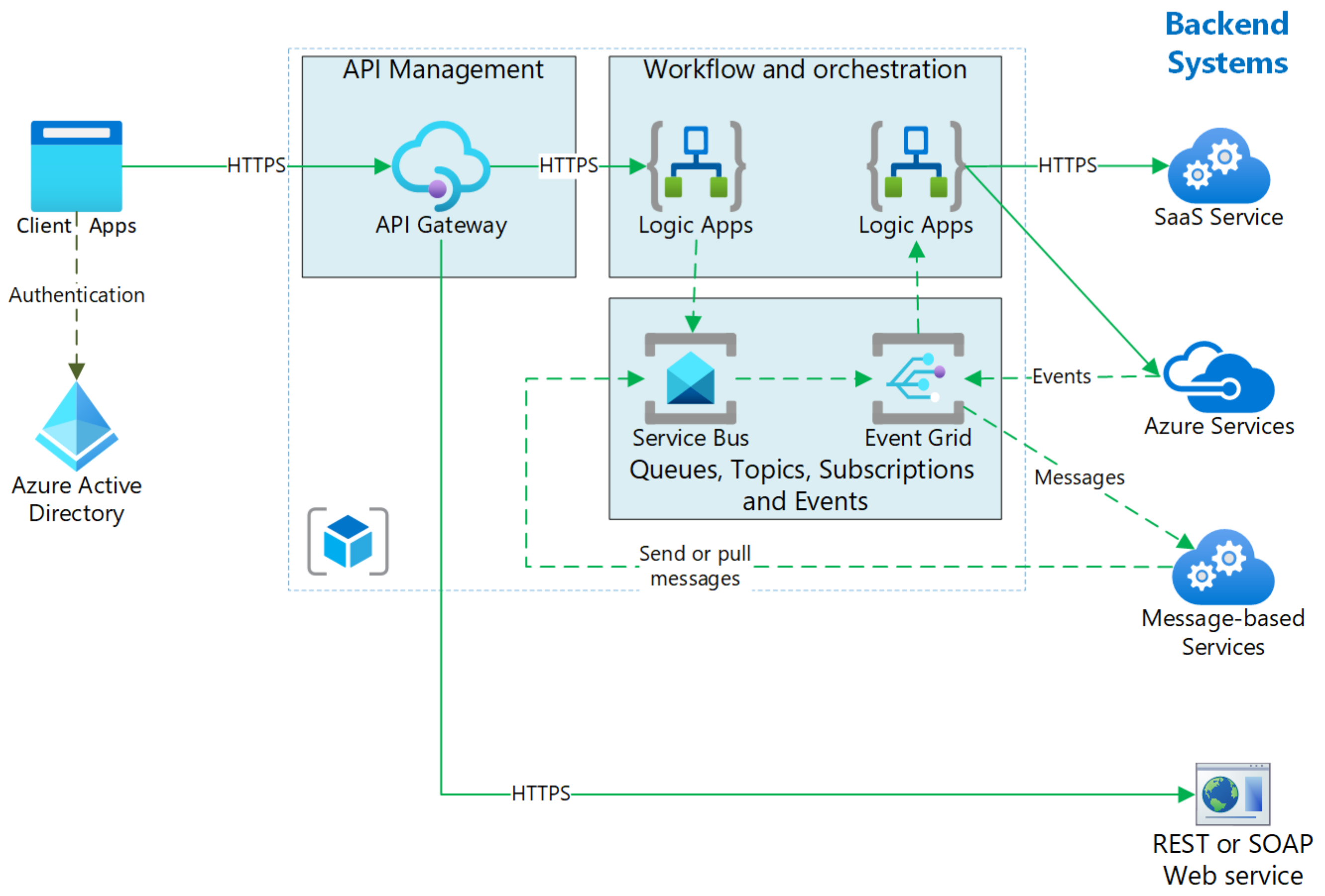
Some other types of diagrams you should research

- Application Architecture Diagram
- Integration Architecture Diagram
- Deployment Architecture Diagram
- DevOps Architecture Diagram
- Data Architecture Diagram

Building Careers Through Education
Everybody Matters.
Stronger Together.
Student, Learner and Client Centric.
Trust and Respect.
Embrace Change.

