INTERNSHIP DOCUMENTATION:

By Rohan Jojo

INTRODUCTION:

In this document, I will share my experience as an intern at Overbrook Technologies and Services. Over the course of a month, I had the opportunity to immerse myself in various aspects of the firm, gaining valuable insights and hands-on experience. This document will detail the topics and projects I worked on during my internship, highlighting the skills and knowledge I acquired.

7 Characteristics of a software engineer:

On the first day of my internship, I was assigned to read *Code Complete: A Practical Handbook of Software Construction* by Steve McConnell. The book outlines seven major characteristics that a software engineer should possess. It emphasizes that if one lacks these traits, they should strive to develop and integrate them into their daily life.

The 7 characteristics are:

- Intelligence and Humility
- Curiosity
- Intellectual Honesty
- Communication and Cooperation
- Creativity and Discipline
- Laziness
- Persistence

These characteristics will be further briefed in the book which will be available in the firm. After the first day, I was told to go through the basics which are mentioned down below to become a better web developer.

What is a Web Browser and How Does it Work?

Web Browser: A software that allows users to view and interact with content on the World Wide Web by retrieving web pages, images, videos, and documents from servers to the user’s device. It acts as a bridge between the user and websites.

History: The first web browser, called WorldWideWeb, was invented by Sir Tim Berners-Lee in 1990.

Functions of a Browser:

1. **Web Page Rendering**: Translates HTML, CSS, and JavaScript files into a web page that users can view and interact with.

2. **Navigation**: Allows users to navigate the web using URLs and hyperlinks.
3. **Tabbed Browsing**: Enables users to open multiple web pages simultaneously in a single window.
4. **Bookmarks and History**: Saves web pages for easy revisiting and keeps track of recently accessed pages.
5. **Search Functionality**: Integrated with search engines to help users find web pages.

Components of a Browser:

- **Front-end/User Interface**: The part where users interact with the browser, including the search bar, navigation buttons, and tabs.
- **Back-end**: Handles rendering HTML, CSS, and JavaScript, and manages communication between the browser and web servers.
- **Browser Engine**: Facilitates user interaction, web page rendering, and communication with other components.
- **Rendering Engine**: Renders web pages, ensuring proper display of web files.
- **JavaScript Interpreter**: Executes JavaScript code for dynamic content and interactivity.
- **Networking**: Manages network communication, resolves URLs to IP addresses, and fetches web resources from servers.

How a Web Browser Works:

1. **DNS Resolution**: The browser translates the domain name into an IP address to locate the server where the web page is stored.
2. **Sending Request**: The browser sends a request to the web server using the HTTP/HTTPS protocol to fetch resources for the web page.
3. **Server Response**: The web server responds with the necessary web files (HTML, CSS, JavaScript).
4. **Rendering**: The browser's rendering engine interprets and renders the web files into a web page.
    - **CSS**: Formats the web page visually.
    - **JavaScript**: Adds interactivity and dynamic behaviour to the web page.

Commonly used words or acronyms in browsers:

URL: Universal Resource Locator: addresses a unique resource on the web.

HTML: Hypertext Markup Language: used for creating web applications.

HTTP: Hypertext Transfer Protocol: used for acquiring resources from the web server.

HTTPS: Hypertext Transfer Protocol Secure: used for encryption purposes to secure the connection between browser and web server.

IP Address: It spots the location of a specific sever connected to the internet.

DNS: Doman Name System is a database containing domain records.

Cookies: this is used to store text file information in the user's device so that when a user revisits the website, the device will have the ability to remember the user's preferences and login information.

What is DNS?

DNS (Domain Name System) translates human-readable domain names (like google.com) into IP addresses (like 271.0.0.1) that computers use to identify each other on the internet.

How DNS Works:

1. **User Request**:
    - You enter a domain name in your browser.
    - The browser first checks its own cache to see if it has the corresponding IP address.
2. **OS Cache Check**:
    - If the browser doesn't have it, the OS checks its cache.
3. **ISP Resolver**:
    - If both caches miss, the request goes to the resolver, typically provided by your ISP.
    - The resolver checks its cache for the IP address.
4. **Root Server**:
    - If the resolver doesn't have the IP, it queries a Root Server.
    - Root Servers know where to find Top-Level Domain (TLD) servers (.com, .org, etc.).

5. **TLD Server**:
    - The Root Server directs the resolver to the appropriate TLD server.
    - TLD servers are coordinated by ICANN and contain information on domain extensions.
6. **Authoritative Name Server**:
    - If the TLD server doesn’t have the IP, it provides the resolver with the authoritative name server for the domain.
    - These servers are provided by domain registrars when a domain is purchased.
7. **Retrieving the IP**:
    - The authoritative name server returns the IP address to the resolver.
    - The resolver caches the IP and passes it back to the OS and browser, which also cache it.
8. **Web Server Request**:
    - The browser now has the IP address and sends a request to the web server for the web files needed to render the webpage.

DNS Types:

- **Normal DNS Lookup**: Domain → IP address
- **Reverse DNS Lookup**: IP address → Domain

TLD Types:

1. **Country Code TLDs**: .jp, .fr, .in
2. **Internationalized Country Code TLDs**: In scripts like Russian, Chinese, Arabic
3. **Generic TLDs**: .net, .org, .edu
4. **Infrastructure TLDs**: .arpa for reverse DNS lookups
5. **New Generic TLDs**: .hot, .pizza, .app, .claims

Additional Details:

- **Glue Records**: Used to prevent loops by providing the IP addresses of authoritative servers directly to the resolver.
- **Multiple Authoritative Name Servers**: To distribute load and ensure reliability if one server fails.

How does a TCP/IP work?

After DNS is resolved from a domain, there should be a connection established between the IP address of the server and user's device. To make sure it has been properly connected a three-way handshake is performed with the help of TCP/IP.

First, the client or the web browser will send the destination server a Synchronize Sequence Number(SYN).

Second, the server will send a SYN-ACK message to the web browser to acknowledge that it has received the SYN message.

Third, the web browser acquires this SYN-ACK message and then send another ACK to the server to finalize the connection.

Once this connection is established, only then the web browser would request the access for the web files from the web server.

BASIC CONCEPTS OF DBMS & HOW TO DESIGN A DATABASE:

The following concepts being written down here are based on the Tomy University Student Database Design Document with **reference to An Introduction to Database Systems, Addison-Wesley Publishing Company** from page 296,300, 302 & 306.

We will follow a step-by-step procedure in how to design a database document wrt Tomy University Student Database Design Document:

1. Interview with Tomy University.
2. Extraction of semantic rules.
3. Translation of semantic rules into Functional Dependencies.
4. Creation and drawing of initial Functional Dependencies.
5. Normalization of the reduced initial Functional Dependencies diagram. (From 1NF to BCNF).

Step 1: INTERVIEW WITH THE ENTERPRISE:

One of the first steps to do before jumping into designing into a database the designer must sit alongside with a representative of the enterprise and discuss on how the enterprise operates and how it can be related to a database. Based on that a semantic rule can be designed.

In the case of the Tomy University Student Database Design Document, they acquired the semantic rules directly from their own instructor.

Step 2: SEMANTIC RULES:

What are semantic rules?

In simple terms semantic rules are rules that define how necessary datum are related. Example in the case of the Tomy University Student Database Design, the semantic rules are:

1) Each student has a social security number, a name, a major, a type of loan, a minor, an address, a telephone number and a guardian.
2) Each loan has a sponsor and an amount.
3) Each sponsor has a name, telephone number and a city for location.
4) Each student has a grade for a given course.
5) Each course has a name, a description, a textbook and a course number.
6) Each textbook has a name, an author and a publisher.
7) Each publisher has a name, address and a telephone number,
8) Each student has one instructor and one teaching assistant for a given course.
9) Each teaching assistant is involved in only one course.
10) Each instructor has a name, address, title, and a final degree.
11) Each final degree for an instructor has a granting university.
12) Each granting university has a name, a telephone number and an address.

From the above semantic rules, we can understand that the rules should be concise and to the point. Therefore, the designer and the representative should clearly understand one another and verify each other.

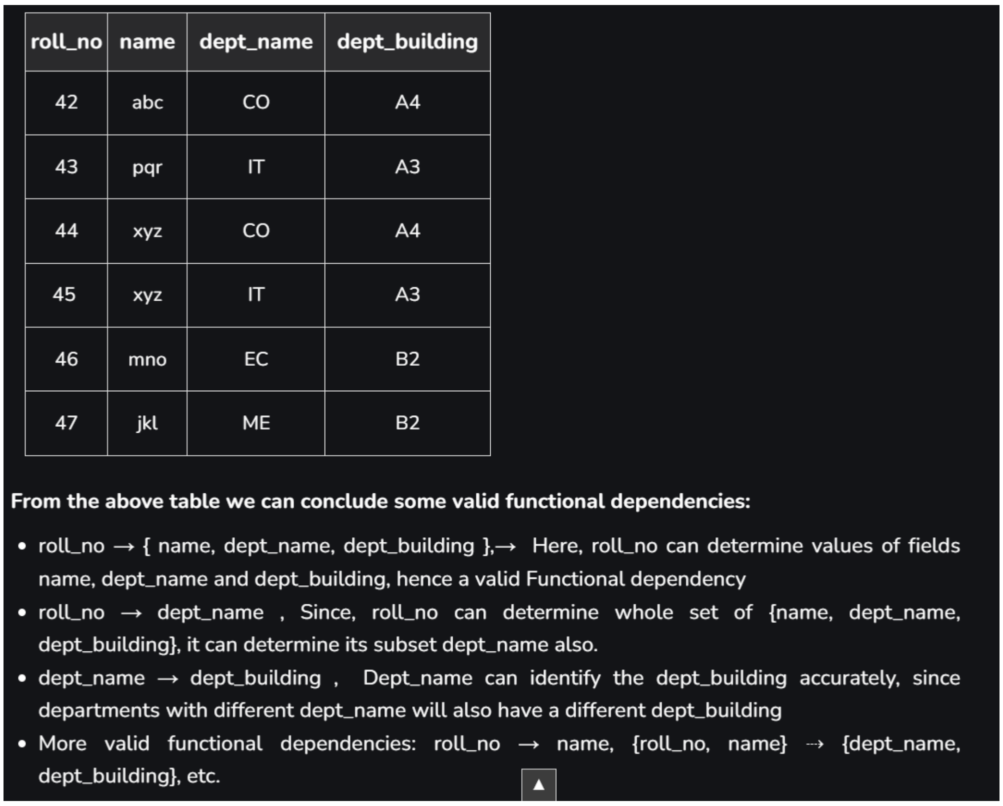
Step 3: TRANSLATION OF SEMANTIC RULES INTO FDs:

What are Functional dependencies?

Functional dependencies can be defined as follows: Consider a relation R, such that an attribute Y → R is functionally dependent on attribute X → R if and only if the 2 tuples of R agree on the values of X and Y.

In simple terms, when two attributes X and Y are said to be functional dependent when values of X can define the values of Y such that Y will be dependent on X (i.e. X→Y). In this case X is the determinant and Y is the dependent. Like primary keys and candidate keys.

Eg)



roll_no	name	dept_name	dept_building
42	abc	CO	A4
43	pqr	IT	A3
44	xyz	CO	A4
45	xyz	IT	A3
46	mno	EC	B2
47	jkl	ME	B2

From the above table we can conclude some valid functional dependencies:

- roll_no → { name, dept_name, dept_building },→ Here, roll_no can determine values of fields name, dept_name and dept_building, hence a valid Functional dependency
- roll_no → dept_name , Since, roll_no can determine whole set of {name, dept_name, dept_building}, it can determine its subset dept_name also.
- dept_name → dept_building , Dept_name can identify the dept_building accurately, since departments with different dept_name will also have a different dept_building
- More valid functional dependencies: roll_no → name, {roll_no, name} → {dept_name, dept_building}, etc.

Let's now translate some semantic rules into FDs, before that lets split each of them into part A and B. A showcases how the attributes are arranged, and B showcases the FD.

1) Each student has a social security number, a name, a major, a type of loan, a minor, an address, a telephone number and a guardian.

A: ID# : Student ID

SSN: Student Social Security Number

Sname: Student's Name

Major: Student's Major

LoanID: Loan

Minor: Student's Minor

Sadd: Student's Address

Stele#: Student's Phone no.

Guard: Student's Guardian

B: ID# → SSN, Sname, Major, LoanID, Minor, Sadd, Stele#, Guard

2) Each loan has a sponsor and an amount.

A: SponsorID: Loan Sponsor ID

Amt: Loan Amount

B: LoanID → SponsorID, Amt

3) Each sponsor has a name, telephone number and a city for location.

A: Spname: Sponsor Name

Sptele#: Sponsor’s Phone no.

Spcity: Sponsor’s City

B: SponsorID → Spname, Sptele, Spcity

4) Each student has a grade for a given course.

A: Course#: Course Number

Grade: Student’s Grade

B: ID#, Course# → Grade

5) Each course has a name, a description, a textbook and a course number.

A: Cname: Course Name

Cdesc: Course Description

BookID: Textbook ID

B: Course# → Cname, Cdesc, BookID

6) Each textbook has a name, an author and a publisher.

A: Bname: Book Name

Author: Author’s Name

PubID: Publishers ID

B: BookID → Bname, Author, PubID

7) Each publisher has a name, address and a telephone number.

A: Pname: Publisher’s Name

Padd: Publisher's Address

Ptele#: Publisher's Phone No.

B: PubID → Pname, Padd, Ptele#

8) Each student has one instructor and one teaching assistant for a given course.

A: InstructorID: Instructor’s ID

TA: Teaching Assistant

B: Course#, ID# → InstructorID, TA

9) Each teaching assistant is involved in only one course.

A: no new definitions

B: TA → Course#

10) Each instructor has a name, address, title, and a final degree.

A: Iname: Instructor’s Name

Iadd: Instructor's Address

Ititle: Instructor's Title

Idegree: Instructor's Final Degree

B: InstructorID → Iname, Iadd, Ititle, Idegree

11) Each final degree for an instructor has a granting university.

