DESIGN OVERVIEW DOCUMENT

The **proxy server** is an intermediary between a client and a server. It acts as a server to the client and as a client to the server. We have implemented a proxy server with the facility of cache. Our server has two parts mainly, one where it acts as the server and one where it acts as the client.

First it establishes a connection with a client over TCP, and once it accepts a request from the client, it checks if the request is well formed, if not it sends back an error. If the request is well formed then the proxy searches the cache for the required cache and if found, the page is served directly from the cache.

Cache is implemented using the map data structure where we map the URL to a particular hash value which also represents the name of the file in the cache repository. There is a single cache repository which contains all the files named according to their mapped values.

The **cache replacement policy** is implemented in two ways. The first way is to remove the oldest object in the cache. This is done by maintaining a time of caching and to store this a set is used which is sorted according to time. The next way is to remove the largest object in the cache. For this we maintain a set which is sorted according to the size of the files stored in the cache repository. If there is a cache miss the request is forwarded to the origin server from where the HTTP request is sent to the proxy and the same is forwarded to the client.

Also the client can send two **types of requests**, the regular GET request and the conditional GET request. In case of a **regular GET** request normal procedure is followed, but in case of a **conditional GET** request a If-Last-Modified-Since header is included in the request. If there is a cache miss then the proxy removes this header from the request and forwards it to the origin server, whereas if there is a cache hit then the proxy compares this date with the date stored and if the stored date is newer the page is served from the cache else it is requested from the origin server.

Also there is a **client** which takes a URL as input and sends the HTTP request to the proxy. The client first gives an option to the user to choose from a request

and a command line. If the user chooses a **request** it asks the user if he/she wants to send a conditional GET request or a regular request, if the user chooses to send a regular GET request the client asks for the URL whereas for the conditional GET request the client also asks for a Last Modified Date. If the user chooses to enter **command** then the client asks the user to enter on of the following command: **Print Log, Print Cache and Search key.** These requests or the commands are further forwarded to the proxy.

Complexity:

Data Structure	**Operation**	**Complexity per operation (n = current size of data)**
Map	Insert	Log n
	Find	Log n
	Delete	Log n
Set	Insert	Log n
	Find	Log n
	Delete	Log n

So for k elements the order for each of the above operation seperately is (k * Log k).