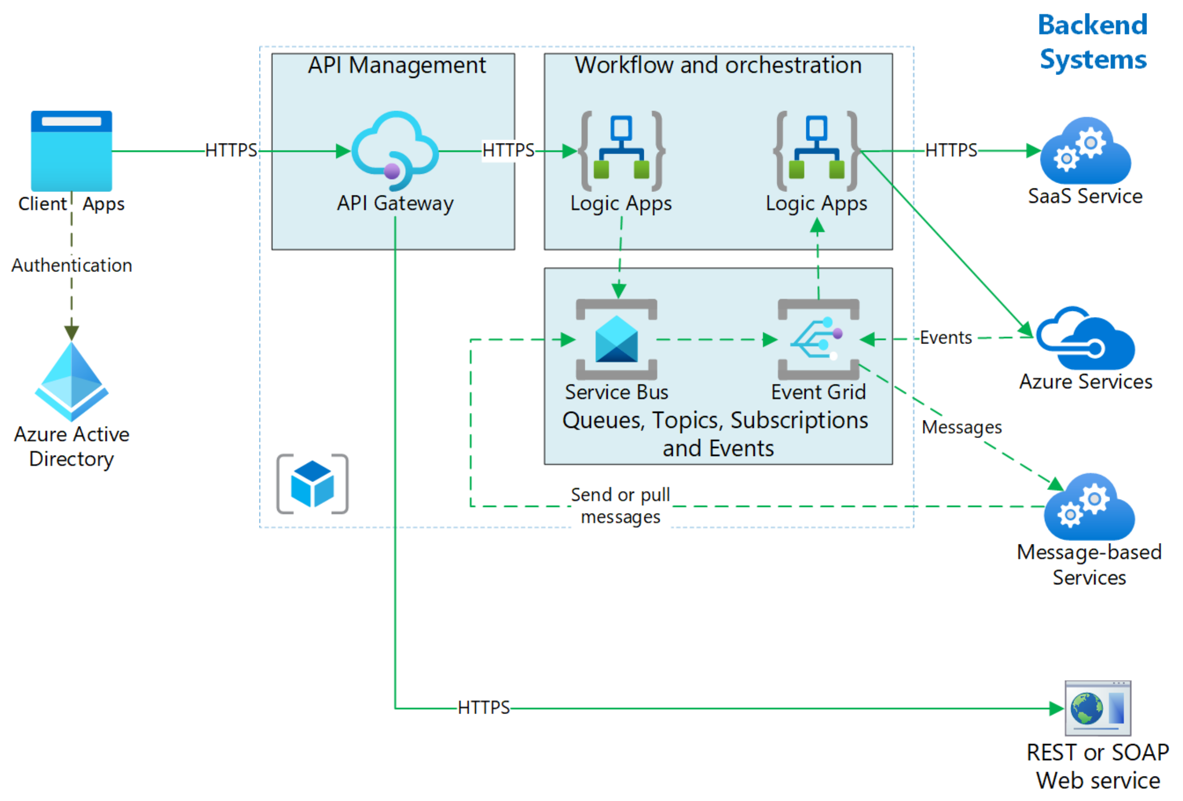
Integration architecture diagram

An integration architecture diagram allows you to visualise the architecture of two applications as they integrate.