A: GunivID: Instructor's Final Univ. Degree ID

B: InstructorID, Idegree → GunivID

12) Each granting university has a name, a telephone number and an address.

A: Gname: Granting university Name

Gtele#: Granting university phone no.

Gadd: Granting university Address

B: GunivID → Gname, Gtele#, Gadd

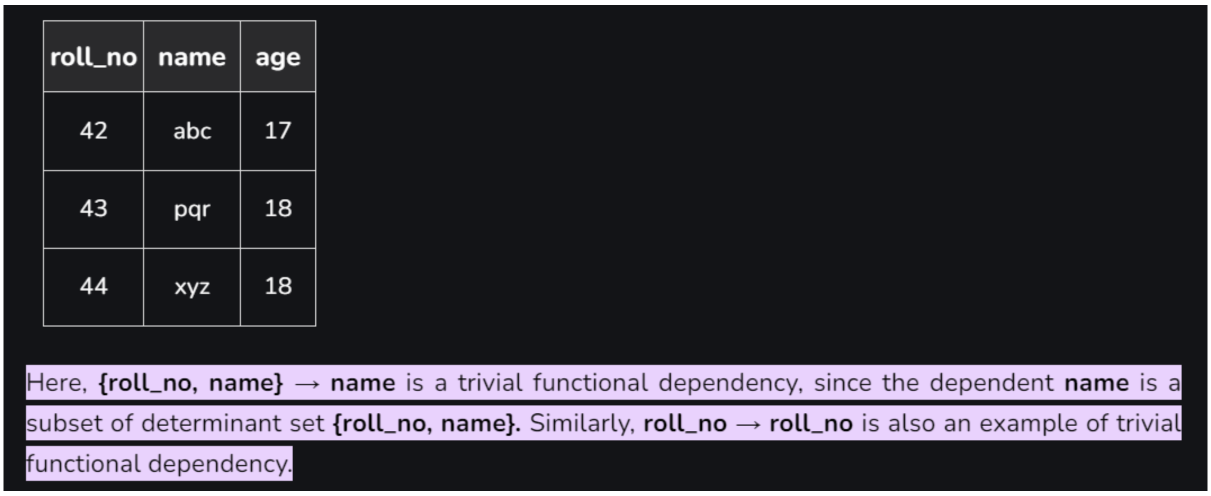
The diagrams for step 4 to step 6 will be mentioned in the paper sheets given by the firm.

Step 4: CREATION AND DRAWING OF INITIAL FD DIAGRAM:

In this step a diagram is created based on the translation of semantic rules to FD and from that we simplify the Database design by Reduction and Normalization.

Step 5: REDUCTION OF THE INITIAL FD:

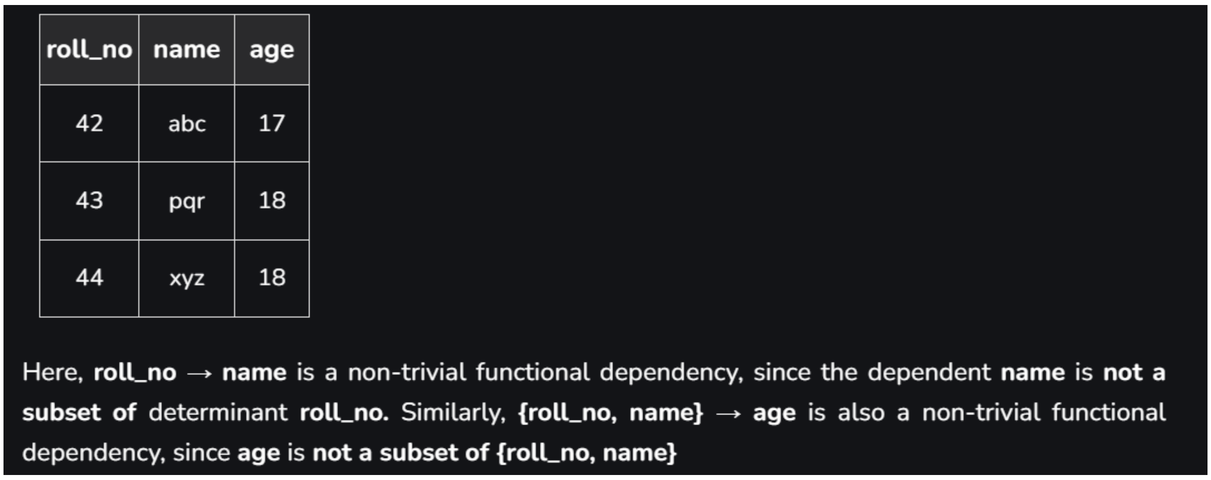
To reduce the FD diagram means that we should remove all the trivial dependencies. Trivial Dependencies are those types of dependencies in which the dependent is a subset of the determinants. Eg) X→Y The attribute values of Y are also present in the attribute values of X. Therefore, Y is a subset of X.



roll_no	name	age
42	abc	17
43	pqr	18
44	xyz	18

Here, **{roll_no, name}** → **name** is a trivial functional dependency, since the dependent **name** is a subset of determinant set **{roll_no, name}**. Similarly, **roll_no** → **roll_no** is also an example of trivial functional dependency.

The FD diagram should be non-trivial for eg)



roll_no	name	age
42	abc	17
43	pqr	18
44	xyz	18

Here, **roll_no** → **name** is a non-trivial functional dependency, since the dependent **name** is **not a subset of** determinant **roll_no.** Similarly, **{roll_no, name}** → **age** is also a non-trivial functional dependency, since **age** is **not a subset of {roll_no, name}**

In this case of the Tomy's University there are no trivial dependencies.

Step 6: NORMALIZATION OF THE REDUCED INITIAL FD DIAGRAM:

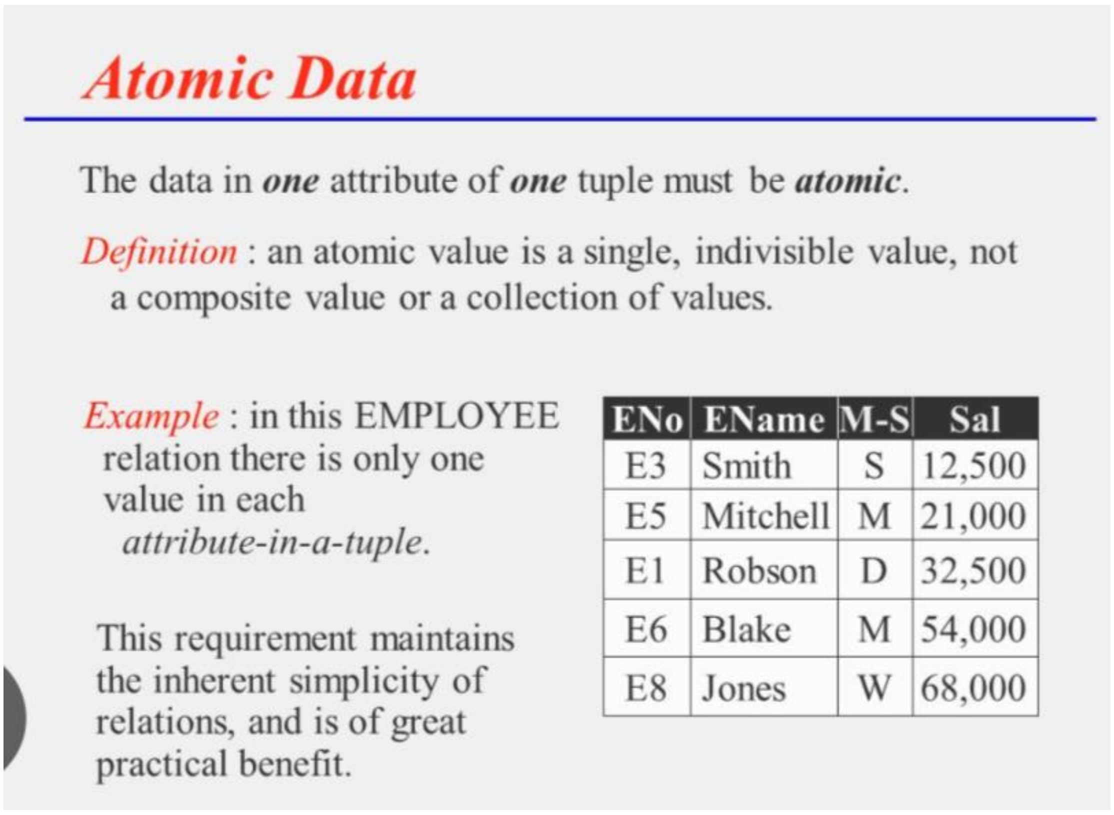
Normalization is conducted to remove all the anomalies which create data losses and loss of data integrity. It consists of 1NF to BCNF.

- **1NF:**

  In this normalization process the attribute values are further broken down to simpler atomic values to maintain data integrity.

  Atomic values are values of an attribute which is unique only to that cell of data. There shouldn't be another value in that cell.

  No multivalued attributes.



Atomic Data

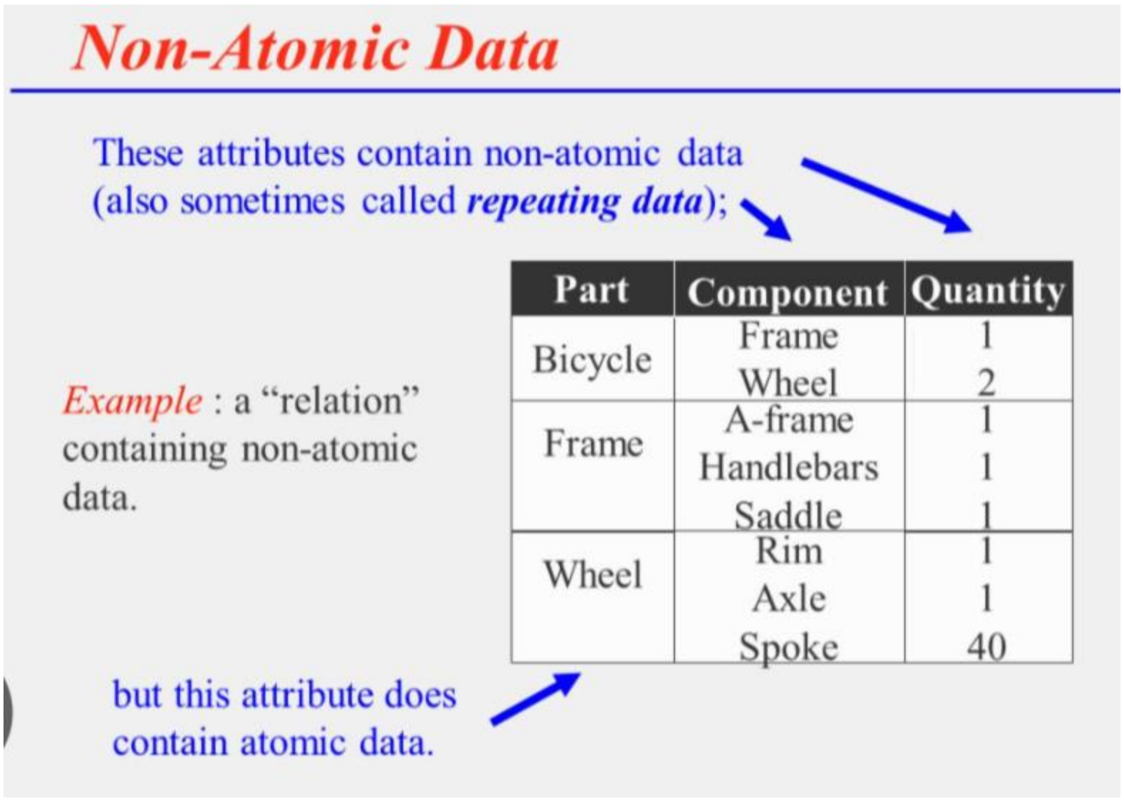
The data in ***one*** attribute of ***one*** tuple must be ***atomic***.

Definition : an atomic value is a single, indivisible value, not a composite value or a collection of values.

Example : in this EMPLOYEE relation there is only one value in each *attribute-in-a-tuple*.

This requirement maintains the inherent simplicity of relations, and is of great practical benefit.

ENo	EName	M-S	Sal
E3	Smith	S	12,500
E5	Mitchell	M	21,000
E1	Robson	D	32,500
E6	Blake	M	54,000
E8	Jones	W	68,000



Non-Atomic Data

These attributes contain non-atomic data (also sometimes called ***repeating data***);

Example : a "relation" containing non-atomic data.

Part	Component	Quantity
Bicycle	Frame	1
	Wheel	2
Frame	A-frame	1
	Handlebars	1
	Saddle	1
Wheel	Rim	1
	Axle	1
	Spoke	40

but this attribute does contain atomic data.

- **2NF:**

  In this type of normalization, it follows 1NF and then every non-key attribute is irreducibly dependent on the primary key such that there are no arrows coming out of a subset of the primary key. In simple words, each primary key will have its own unique table structure.

  No partial dependency, only accepts full dependency

  i.e.) AB→C, there shouldn't be A→C or B→C.

- **3NF:**

  In this type of normalization, it follows 2NF and every non-key attribute is non-transitively dependent on the primary key. That means all transitive dependencies must be eliminated.

