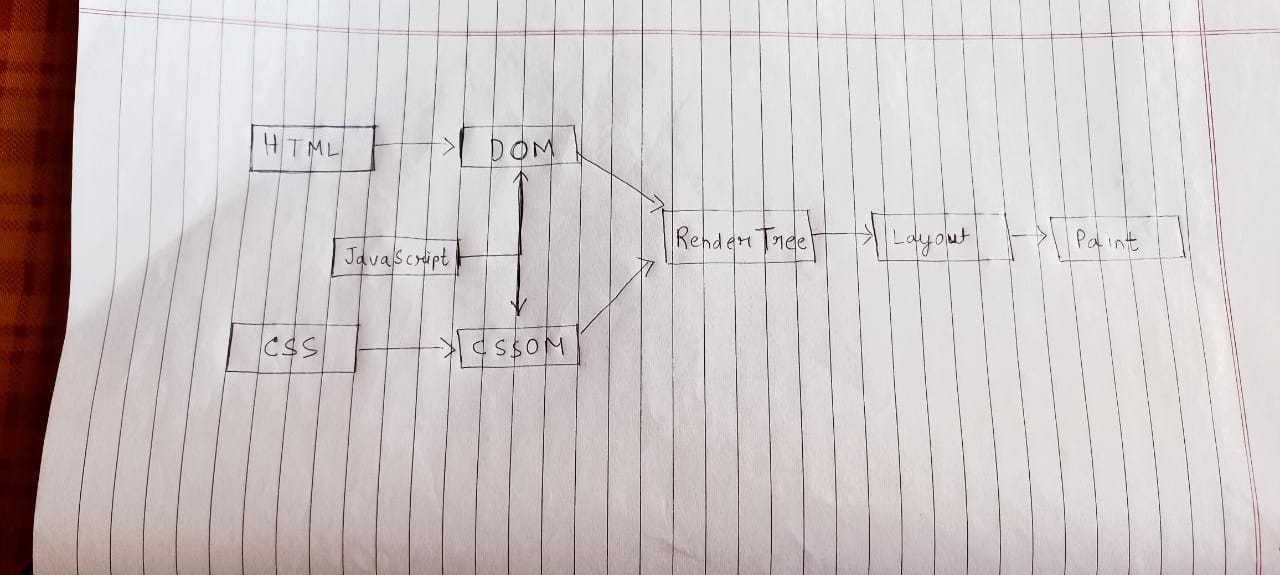
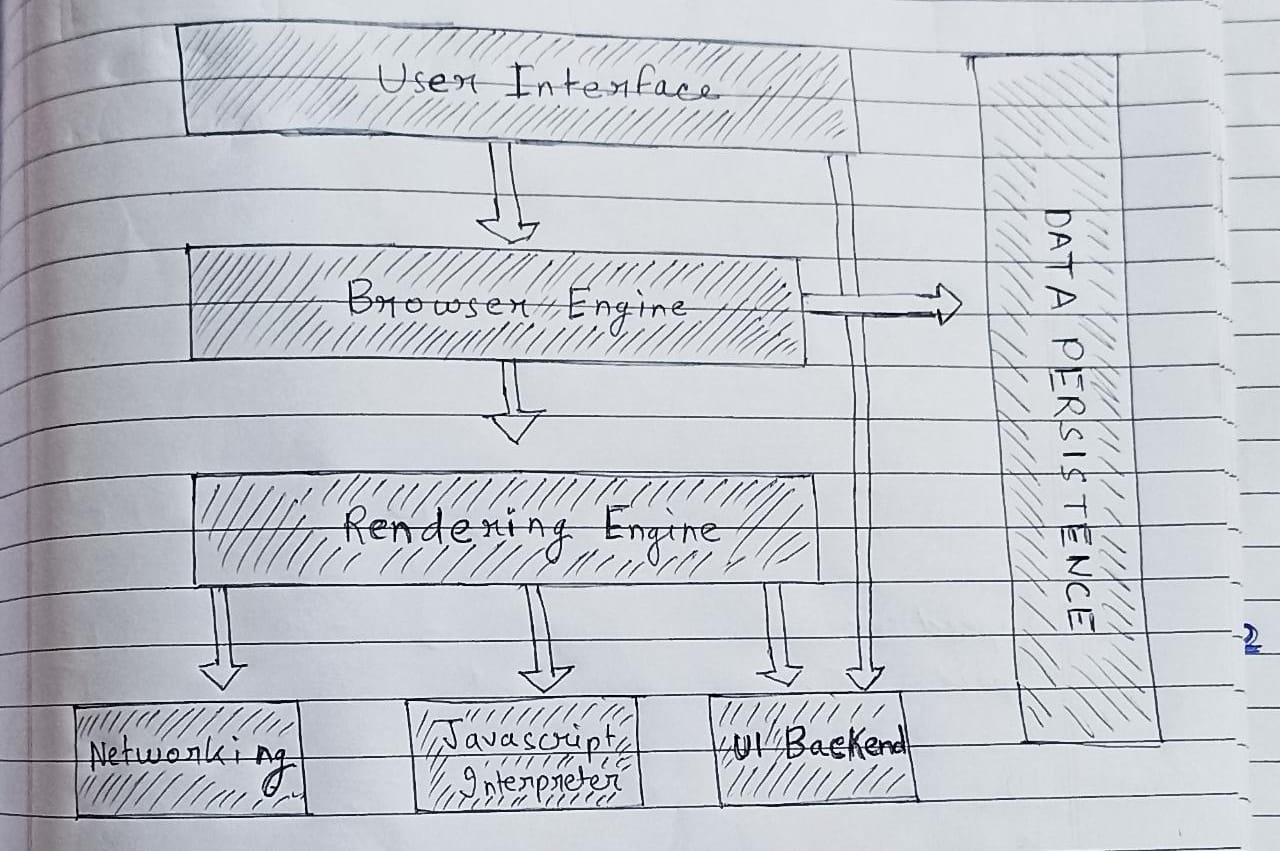
Exercise 1.1