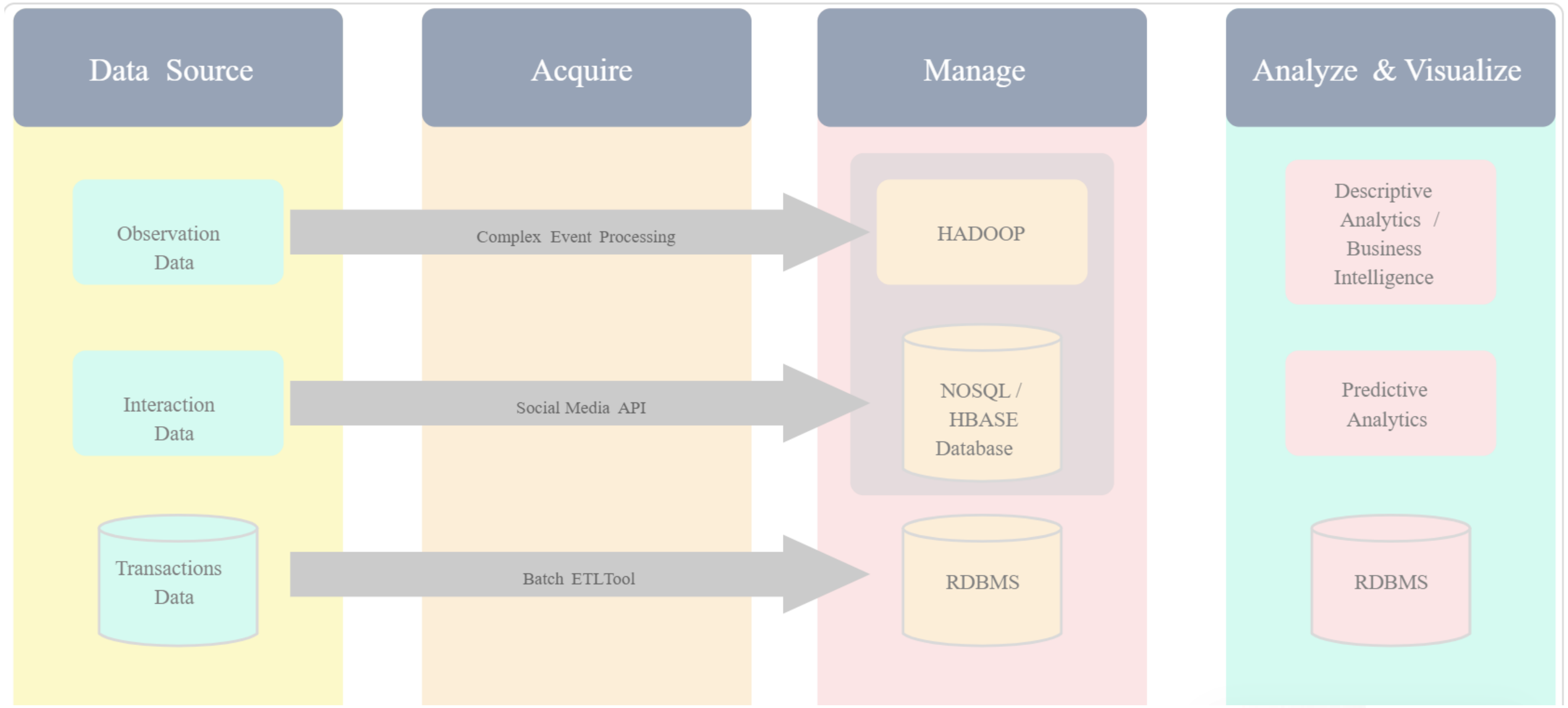
Activity discussion

- **Identify one existing security control in this architecture**
- **Discuss two more security measures to improve its security**



Submit your responses to the chat!

Data architecture diagram – Case study



Building Careers Through Education

Everybody Matters. Stronger Together. Student, Learner and Client Centric. Trust and Respect. Embrace Change.

BPP

Case study: Protecting data at rest and in motion

Data at Rest

•**Encryption with Key Management**:

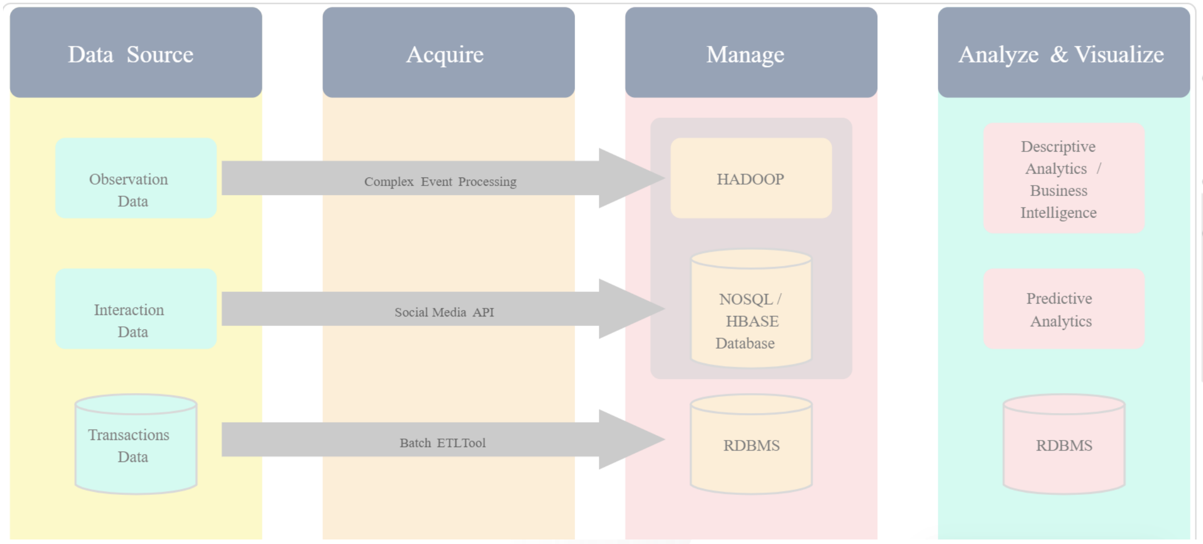
- Use AES-256 encryption for databases
- Use file encryption for Hadoop file system
- integrate a comprehensive key management platform

•**Access Control Mechanisms**:

- Implement role-based access control (RBAC)
- particularly for systems managing PII.

•**Secure Backup Solutions**:

- Encrypt all backups to ensure that data remains secure,



Case study: Protecting data at rest and in motion

Data in motion

Use of Secure Protocols:

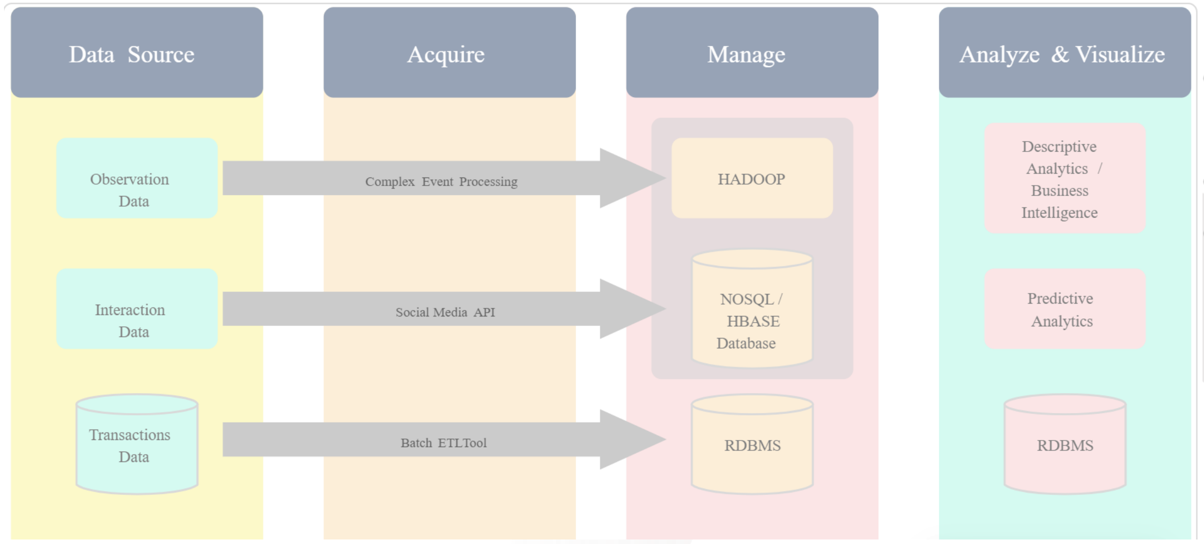
•Enforce TLS for all data transmissions.

Kerberos for Authentication:

Implement Kerberos to provide strong authentication for data exchanges within network environments, ensuring that data transfers are secure and authenticated.