- **BCNF (Boyce Codd Normal Form):**

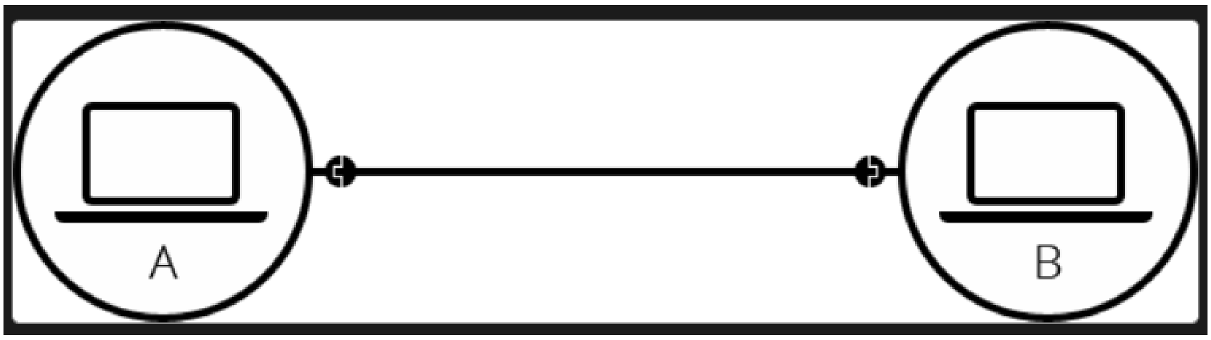
  This type of normalization follows 3NF and all determinants in the database are candidate keys or super key. If not, then it must be removed.

HOW DOES THE INTERNET WORK?

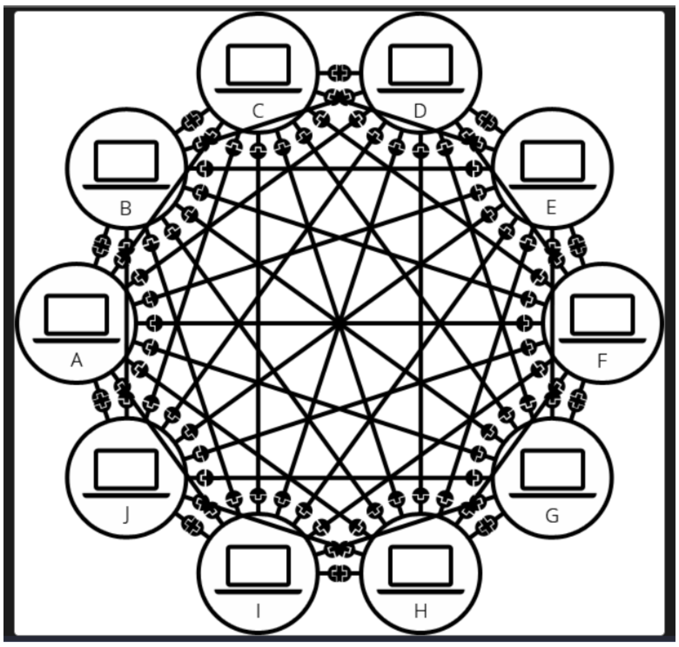
Internet is large network of computers which communicate to one another. It is considered as the backbone of the Web. The Internet is a technical infrastructure which allows billions of computers to be connected all together. Among those computers, some computers (called Web servers) can send messages intelligible to web browsers.

A simple network:

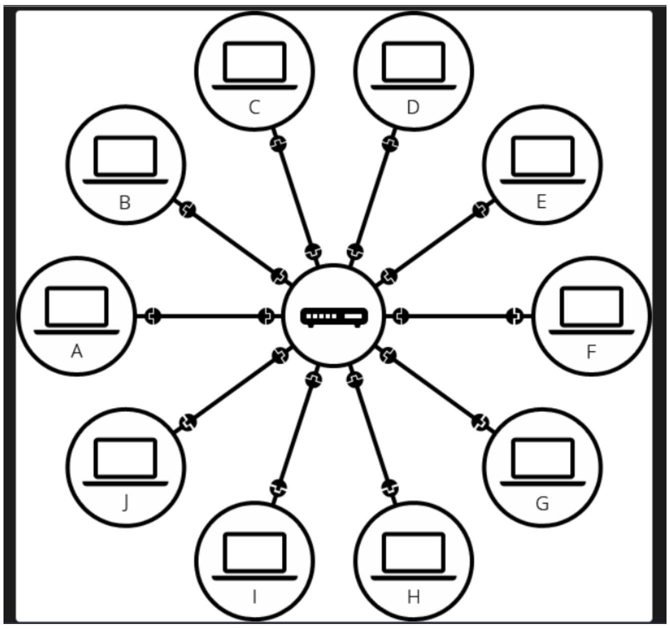
Let's consider how two different computers communicate to one another, we know that we must establish a connection between them. It can either be connected physically using an ethernet cable or wirelessly i.e. either by Bluetooth or Wi-Fi.



But in terms of connecting let's say 10 computers then we will have about 45 cables connecting to each and every computer. To avoid that we use a computer like medium called routers.

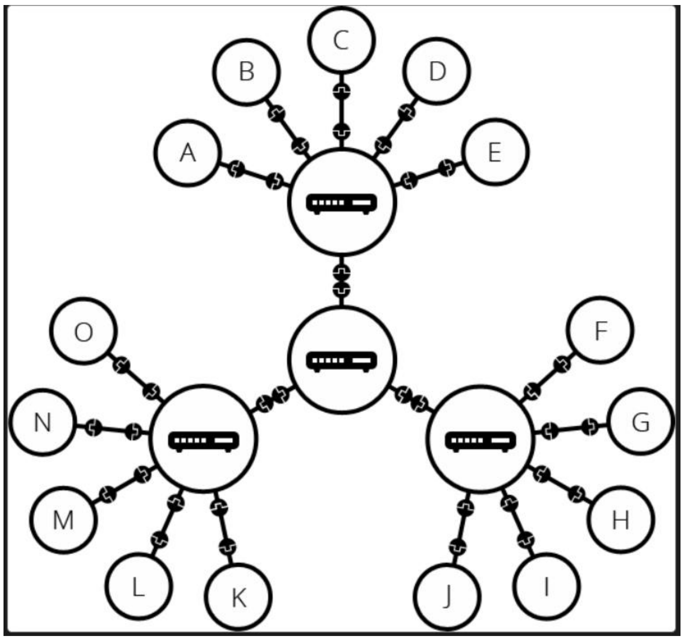


Through routers, all the computers need just a single cable to be connected to the router and the router helps in passing the message to that desirable computer.

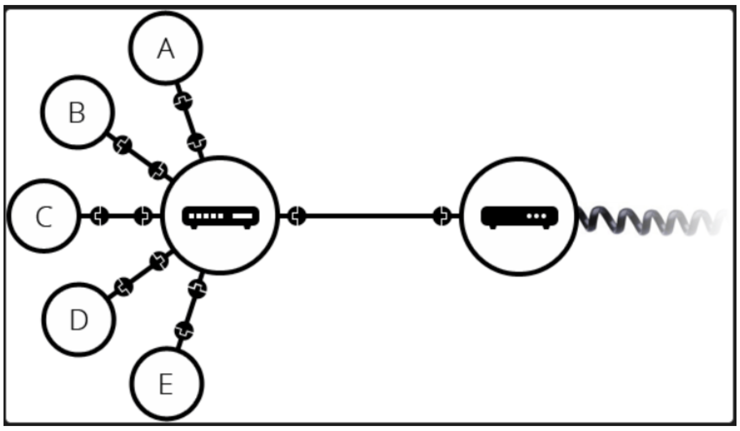


A network of networks

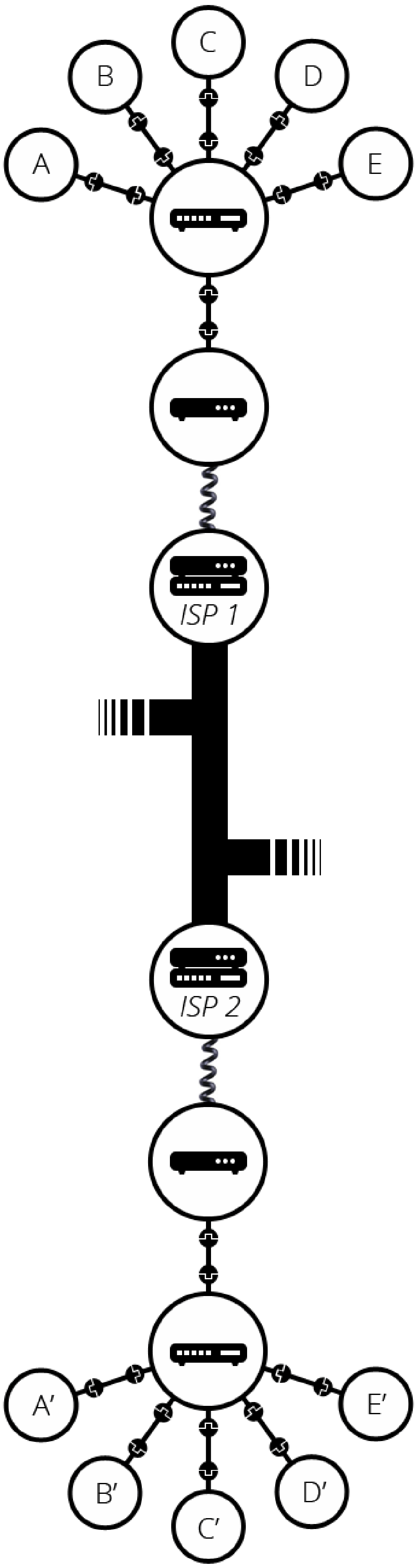
It is possible to connect between two routers like that of computers. Therefore, we can connect numerous amounts of routers such that it can scale to infinity.



This type of network will only be connected to oneself but to connect with other networks like our friends, coworkers etc we will need to connect it with them. But the problem with such connections we can't extend the cables to each place but instead we use telephone lines which extend to each place in the world. A device which will help the router to connect from its own network to another network is called a modem.



To communicate from one network to another, we use an ISP device to do so. The ISP device (i.e. Internet Service Provider) consists of special routers that are all linked together and can also access other ISP routers. So, the message carried from our network is carried through the network of ISP networks to the destination network.


C
B
D
A
E
ISP 1
ISP 2
A′
E′
B′
D′
C′

To send the message to that computer in the network we must use an IP address.

DIFFERENCE BETWEEN WEB PAGE, WEBSITE, WEB SERVER AND SEARCH ENGINE

Web page:

It is a simple document that is displayed by a web browser. Such documents are written in html language. All web pages can be accessed through a unique address.

Website:

A website is a collection of webpages that share a unique domain name. Each web page of a given website provides explicit links, most of the time in the form of clickable texts. To access the website, we must type its domain name in the browser.

Web server:

A web server is a computer hosting one or more websites. Hosting here means that the server consists of all the web files needed for a web page and the web server will send these files to open the web page of a web site to a web browser.

Search Engine:

Search Engine is a special kind of web site that helps users to find web pages from other websites. E.g.) Google, Bing, DuckDuckGo, Yahoo etc.

WHAT ARE WEB SERVERS AND HOW IT WORKS?

A web server is a combination of hardware and software,

- Hardware side: a web server is a computer that stores web server software and web files as well. It is also connected with the internet to interchange physical data with other devices connected to the web.
- Software side: This side of the server controls how the web users access certain files. This is usually conducted using HTTP servers, in which it sends certain files based on the domain name of the website provided by the web users.

The browser sends a request to the server to acquire the files. The web server based on the IP address it finds the location of these files and forwards it back to the web browser. This is all happening via HTTP.

HTTP provides certain rules which are:

- It's usually the client that send in a HTTP request to the server and the servers respond to a client's HTTP request. A server can also send the necessary web files before even the client requests for it by a mechanism called server push.

- When sending a request to a web server, the client must also send the URL in the HTTP.
- The web server must answer each request and send an error message at least.

On a web server side,

- When receiving the request from the web user, the web server performs search on the given URL and find if the URL matches with the existing files.
- If the web server finds the file, it sends the file back to the browser and if not, it will check if it can generate the file dynamically based on the request. This depends on static or dynamic content.
- If it still doesn’t find the file, then it would return a 404-error page.

Static and Dynamic content:

Static content: It’s those websites that have fixed content regardless of how many times we may reload it or re-host it. Usually consist of languages such as HTML, CSS & JavaScript. To make certain changes we must hard code it to make any changes, such as establishing certain names or characters in the header of the section. E.g. Portfolio websites, brochure websites etc.

Dynamic content: It’s those websites that have different content either when we reload the page or execute a search or click a button. These types of changes happen due to extra addition of languages such as Python, Ruby and PHP along with the typical languages (i.e. HTML, CSS & JavaScript). The changes happen on the go when the client requests for the web files from the web server, the output is pasted onto the web page and then send back to the client. E.g. Twitter, Netflix etc. In a dynamic server, it consists of an application server which helps in updating the necessary changes on the web page before sending it to the client.

DIFFERENCE BETWEEN HTTP AND HTTPS

HTTP:

- HTTP’s URL begins with http://.
- Uses port number 80 for communication.
- It is insecure such that any middleman between the browser and server can read the data easily.
- It works at Application Layer.
- It’s faster than HTTPS

HTTPS: This uses a combination of HTTP along with SSL/TLS. This encrypts all message substance, including the HTTP headers and the request/response data. The verification is done by a trusted third party who will sign in to the server-side digital certificates.

- URL begins with https://.
- Uses port number 443.
- It protects the passing of information at the cost of time.
- It's in Transport Layer.
- It is slower compared to HTTP

WHAT IS SSL AND TLS?

SSL(Secure Socket Layer):

SSL is a network protocol that provides security between the browser and the web server by encrypting the link between them.

How does SSL work?

