Q1)

Although this basic proxy server acts as a basic filter between the user and a web server, it has many significant limitations.

The most noticeable immediately is that it can only be used for http requests rather than https, which majority of modern websites use. This means that SSL/TLS cannot be used with this proxy server.

It is also significantly limited in the way it handles subsequent requests for the same website.

Another limitation is that redirects from one website to another are not handled through the proxy server, but via the browser directly. An issue with this is that if a whitelisted website contains a link to a blocked website, it will still redirect the user and bypass the proxy server.

In its current form, the proxy server is not actually performing any blocking for certain websites. It is simply acting as a passthrough

Q2)

Https could be supported by using X.509 certificates and the python ssl package that supports TLS. This is usually implemented by authenticating the server. Before connecting, the socket is wrapped. The web server then sends a certificate and the client checks the certificate chain to see if it is signed by a trusted CA before sending data. This also adds encryption to any data sent between the proxy server and the web server.

Q3)

The proxy server uses TCP sockets. This is controlled when the socket is created, by setting the socket type to SOCK\_STREAM. This has been confirmed by observing Wireshark output indicating that TCP is used.

DNS is also used to translate the hostname into an IP address. The user provides a URL. From this, the hostname is extracted and passed as a parameter to the Socket.connect() function. Because the hostname is not an IP address, the function must use DNS to find the IP address of the requested host.

Q4)