1. These are the following steps how the browser takes a URL and then fetches the desired result

* A URL is provided to the browser
* Browser looks up IP address for the domain.
* Browser initiates TCP connection with the server.
* Browser sends the HTTP request to the server.
* Server processes request and sends back a response.
* Browser renders the content.





a. The Main function of a web Browser is to provide the web Resource a user wants by requesting it from the server, the web Browser is basically an application Software that we use to access the internet, We provide a web address to the browser and the browser then takes to our desired web page depending upon the address, A browser takes us anywhere on the internet , The Web Browser retrieves information from various parts of the web and the user can see it on their desktop or their smart phone, The information is transferred using the Hypertext Transfer Protocol, The Hypertext Transfer Protocol defines how text, images and video are transmitted on the web. We provide a URL with the URL prefix (HTTP / HTTPS), Depending upon the URL Prefix the browser interprets the URL and displays the content to the user.

b. The High-Level Components of a Browser are:

1. User Interface- The User Interface is basically what is presented to the User to interact with, It consists of the address bar, the back and forward button , reload any other visual elements that a user can interact with.
2. Rendering Engine - The Browser has a rendering engine that is responsible for the visual representation of the web page, it is its responsibility to construct the page by applying the right structure and colors, The engine takes in HTML and CSS documents and displays the interpretation of both.
3. Browser Engine- The Browser Engine acts as a Marshal who directs actions between the User Interface and the Rendering Engine, as well as external communication with Servers.
4. Network- This component is responsible for managing network calls using standard protocols like HTTP or FTP. It also looks after security issues associated with internet communication.
5. JavaScript Interpreter- JavaScript provides the necessary behavior to our websites, This allows to interpret the JS code in a document.
6. Data Storage- This is a persistence layer. The browser may need to save all sorts of data locally

such as cookies, Browsers also support storage mechanisms such as local Storage, IndexedDB, WebSQL and File System

1. UI backend- Used for drawing basic widgets like combo boxes and windows. This backend exposes a generic interface that is not platform specific. Underneath it uses operating system user interface methods.

c. The Rendering Engine is a very important component of the web Browser, It is basically responsible for the for the proper visual representation of a web Page, It basically constructs the web page , its structure ,layout and colors , The Rendering Engine takes in HTML and CSS files and then interpret the structure from the HTML and the styles of the various elements from CSS and then displays an amalgamation of both.

d. Parsing means analyzing and converting a program into an internal format that a runtime environment can actually run, The browser parses HTML into a DOM tree. HTML parsing involves tokenization and tree construction. HTML tokens include start and end tags, as well as attribute names and values. If the document is well-formed, parsing it is straightforward and faster. The parser parses tokenized input into the document, building up the document tree. When the HTML parser finds non-blocking resources, such as an image, the browser will request those resources and continue parsing. Parsing can continue when a CSS file is encountered, but <script> tags—particularly those without an async or defer attribute—blocks rendering, and pauses parsing of HTML.

When the browser encounters CSS styles, it parses the text into the CSS Object Model (or CSSOM), a data structure it then uses for styling layouts and painting. The browser then creates a render tree from both these structures to be able to paint the content to the screen. JavaScript is also downloaded, parsed, and then executed. JavaScript parsing is done during compile time or whenever the parser is invoked, such as during a call to a method.

e. Allows inline and stored scripts to be executed within ingest pipelines. The Script Processor leverages caching of compiled scripts for improved performance. The script processor uses the script cache to avoid recompiling the script for each incoming document.

f. The CSSOM and DOM trees are combined into a render tree, Render tree contains only the nodes required to render the page, which is then used to compute the layout of each visible element and serves as an input to the paint process that renders the pixels to screen. Optimizing each of these steps is critical to achieving optimal rendering performance.

g. Scripts are loaded in the order encountered in the page. It does not matter whether it is an external script or an inline script - they are executed in the order they are encountered in the page. Inline scripts that come after external scripts are held until all external scripts that came before them have loaded and run. When a script tag is inserted dynamically, how the execution order behaves will depend upon the browser. A script tag with async may be run as soon as it is loaded. In fact, the browser may pause the parser from whatever else it was doing and run that script. So, it really can run at almost any time. If the script was cached, it might run almost immediately. If the script takes a while to load, it might run after the parser is done. A script tag with defer waits until the entire parser is done and then runs all scripts marked with defer in the order they were encountered. This allows you to mark several scripts that depend upon one another as defer.

h. Layout computes the exact position and size of each object. Finally, now that we know which nodes are visible, and their computed styles and geometry, we can pass this information to the final stage- Painting, which converts each node in the render tree to actual pixels on the screen.