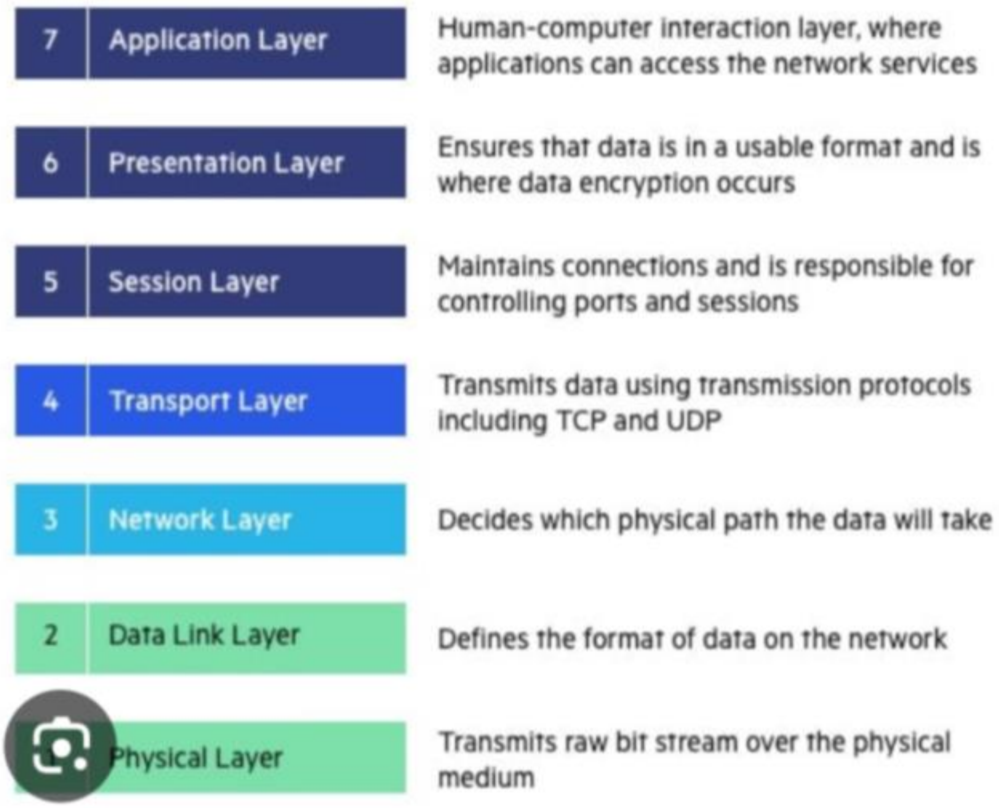
- Authentication: Well SSL starts by a handshake process and make sure to authenticate between the two devices and make sure that the two devices are who they said to be.
- Encryption: SSL encrypts data that is transmitted from one device to another, and makes sure that when someone intercepts the data, he/she will not be able to decode it.
- Data integrity: This makes sure that the data transmitted is the same that is sent by the sender and not tampered with.

TLS (Transport Layer Securities):

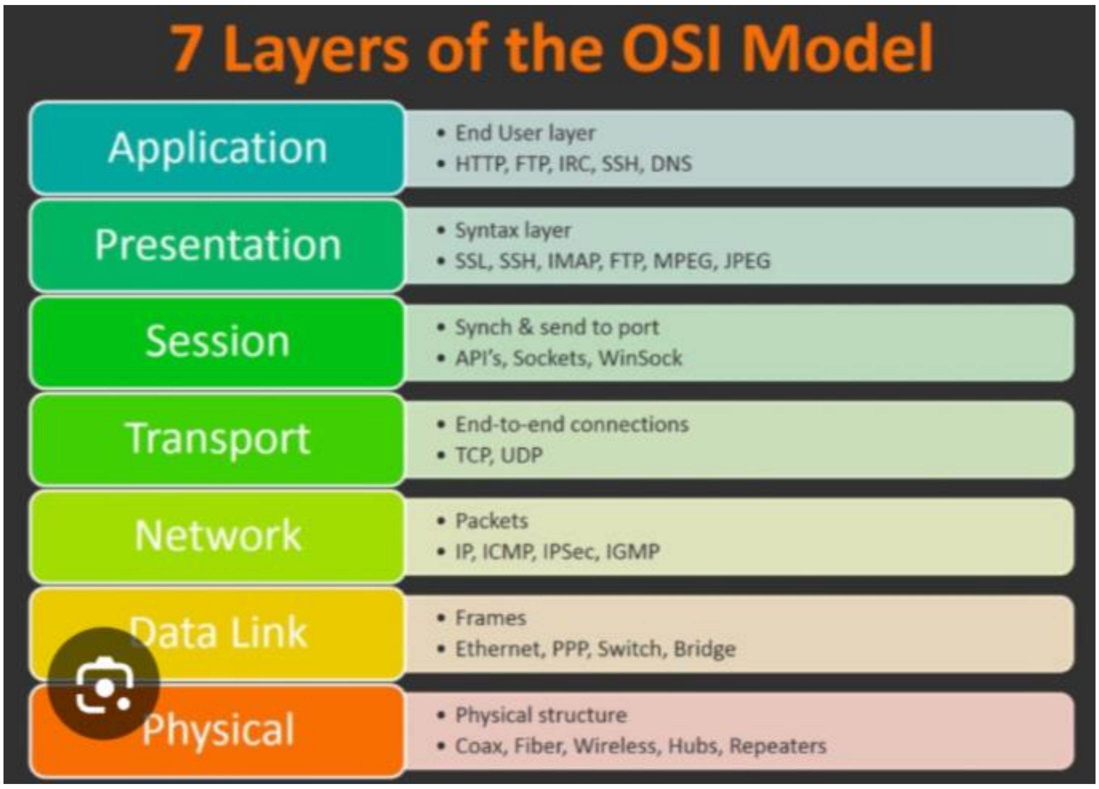
This type of security is done in the transport layer and ensures no third party overhear or tampers with the communication. It is an evolved form of SSL.

Both the TLS and SSL has a certificate on the origin server which holds information about who owns the domain and the server's public key, both of which are important for validating server's identity.

OSI MODEL



7	Application Layer	Human-computer interaction layer, where applications can access the network services
6	Presentation Layer	Ensures that data is in a usable format and is where data encryption occurs
5	Session Layer	Maintains connections and is responsible for controlling ports and sessions
4	Transport Layer	Transmits data using transmission protocols including TCP and UDP
3	Network Layer	Decides which physical path the data will take
2	Data Link Layer	Defines the format of data on the network
1	Physical Layer	Transmits raw bit stream over the physical medium

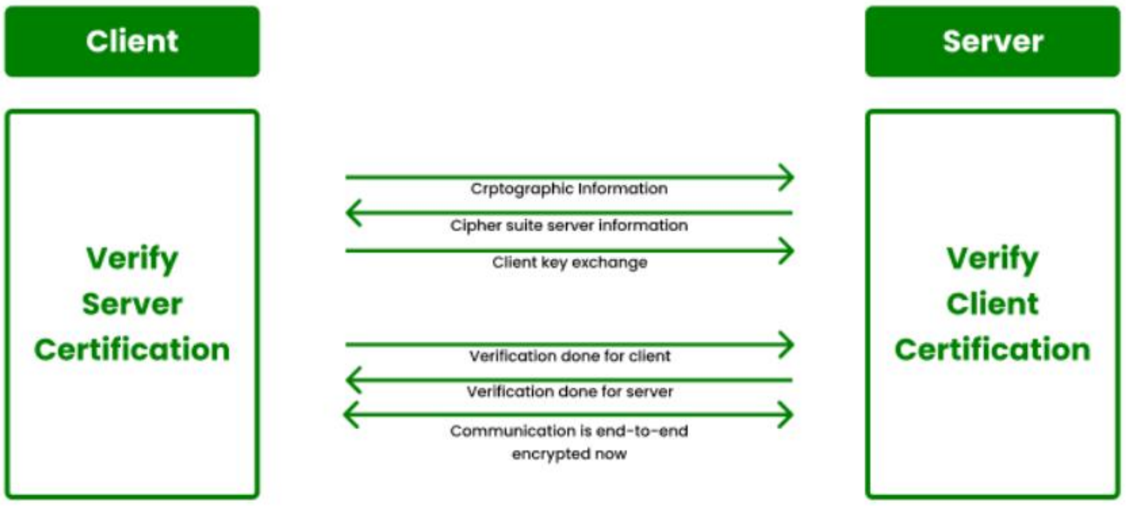


7 Layers of the OSI Model

Application	• End User layer • HTTP, FTP, IRC, SSH, DNS
Presentation	• Syntax layer • SSL, SSH, IMAP, FTP, MPEG, JPEG
Session	• Synch & send to port • API's, Sockets, WinSock
Transport	• End-to-end connections • TCP, UDP
Network	• Packets • IP, ICMP, IPSec, IGMP
Data Link	• Frames • Ethernet, PPP, Switch, Bridge
Physical	• Physical structure • Coax, Fiber, Wireless, Hubs, Repeaters

SSL/TLS 6-WAY HANDSHAKE

1) Browser sends a list of SSL/TLS versions and encryption algorithms that it can work with the server.
2) The server chooses the best encryption algorithm and TLS version and replies with a certificate which contains the server's public key, to verify with the server.
3) The browser checks if the server certificate is legitimate and generates a pre-master key so that later, they can generate a unique key.
4) The Browser also encrypts the pre-master key with the server's public key and sends it to the server.
5) The server decrypts it using its private key and acquire the premaster key which will make it a symmetric key.
6) Now the channel is secured and communication between the server and client is end-to-end encrypted.


Client
Server
Verify Server Certification
Verify Client Certification
Crptographic Information
Cipher suite server information
Client key exchange
Verification done for client
Verification done for server
Communication is end-to-end encrypted now

ABOUT APACHE AND MICROSOFT IIS

Apache:

Apache is an open-source web server which is responsible for accepting HTTP requests. Apache can run on almost any operating system such as windows, macOS, UNIX and Linux.

IIS:

It is a Microsoft web server that runs on windows operating system. It is not an open-source web server. It does accept HTTP, FTP and SMTP requests but will only run stable in windows OS. .NET and ASPX scripting languages are offered by Microsoft.

.NET is an open-source platform used for building desktop, web & mobile applications that can run natively on an OS.

Processes and Threads

What are Processes?

- A process is a set of instructions that a computer follows to perform a task.
- When you run a program, your computer creates a process to execute the code and produce the desired output.

What are Threads?

- Threads allow a computer to perform multiple tasks at the same time.
- A single process can contain multiple threads, each handling different parts of a task simultaneously.
- In a web browser, threads help manage various tasks like handling user interactions, rendering the display, and running scripts.

Browser Processes and Threads

Main Thread in Browsers:

- The main thread handles user events, renders and paints the display, and runs most of the code for web pages.
- JavaScript usually runs on this main thread, and if a script takes a long time, it can make the page unresponsive.

Web Workers:

- Web Workers allow scripts to run in the background, separate from the main thread.
- This prevents long-running scripts from blocking the main thread, keeping the page responsive.

Service Workers:

- Service Workers run in the background, even when the user is not actively using the web page.
- They can notify the user of updates (like new emails) and help cache data to load pages faster.

Chrome Browser Processes

Chrome uses several processes to manage different tasks. Here are some key ones:

1. **UI Process:** Manages the user interface, like the address bar and buttons.
2. **Browser Process:** Controls the overall browser functions.
3. **Network Process:** Handles network requests.
4. **Storage Process:** Manages data storage.
5. **Plugin Process:** Manages plugins like Flash.
6. **Renderer Process:** Renders the content of web pages.
7. **Device Process:** Handles communication with hardware devices.
8. **GPU Process:** Manages tasks that use the Graphics Processing Unit (GPU).

How They Work Together:

- These processes work together to make the browser function smoothly.
- They communicate through Inter-Process Communication (IPC) to share necessary data.

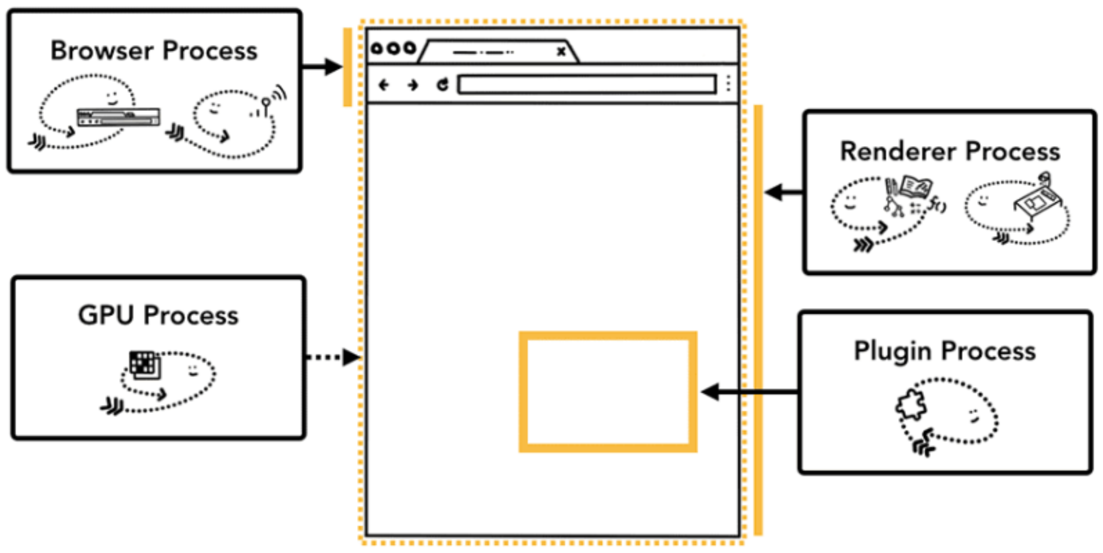


Figure 9: Different processes pointing to different parts of browser UI

Saving more memory - Servification in chrome:

Sometimes the web browser tends to either split the processes into several services or merge it into one aggregate function. This depends on the computer being used, if the computer has a powerful hardware, then the processes are split into each service and if not, then the process will be aggregated together to save memory footprint.

In terms of Chrome. We begin with the browser process.

Step 1: Handling inputs.

This is done by the browser process; the user puts forth the text into a search bar and the UI processor gets initiated to check whether the text is a URL link or if it's search query.

Step 2: Start Navigation:

In this step the user clicks enter and then the browser process starts searching for the DNS of the provided link and find the IP address of that link and at the same time establish an TLS protocol connection between the browser and the web server. This is all done with the help of the network process which is usually under the browser process.

Step 3: Read Response:

In this step a response body is acquired from the web server in which the network process reads it and checks the type of file it is being sent and the data type. It performs the necessary comparison between malicious site and the data acquired to display a warning page. If the file type is in HTML format, then it is passed to the Render Process or else if it is in the form of a zip file then it is given for downloading. If there is any missing data or wrong data, it performs MIME Type sniffing.