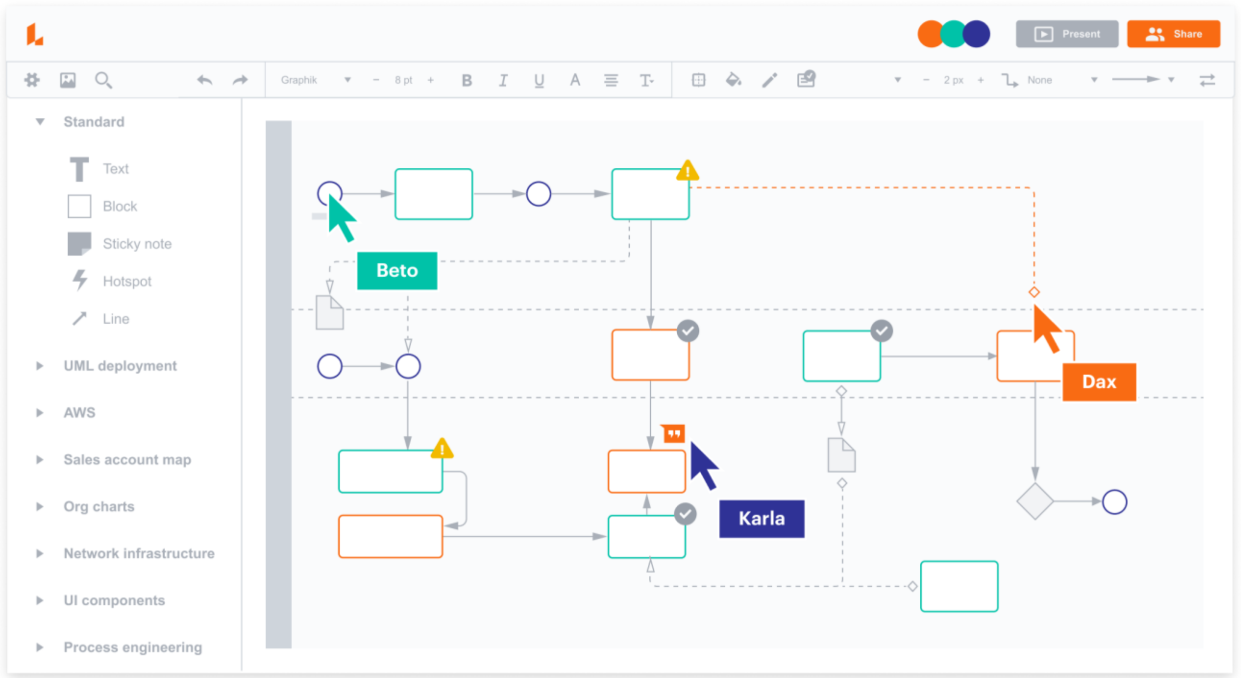
Real-time Threat Detection:

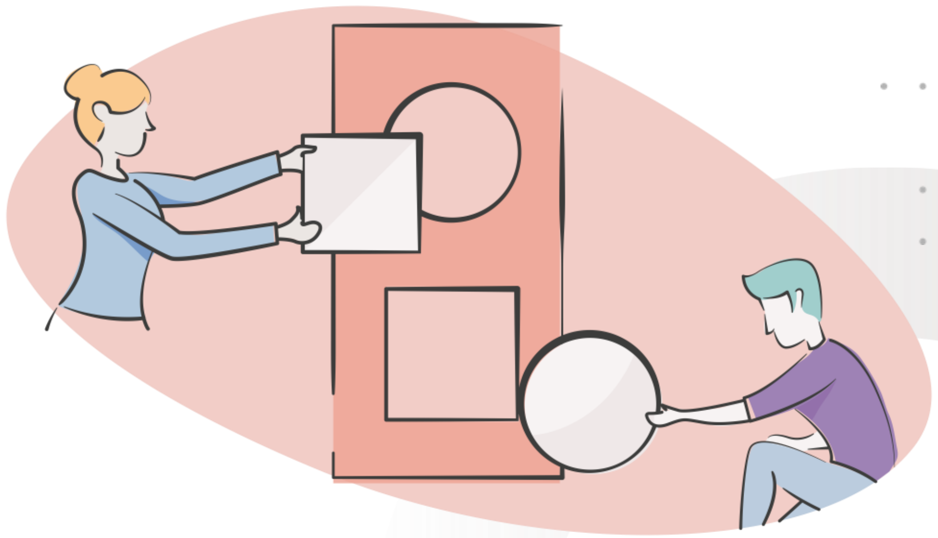
Integrate an Intrusion Detection System (IDS) to monitor and respond to potential security threats against data in transit.