Step 4: Find a Render Process:

Once network process accepts the given URL link and it's respond body, it informs the UI process to find a Render Process. Usually in chrome once a URL link is given by the and UI process confirms it to be a link, it performs step 2 and simultaneously the UI process establishes a connection between the render process and start passing the data for rendering. Once the rendering is done, the Render Process is on standby before displaying it and it waits until the network process gives an approval to the UI process.

Step 5: Commit Navigation:

In this step and IPC is established between the browser process and the render process, so that it can continuously send the html data to the render process. Once a commit is heard from the renderer process side then the Navigation is successful.

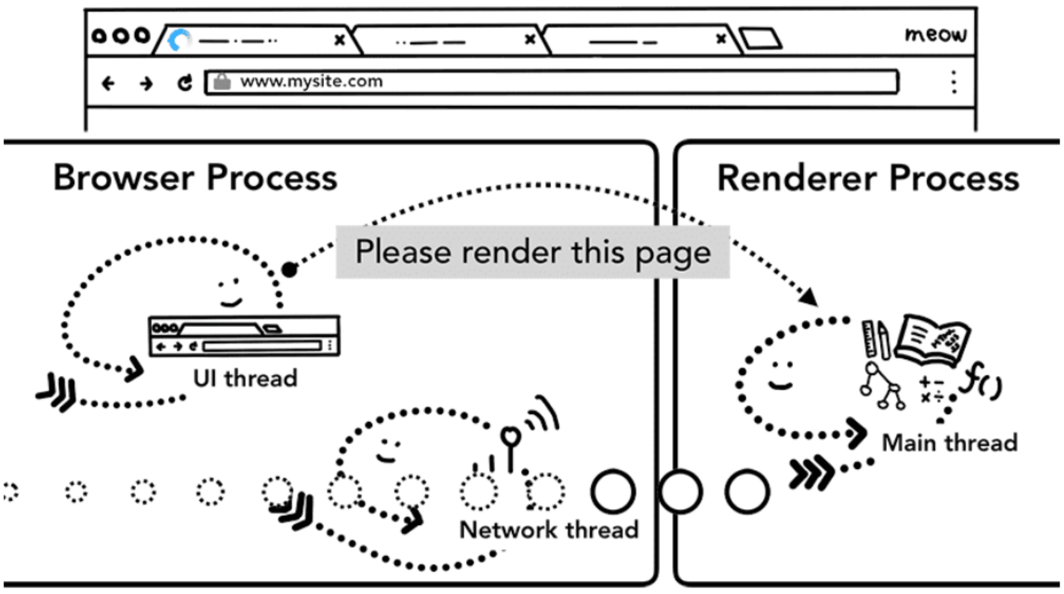


Figure 6: IPC between the browser and the renderer processes, requesting to render the page

Step 6: Initial Load is Complete:

In this step, the Renderer sends back an IPC stating that it has finished rendering each HTML data it has received. This is the point where the load animation on the tab bar stops indicating that the web page has completely rendered the HTML file.

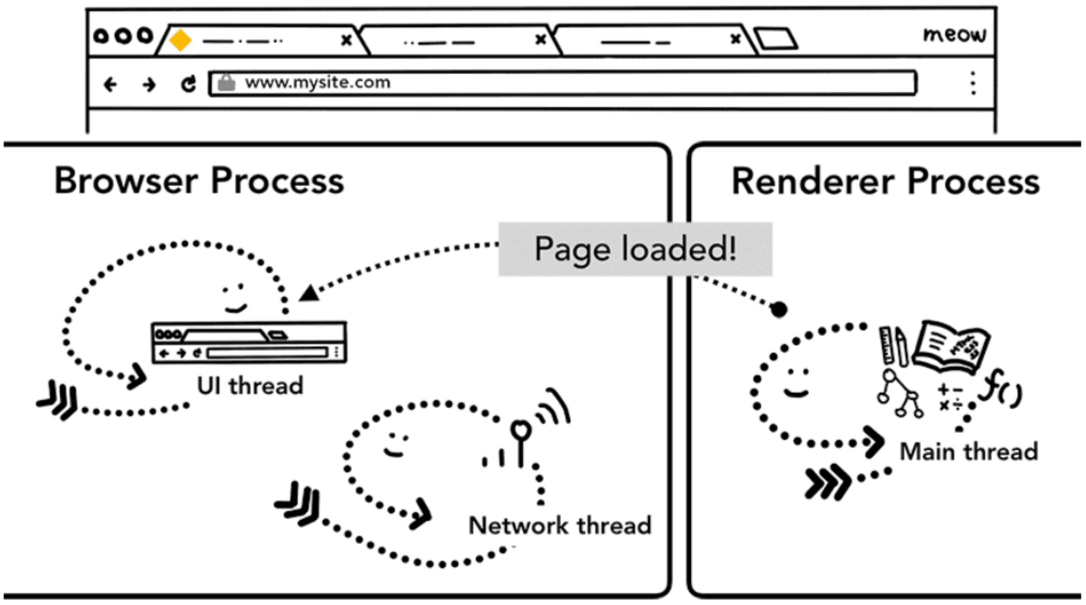


Figure 7: IPC from the renderer to the browser process to notify the page has "loaded"

RENDERER PROCESS:

In this part we will further discuss about how the Renderer will work in the web browser. This discusses about how a parsing is conducted on each data.

Parsing is a method of analysing the symbols in each file.

DOM:

First the HTML file is parsed into DOM (Document Object Model), which is an internal representation of the page and data structure.

These parsers are usually in the form of trees and in a Renderer process there will be two parses which are in the form of trees. They are made in the basis of HTML and CSS.

Both will be merged to form this unique parser called the layout tree that will describe the design and structure of the web page and will be rendered into the user's web browser.

This parse tree is also known as DOM tree.

Paint:

Painting is another step in terms of Rendering which will determine the shape, size, location, order and colour of each element.

Renderer Process communicates with the browser process in the form of IPC which can either be in the form of stack, queues and pipelining.

LIGHTHOUSE :

It's a tool used to measure the performance of a website or web page to determine how fast it can render in. The longer it takes to render, the more inefficient it becomes.

CACHE:

Cache is used for storing data's that are easy to access by the CPU. They are usually faster than RAM and smaller in terms of memory. Cache is usually volatile, and it is made from SRAMs. Cache doesn't store any programming codes like RAM.

L1, L2 & L3 CACHE:

Cache can be classified based on each level that varies based on speed, size and location.

L1 (Level 1 Cache or Primary Cache):

L1 cache is the smallest and the fastest memory level. In this case it consists of 64KB memory such that in a quad core computer it will have 256KB worth of memory. L1 cache is further split into two L1-I (Instruction) and L1-D (Data).

L2 (Level 2 Cache or Secondary Cache):

L2 cache is slightly larger than the L1 cache and is slightly slower as well. A high end L2 Cache usually uses about 32 MB but it's usually ranges from 6-12MB.

L3 Cache:

This is larger and slower cache memory compared to L1 and L2 such that it might be just 2 times faster than RAM itself. This type of cache was kept as a separate chip from the motherboard but now it's kept on the motherboard for efficiency's sake. This is large enough to store bigger data and this helped in improving its performance.

SRAM (Static-RAM):

This is a semiconductor which consists of 4-6 transistors and once the flip flop stores the bit, it keeps it stored until the opposite bit is stored in it. It is used as cache memory for CPUs.

Difference between SRAM & DRAM(Dynamic-RAM):

SRAM:

- It stores information if the power is supplied.
- Transistors are used to store information in SRAM.
- Since there are no capacitors there's no refreshing unit required.
- SRAM is faster than DRAM
- More expensive
- Used as cache memories
- Low Latency
- Consumes less power and generates less heat
- Has higher data transfer rate
- Used for high performance applications.

DRAM:

- It stores information if there is power supplied and a few milliseconds when the power is off.
- Uses capacitors to store information
- Refreshing unit needed
- DRAM is slower

- Less Expensive
- Used in main memories.
- High Latency
- Consumes more power and generates more heat.
- Lower Data transfer rate
- Used for general purpose applications.

WHAT IS MVC?

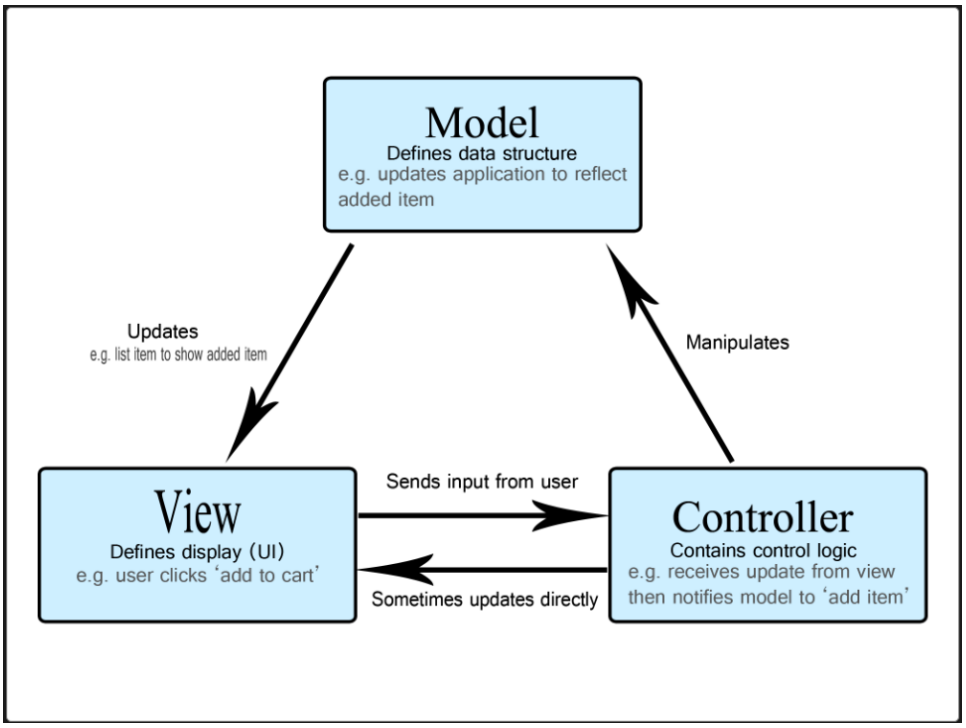
Model View Controller (MVC) is a commonly used pattern that is used for implementing user interface, data and controlling logic. MVC was created by Trygve Reenskaug.

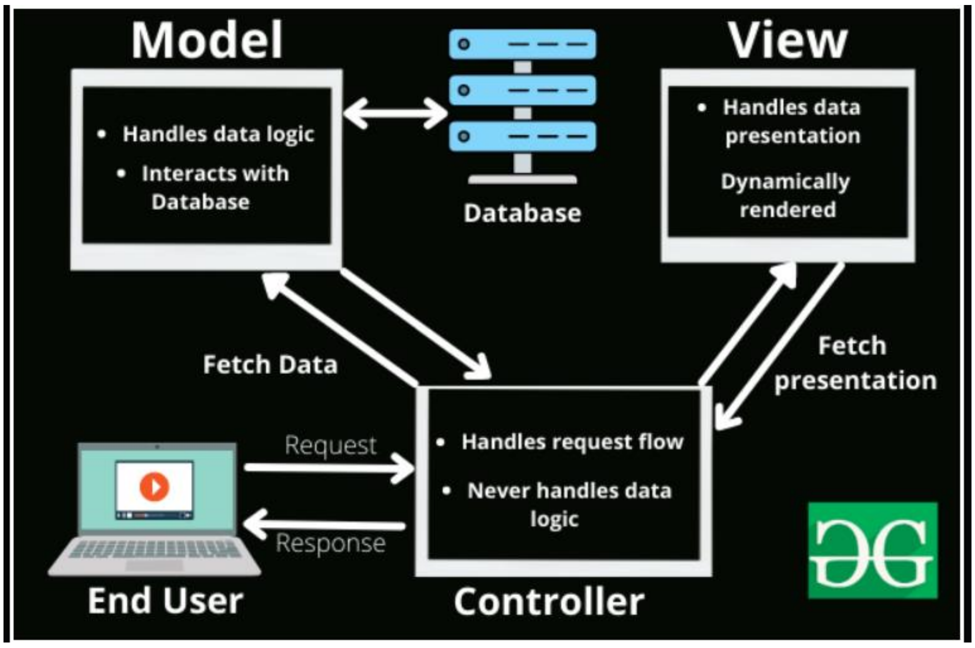
Before we delve deeper into the topic there are a few phrases that must be looked upon:

- Business logic: it is the custom rules for handling the transfer of data between a database and user interface.

MVC can be further split into:

- Model: Used for managing the data and business logic.
- View: Handles layout and display.
- Controller: Routes command over the model and view part.


Model
Defines data structure
e.g. updates application to reflect added item
Updates
e.g. list item to show added item
Manipulates
View
Defines display (UI)
e.g. user clicks 'add to cart'
Sends input from user
Sometimes updates directly
Controller
Contains control logic
e.g. receives update from view then notifies model to 'add item'



MODEL:

The model usually contains the structure or types of data which is used for displaying in the UI. If the data changes, then the view also changes.

VIEW:

The view makes sures how data should be displayed to the user.

CONTROLLER:

The controller has the logic means to control the view and model. It ensures that the view will have the ability to acquire the input response from the user and it will have the ability to update the model.

MVC ON THE WEB :

In terms of the web page the Model used is usually the Database model such as SQL Server and MySQL server. The view part is usually the browser which acts the UI viewport and is controlled and arranged by the codes such as HTML and CSS. The controller part is usually the python flask, Django or JavaScript which logically manipulate both the data stored in the model and how the data is displayed in the view.

SOURCE CODE:

VIEW:

Front End:

base.html (the home page):

```
<!DOCTYPE html>

<html lang="en-US" dir="ltr">

  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.1.3/dist/css/bootstrap.min.css"
integrity="sha384-
MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"
crossorigin="anonymous">

<link rel="stylesheet" href="{{ url_for('static', filename='base.css') }}">

<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/popper.js@1.14.3/dist/umd/popper.min.js"
integrity="sha384-
ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49"
crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.1.3/dist/js/bootstrap.min.js"
integrity="sha384-
ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy"
crossorigin="anonymous"></script>

<body>

<head>

  <title>Employee Details</title>

  <div class="Header">

    <a href="/"><img src="{{ url_for('static', filename='Elytra_Wings2.png') }}"></a>

    <div class="nav-buttons">

      <button onclick="window.location.href='/'" class="button1">Home</button>
```

```
          <button onclick="window.location.href='/addemp'" class="button1">Add
Employee</button>

       </div>

    </div>

   <script crossorigin="anonymous"

   src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js" >

   </script>

</head>

<h1>EMPLOYEE DETAILS:</h1>


    <table>

       <thead>

          <tr>

          <th scope="col">EMPID</th>

          <th scope="col">EMPNAME</th>

          <th scope="col">DESIGNATION</th>

          <th scope="col">DEPT</th>

          <th scope="col">Update/Delete</th>

          </tr>

       </thead>

       <tbody>

          {% for tip in mssqltips %}

             <tr>

                <td>{{ tip.EMPID }}</th>

                <td>{{ tip.EMPNAME }}</td>

                <td>{{ tip.DESIGNATION }}</td>

                <td>{{ tip.DEPT }}</td>

                <td><button onclick="window.location.href='updatemp/{{tip.EMPID}}'"
class="button2">&#9998;</button> / <button
```

```
onclick="window.location.href='go_to_popup-box/{{tip.EMPID}}'"
class="button3">&#x1F5D1;</button></td>

            </tr>

          {% endfor %}

        </tbody>

    </table>

    <div id="popup-box" class="modal">

        <div class="content">

            <h2>

                ALERT!

            </h2>

            <p>Are you sure you want to delete the content mentioned.</p>

            <button onclick="window.location.href='/delemp'" class="button4">Yes</button>

            <button onclick="location.href='#'" class="button4">No</button>

        </div>

    </div>

</body>

</html>
```

Add.html(page to add employees):

```
<!DOCTYPE html>

<html lang="en-US" dir="ltr">

<link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.1.3/dist/css/bootstrap.min.css"
integrity="sha384-
MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"
crossorigin="anonymous">

<link rel="stylesheet" href="{{ url_for('static', filename='Addd.css') }}">

<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous"></script>
```

```
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.14.3/dist/umd/popper.min.js"
integrity="sha384-
ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49"
crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.1.3/dist/js/bootstrap.min.js"
integrity="sha384-
ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy"
crossorigin="anonymous"></script>

  <head>

    <title>Add Employee details</title>

    <script crossorigin="anonymous"
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
></script>

    <div class="Header">

      <a href="/"><img src="{{ url_for('static', filename='Elytra_Wings2.png') }}"
id="img1"></a>

      {% block content %}

      {% with message = get_flashed_messages() %}

        {% if message %}

          {% for msg in message %}

          <div class="alert alert-warning alert-dismissible fade show" role="alert">

              {{msg}}

            <button type="button" class="close" data-dismiss="alert" aria-
label="Close">

              <span aria-hidden="true">&times;</span>

            </button>

          </div>

          {% endfor %}

        {% endif %}

      {% endwith %}

      <div class="nav-buttons">
```

```
            <button onclick="window.location.href='/'" class="button1">Home</button>
            <button onclick="window.location.href='/addemp'" class="button1">Add
Employee</button>
          </div>
          {% endblock %}
      </div>
   </head>
<body>
   <div class="cardthings">
      <div class="content">
          <div class="left">
          <h1>ADD NEW EMPLOYEE:</h1>
          <form action="/addemp" method="POST">
          <div class="textbox">
            <label>
               EMPID:
            </label>
            <br>
            <input type="text" name="EMPID" id="EMPID" required >
            <br>
          </div>
          <div class="textbox">
            <label>
               EMPNAME:
            </label>
            <br>
            <input type="text" name="EMPNAME" id="EMPNAEME" required >
            <br>
```

```
        </div>
        <div class="textbox">
          <label>
            DESIGNATION:
          </label>
          <br>
          <input type="text" name="DESIGNATION" id="DESIGNATION" required>
          <br>
        </div>
        <div class="textbox">
          <label>
            DEPT:
          </label>
          <br>
          <input type="text" name="DEPT" id="DEPT" required>
          <br>
        </div>
        <div class="textbox">
          <label>
            COMID:
          </label>
          <br>
          <input type="text" name="COMID" id="COMID" required>
          <br>
        </div>
        <button type="submit" class="button2">Submit</button>
        <button type="button"
onclick="window.location.href='/'"  class="button3">Cancel</button>
```

```
          </form>
        </div>
        <div class="right">
          <img src="{{ url_for('static', filename='designer3.png') }}" class="img2">
        </div>
    </div>
</body>
</html>
```

Base.css(Style sheet for home page):

```
body{
    background-color: #F6F2E9;
    padding:0;
    margin:0;
}
img{
    max-width: 170px;
    margin: 0;
}
.Header{
    display: flex;
    justify-content: space-between; /* Align items to the left and right */
    align-items: center; /* Align items vertically in the center
    padding: 10px 20px;*/
}
.nav-buttons{
    display: flex;
}
h1{
    font-size: 400%;
    font-family: serif;
    text-align: center;
}
ul{
    list-style-type: none;
    font-size: 20px;
    margin: 0;
    padding: 0px;
    overflow: hidden;
    padding-right:50px;
}
.button1{
    float: right;
    background-color: #E1C39F;
    border: none;
    color: black;
    padding: 15px 32px;
    text-align: center;
    text-decoration: none;
    font-size: 16px;
    transition-duration: 0.4s;
}
.button1:hover{
    background-color: white;
}
.button2{
    background-color: rgba(255, 255, 255, 0);
```

```
  color: black;
  text-align: center;
  text-decoration: none;
  font-size: 20px;
  transition-duration: 0.4s;
  border:none;
  border-radius: 50%;
}
.button2:hover{
  background-color: rgb(0, 206, 0);
}
.button3{
  background-color: rgba(255, 255, 255, 0);
  color: black;
  text-align: center;
  text-decoration: none;
  font-size: 20px;
  transition-duration: 0.4s;
  border-radius: 50%;
  border:none;
}
.button3:hover{
  background-color: rgb(255, 53, 53);
}
.button4{
  background-color: #E1C39F;
  color: black;
  text-align: center;
  text-decoration: none;

  font-size: 20px;

  transition-duration: 0.4s;

  border:none;

}

.button4:hover{

  background-color: rgb(255, 255, 255);

}

table,th,td{

  border: 1px solid;

  border-collapse:collapse;


}

table{

  width:100%;

  height:50%;

  margin-bottom: 30px;

}

th{

  background-color: #c1955f;

  height: 50px;

  text-align: center;

}

td{

  height:50px;

  text-align: center;

}

tr:nth-child(even) {
  background-color: #E1C39F;
}
.box {
  background-color: black;
  height: 100vh;
  display: flex;
  align-items: center;
  justify-content: center;
}

.box a {
  padding: 15px;
  background-color: #fff;
  border-radius: 3px;
  text-decoration: none;
}

.modal {
  position: fixed;
  inset: 0;
  background: rgba(254, 192, 126, 0.7);
  display: none;
  align-items: center;
  justify-content: center;
}

.content {
  position: relative;
```

```
   background: #F6F2E9;

   padding: 1em 2em;

   border-radius: 4px;

   text-align: center;

}


.modal:target {

   display: flex;

}
```

Update.html (Update page):

```
<!DOCTYPE html>

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.1.3/dist/css/bootstrap.min.css" integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO" crossorigin="anonymous">

<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/popper.js@1.14.3/dist/umd/popper.min.js" integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49" crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.1.3/dist/js/bootstrap.min.js" integrity="sha384-ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy" crossorigin="anonymous"></script>

<html lang="en-US" dir="ltr">

   <link rel="stylesheet" href="{{ url_for('static', filename='Addd.css') }}">

   <head>
```

```
    <title>
      Employee Update
    </title>
    <script crossorigin="anonymous"
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
></script>
    <div class="Header">
      <a href="/"><img src="{{ url_for('static', filename='Elytra_Wings2.png') }}"
id="img1"></a>
      <div class="nav-buttons">
        <button onclick="window.location.href='/'" class="button1">Home</button>
        <button onclick="window.location.href='/addemp'" class="button1">Add
Employee</button>
      </div>
    </div>
  </head>
  <body>
    <div class="cardthings">
      <div class="content">
        <div class="left">
        <h1>
          EDIT EMPLOYEE
        </h1>
        <form action="" method="POST">
          <div class="textbox">
          <label>
            EMPLOYEE NAME:
          </label>
          <br>
```

```
            <input type="text" name="EMPNAME" value="{{tip.EMPNAME}}" required>

            <br>

            </div>

            <div class="textbox">

            <label>

              DESIGNATION:

            </label>

            <br>

            <input type="text" name="DESIGNATION" value="{{tip.DESIGNATION}}"
required>

            <br>

            </div>

            <div class="textbox">

            <label>

              DEPT:

            </label>

            <br>

            <input type="text" name="DEPT" value="{{tip.DEPT}}" required>

            <br>

            </div>

            <button type="submit" class="button2">Submit</button>

            <button type="button"
onclick="window.location.href='/'"  class="button3">Cancel</button>

          </form>

        </div>

        <div class="left">

          <img src="{{ url_for('static', filename='Designer4.png') }}" class="img2">

        </div>

      </div>
```

```
  </body>
</html>
```

Addd.css(Style Sheet for update and add employee page):

```
#img1{
  max-width: 170px;
  margin: 0;
}
.Header{
  display: flex;
  justify-content: space-between; /* Align items to the left and right */
  align-items: center; /* Align items vertically in the center
  padding: 10px 20px;*/
}
.nav-buttons{
  display: flex;
}

h1{
  padding-top: 20px;
  font-size: 200%;
  font-family: serif;
}

.textbox{
  padding-bottom: 30px;
}

body{
  background-color: #F6F2E9;

  text-align: left;

}

.cardthings{

  display: flex;

  justify-content: center;

  align-items: center;

}

.content{

  display: flex;

  background-color: white; /* Ensure the box has a white background */

  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

  width: 65%;

  padding-bottom: 20px;

  border-radius: 6px;

  justify-content: space-between;

}


.img2{

    max-width: 100%; /* Ensure the image doesn't exceed its container */

    height: 100%;

    padding-top: 20px;

    padding-right: 20px;

}


input[type=text]{

  border-radius: 6px;

  border: 1px solid lightgray;
  width: 75%;
  font-size: 15px;
}

input[type=text]:focus {
  border: 3px solid #555;
}

.button2{
  background-color: #E1C39F;
  font-size: 20px;
  text-align: center;
  border: none;
  padding: 15px 32px;
  display: inline-block;
  border-radius: 14px;
  transition-duration: 0.4s;
}

.button2:hover{
  background-color: rgb(0, 206, 0);
}
.button3{
  background-color: #E1C39F;
  font-size: 20px;
  text-align: center;
  border: none;
  padding: 15px 32px;
  display: inline-block;
  border-radius: 14px;
  transition-duration: 0.4s;
}

.button3:hover{
  background-color: rgb(255, 53, 53);
}

.button1{
  float: right;
  background-color: #E1C39F;
  border: none;
  color: black;
  padding: 15px 32px;
  text-align: center;
  text-decoration: none;
  font-size: 16px;
  transition-duration: 0.4s;
}

.button1:hover{
  background-color: white;
}
.right{
  text-align: right;
  width: 50%;
}
```

```
.left{
   text-align: left;
   padding-left: 25px;
   width: 50%;
}
```

MODEL:

Model.py:

```
import sqlalchemy as sa
from sqlalchemy import create_engine,text
import urllib


DRIVER_NAME = "SQL SERVER"
SERVER_NAME = r"LAPTOP-IM98DJ4G\SQLEXPRESS03"
DATABASE_NAME = "EMPLOYER"


connection_string = (
   "Driver={SQL Server};"
   "Server=LAPTOP-IM98DJ4G\\SQLEXPRESS03;"
   "Database=EMPLOYER;"
   "Trusted_Connection=yes;"
)
connection_url = urllib.parse.quote_plus(connection_string)


coxn = create_engine(f"mssql+pyodbc:///?odbc_connect={connection_url}")
print("Connection passed")


def acquire(mssqltips=[]):
  with coxn.connect() as connection:
    result = connection.execute(text("SELECT EMP_ID, EMP_NAME, DESIGNATION,
DEPT FROM EMPLOYEE"))
    for row in result.fetchall():
      mssqltips.append({
        "EMPID": row[0],
        "EMPNAME": row[1],
        "DESIGNATION": row[2],
        "DEPT": row[3],
      })
  return mssqltips
def count(EMPID,COMID):
  conn=coxn.raw_connection()
  cursor=conn.cursor()
  cursor.execute("SELECT COUNT(*) FROM EMPLOYEE WHERE EMP_ID=?",EMPID)
  result=cursor.fetchone()
  cursor.execute("SELECT COUNT(*) FROM EMPLOYEE WHERE COM_ID=?",COMID)
  result1=cursor.fetchone()
  cursor.close()
  conn.close()
  return result, result1
def insert(EMPID,EMPNAME,DESIGNATION,DEPT,COMID):
  conn=coxn.raw_connection()
  cursor=conn.cursor()
  cursor.execute("INSERT INTO EMPLOYER.dbo.EMPLOYEE (EMP_ID, EMP_NAME,
DESIGNATION, DEPT, COM_ID) VALUES
(?,?,?,?,?)",(EMPID,EMPNAME,DESIGNATION,DEPT,COMID))
  cursor.commit()
  cursor.close()
```

```
    conn.close()
    return
def fetch(cr=[],id=""):
    conn=coxn.raw_connection()
    cursor=conn.cursor()
    cursor.execute("SELECT * FROM EMPLOYER.dbo.EMPLOYEE WHERE EMP_ID=?",id)
    for row in cursor.fetchall():
        cr.append({"EMPID":row[0],"EMPNAME":row[1],"DESIGNATION":row[2],"DEPT":row[
3],"COMID":row[4]})
    conn.close()
    return cr
def updation(EMPNAME="",DESIGNATION="",DEPT="",id=""):
    conn=coxn.raw_connection()
    cursor=conn.cursor()
    cursor.execute("UPDATE EMPLOYER.dbo.EMPLOYEE SET EMP_NAME=?,
DESIGNATION=?, DEPT=? WHERE EMP_ID=?",EMPNAME,DESIGNATION,DEPT,id)
    cursor.execute("UPDATE EMPLOYER.dbo.EMPLOYEE SET DEPT=? WHERE EMP_ID=?",
DEPT,id)
    conn.commit()
    conn.close()
    return
def deletion(id=""):
    conn=coxn.raw_connection()
    cursor=conn.cursor()
    cursor.execute("DELETE FROM EMPLOYER.dbo.EMPLOYEE WHERE EMP_ID=?",id)
    cursor.execute("DELETE FROM EMPLOYER.dbo.COMMUNICATION WHERE
EMP_ID=?",id)
    cursor.execute("DELETE FROM EMPLOYER.dbo.PROJECT_SALARY WHERE
EMP_ID=?",id)
```

```
    conn.commit()
    conn.close()
    return
```