Lab activity

Drawing an architecture diagram using LucidChart





Building Careers Through Education

Everybody Matters. Stronger Together. Student, Learner and Client Centric. Trust and Respect. Embrace Change.

BPP

Session wrap-up

Building Careers Through Education

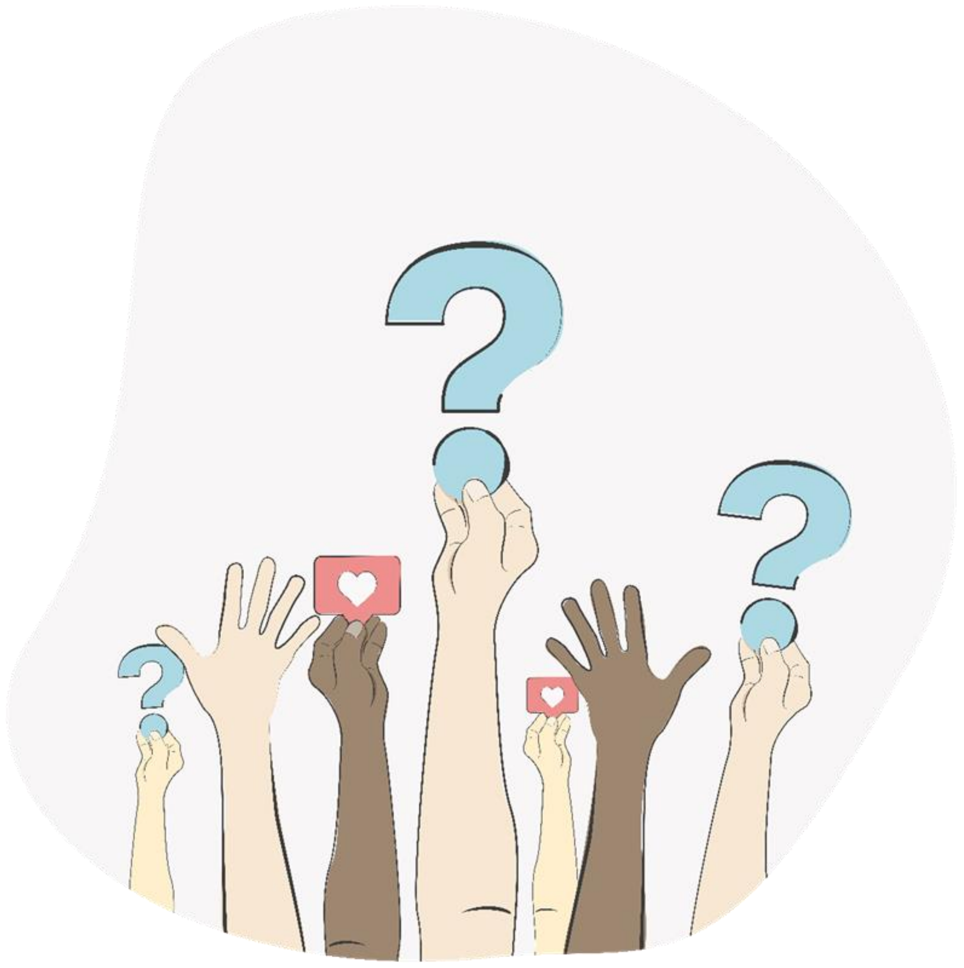
Everybody Matters.

Stronger Together.

Student, Learner and Client Centric.

Trust and Respect.

Embrace Change.



Any questions or feedback?

BPP