ROUTE:

app.py:

```
from flask import Flask, render_template, redirect, request, flash,url_for
from model import acquire,count,insert,fetch,updation,deletion
from controller import acquirenewinfo, updatefetch
app = Flask(__name__)
app.secret_key="Hello World"
temp=None
@app.route("/")
def base():
    mssqltips = []
    acquire(mssqltips)
    return render_template("base.html", mssqltips=mssqltips)
@app.route("/addemp", methods=['GET','POST'])
def addemp():
    if request.method=='GET':
        return render_template("Add.html")
    if request.method=='POST':
        EMPID,EMPNAME,DESIGNATION,DEPT,COMID,flag=acquirenewinfo()
        if(flag==1):
            flash("The EMPID filled in is incorrect please start with character E and make sure
it doesn't exceed 10 characters")
            return redirect('/addemp')
        if(flag==2):
```

```
      flash("Please don't fill the text boxes just with blank spaces")
      return redirect('/addemp')
    if(flag==3):
      flash("The COMID filled in is incorrect please start with character C and make sure
it doesn't exceed 10 characters")
      return redirect('/addemp')
    result,result1=count(EMPID,COMID)
    if(result[0]>0):
      flash("The entered EmpID is currently in use please try another EmpID")
      return redirect('/addemp')
    elif(result1[0]>0):
      flash("Entered COMID already exists please try a different one")
      return redirect('/addemp')
    else:
      insert(EMPID,EMPNAME,DESIGNATION,DEPT,COMID)
      return redirect('/')
@app.route('/updatemp/<string:id>',methods=['GET','POST'])
def update(id):
  cr=[]
  if request.method=='GET':
    fetch(cr,id)
    return render_template("update.html",tip=cr[0])
  if request.method=='POST':
    EMPNAME,DESIGNATION,DEPT,flag=updatefetch()
    if(flag==1):
      flash("Please don't fill the text boxes just with blank spaces")
      return redirect('/')
    updation(EMPNAME,DESIGNATION,DEPT,id)
```

```
      return redirect('/')
@app.route("/go_to_popup-box/<string:id>")
def popup(id):
   global temp
   temp=id
   return redirect(url_for('base')+'#popup-box')
@app.route('/delemp')
def delete():
   deletion(temp)
   return redirect('/')
if __name__ == "__main__":
   app.run(debug=True)
```

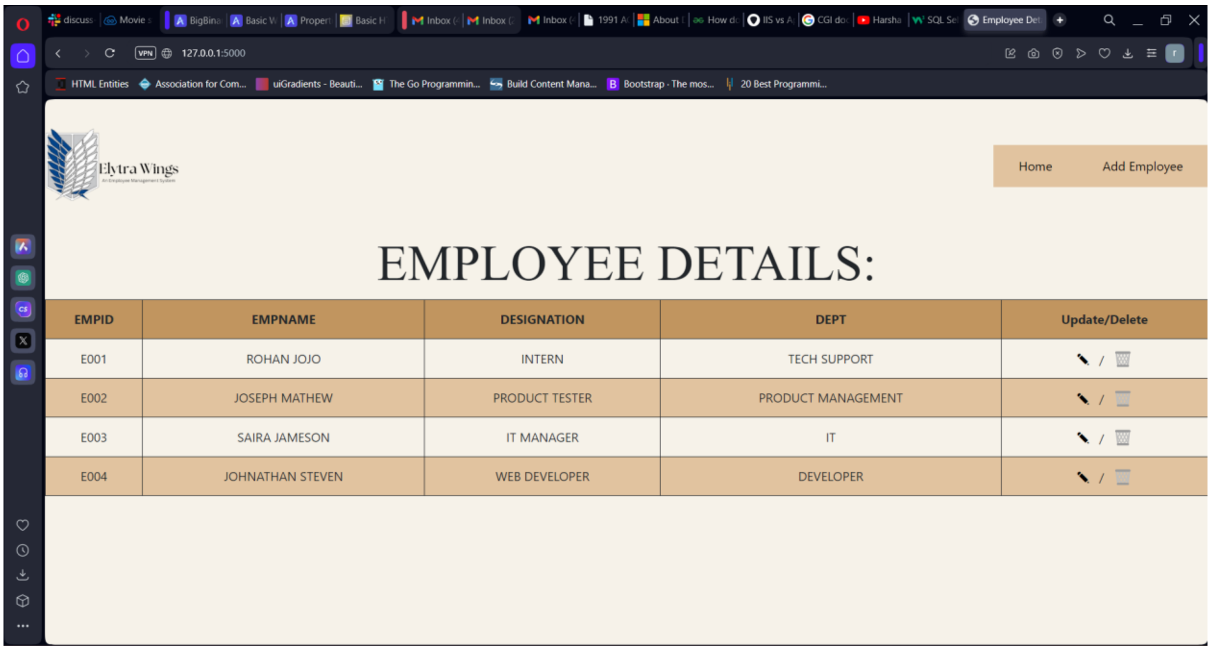
CONTROLLER:

Controller.py:

```
from flask import Flask, render_template, redirect, request, flash, url_for


def acquirenewinfo():
   flag=0
   EMPID=request.form["EMPID"]
   EMPNAME=request.form["EMPNAME"]
   DESIGNATION=request.form["DESIGNATION"]
   DEPT=request.form["DEPT"]
   COMID=request.form["COMID"]
   EMPID=EMPID.upper()
   EMPNAME=EMPNAME.upper()
   DESIGNATION=DESIGNATION.upper()
```

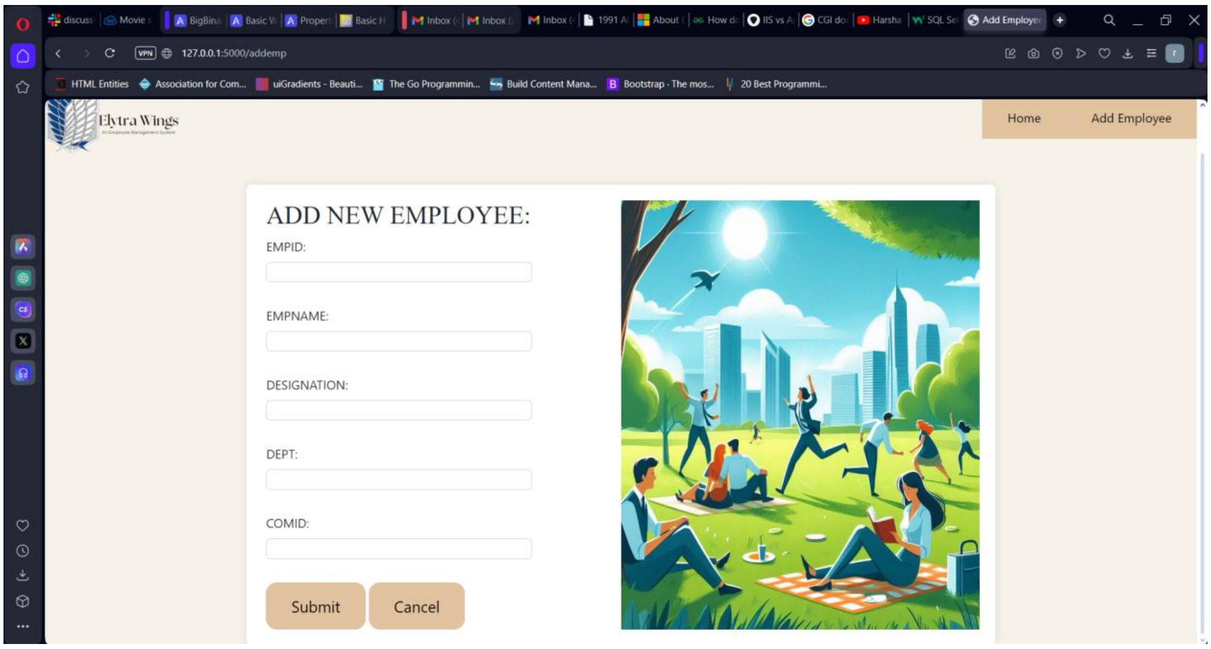
```
  DEPT=DEPT.upper()
  COMID=COMID.upper()
  if(EMPID[0]!='E' or len(EMPID)>10):
    flag=1
    return EMPID,EMPNAME,DESIGNATION,DEPT,COMID,flag
  if(EMPNAME.isspace()==True or DESIGNATION.isspace()==True or
DEPT.isspace()==True or COMID.isspace()==True):
    flag=2
    return EMPID,EMPNAME,DESIGNATION,DEPT,COMID,flag
  if(COMID[0]!='C' or len(COMID)>10):
    flag=3
    return EMPID,EMPNAME,DESIGNATION,DEPT,COMID,flag
  else:
    return EMPID,EMPNAME,DESIGNATION,DEPT,COMID,flag


def updatefetch():
  flag=0
  EMPNAME=request.form["EMPNAME"]
  DESIGNATION=request.form["DESIGNATION"]
  DEPT=request.form["DEPT"]
  EMPNAME=EMPNAME.upper()
  DESIGNATION=DESIGNATION.upper()
  DEPT=DEPT.upper()
  if(EMPNAME.isspace()==True or DESIGNATION.isspace()==True or
DEPT.isspace()==True):
    flag=1
    return EMPNAME,DESIGNATION,DEPT,flag
  else:
    return EMPNAME,DESIGNATION,DEPT,flag
```

OUTPUT:



EMPLOYEE DETAILS:

EMPID	EMPNAME	DESIGNATION	DEPT	Update/Delete
E001	ROHAN JOJO	INTERN	TECH SUPPORT	/
E002	JOSEPH MATHEW	PRODUCT TESTER	PRODUCT MANAGEMENT	/
E003	SAIRA JAMESON	IT MANAGER	IT	/
E004	JOHNATHAN STEVEN	WEB DEVELOPER	DEVELOPER	/



ADD NEW EMPLOYEE:

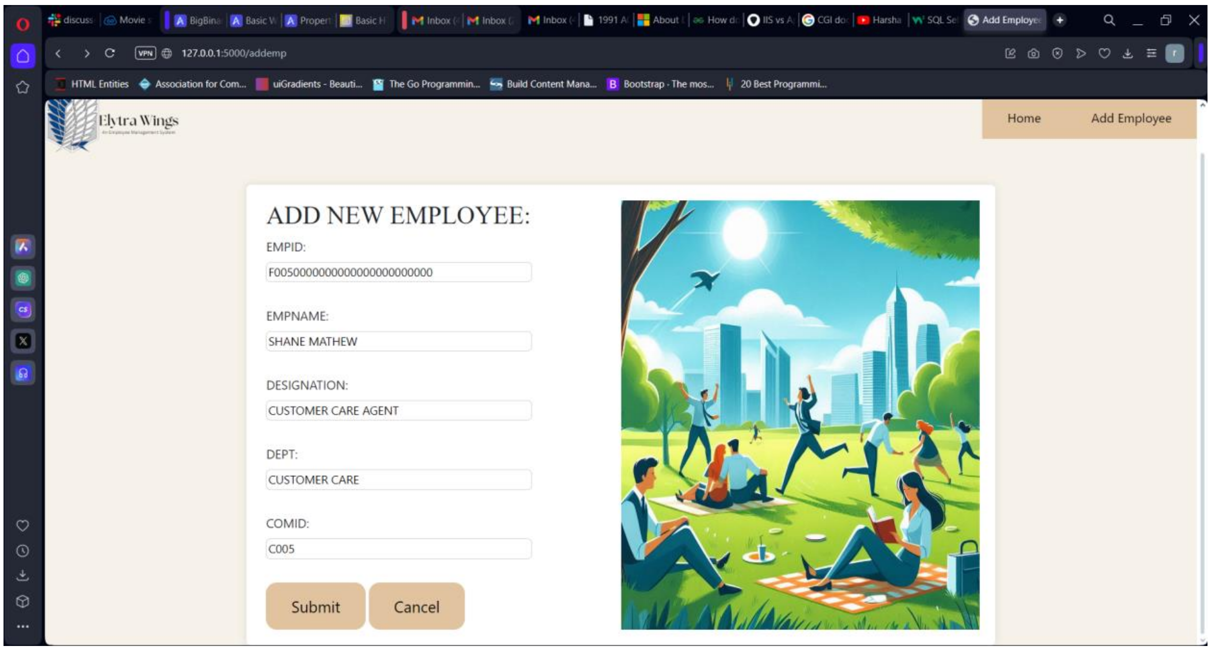
EMPID:

EMPNAME:

DESIGNATION:

DEPT:

COMID:



ADD NEW EMPLOYEE:

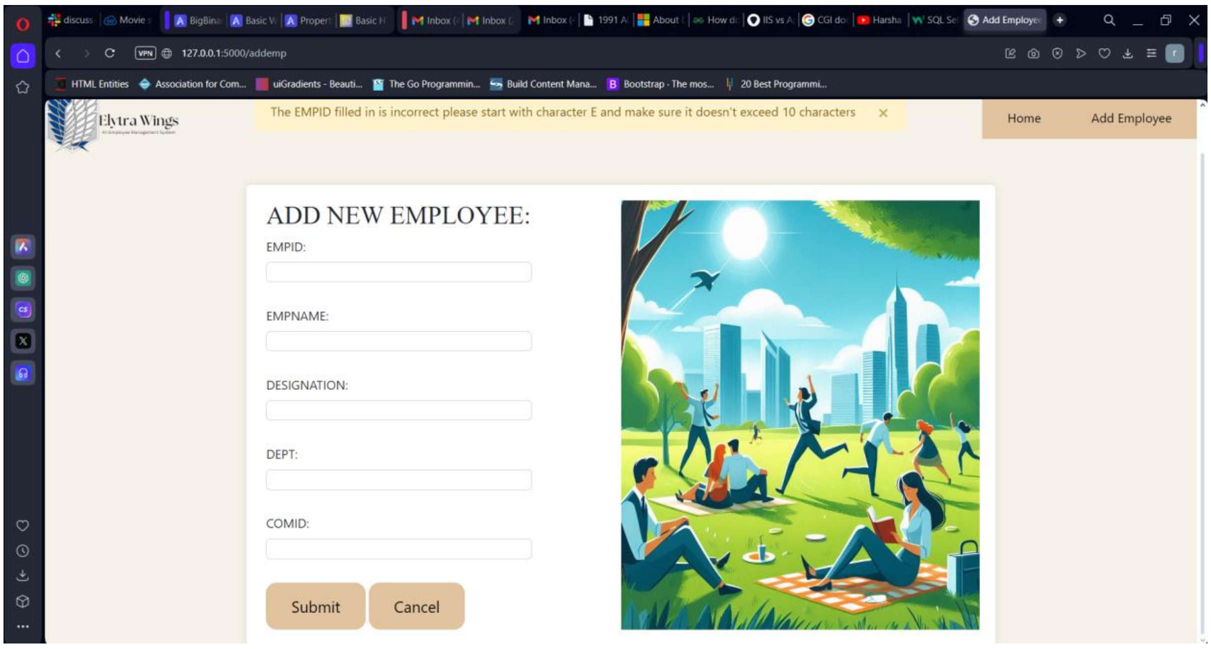
EMPID: F00500000000000000000000000

EMPNAME: SHANE MATHEW

DESIGNATION: CUSTOMER CARE AGENT

DEPT: CUSTOMER CARE

COMID: C005



The EMPID filled in is incorrect please start with character E and make sure it doesn't exceed 10 characters

ADD NEW EMPLOYEE:

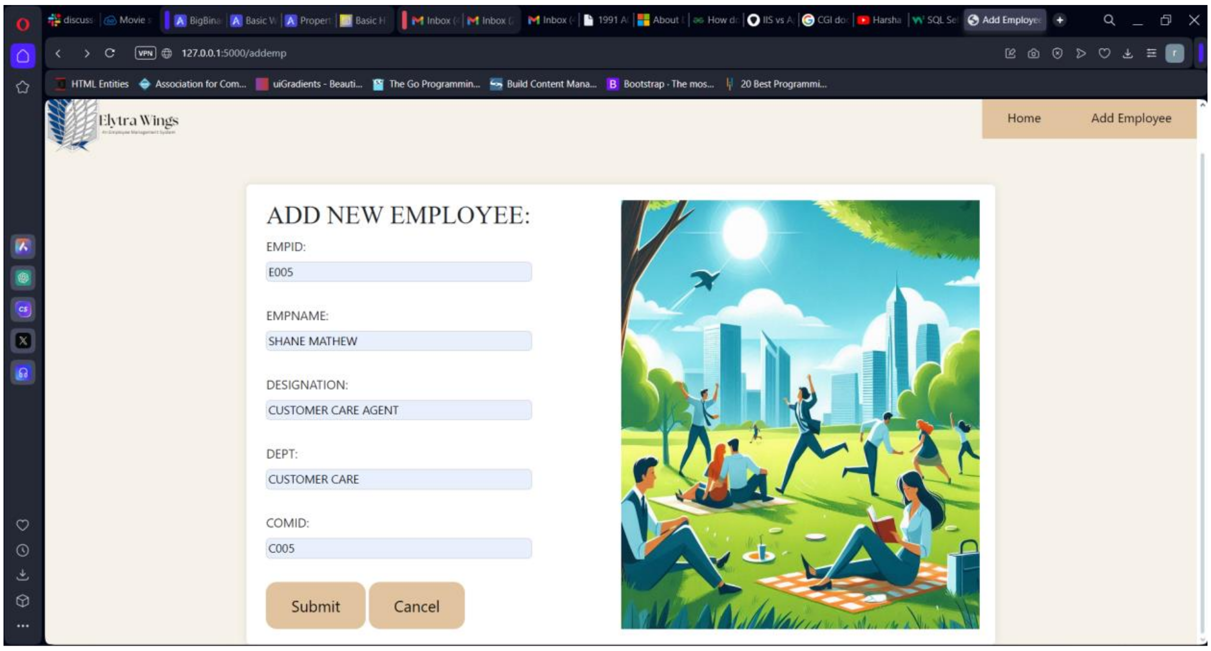
EMPID:

EMPNAME:

DESIGNATION:

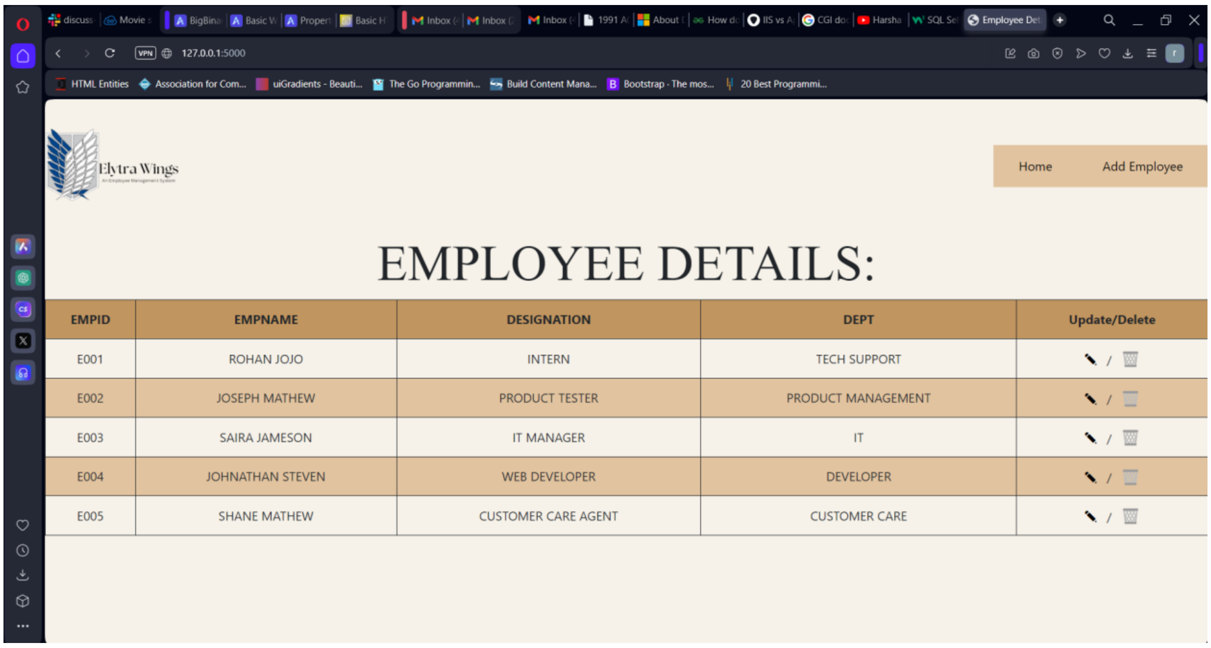
DEPT:

COMID:



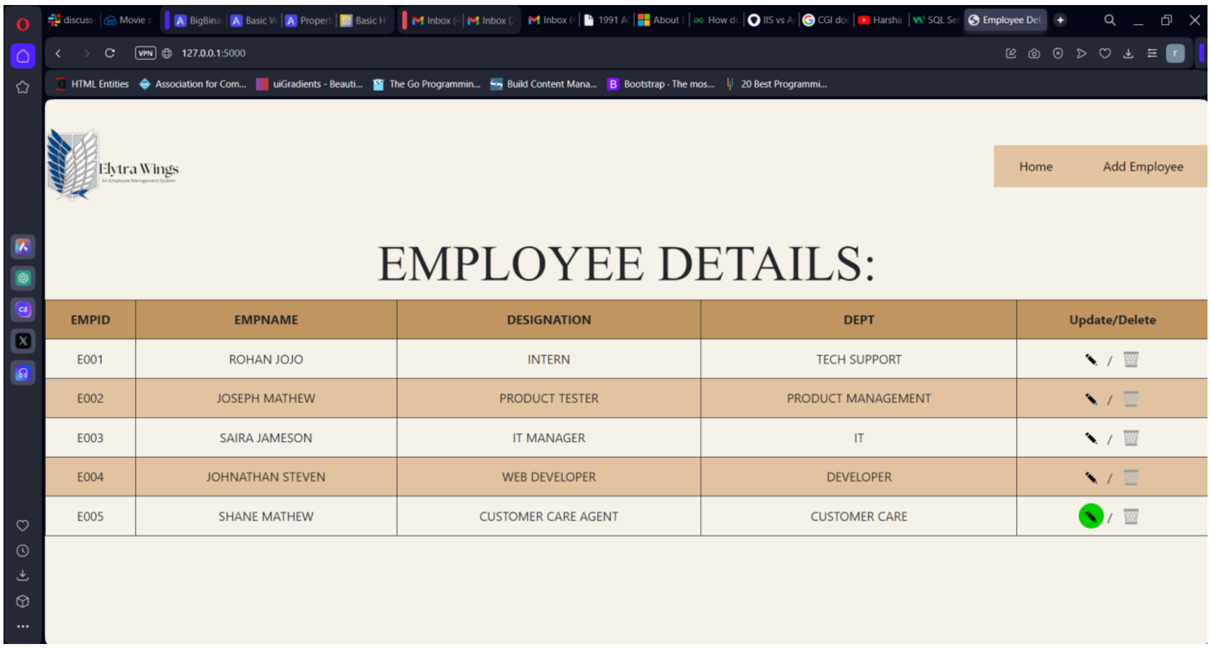
ADD NEW EMPLOYEE:

EMPID:	E005
EMPNAME:	SHANE MATHEW
DESIGNATION:	CUSTOMER CARE AGENT
DEPT:	CUSTOMER CARE
COMID:	C005



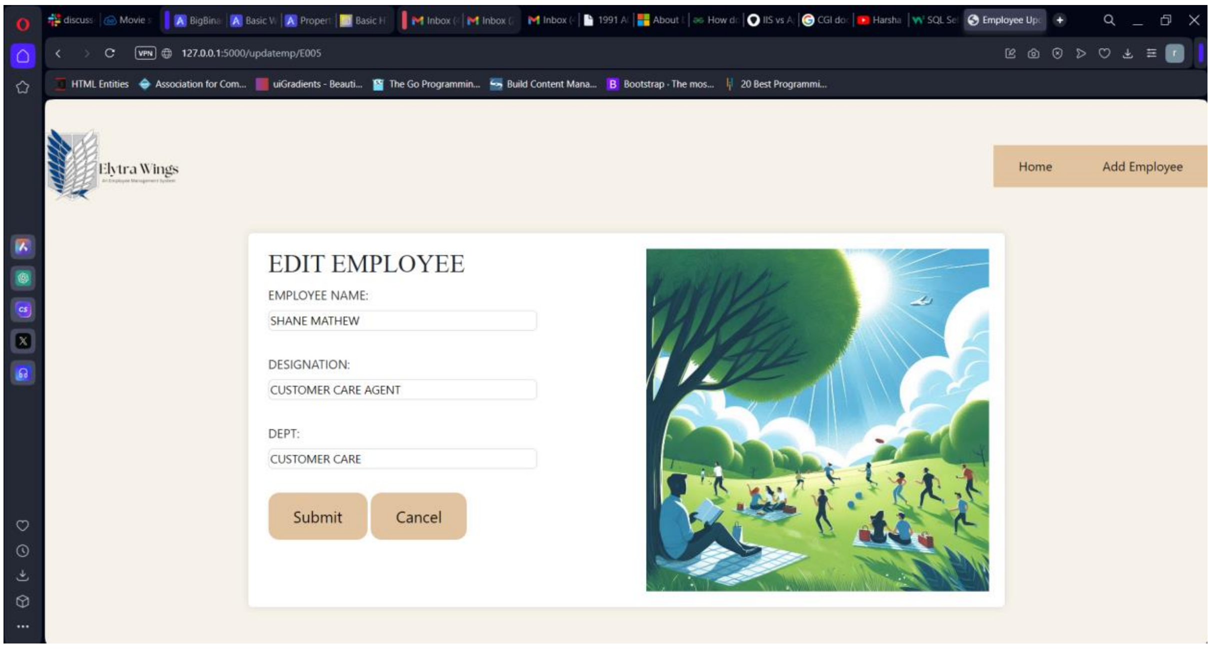
EMPLOYEE DETAILS:

EMPID	EMPNAME	DESIGNATION	DEPT	Update/Delete
E001	ROHAN JOJO	INTERN	TECH SUPPORT	/
E002	JOSEPH MATHEW	PRODUCT TESTER	PRODUCT MANAGEMENT	/
E003	SAIRA JAMESON	IT MANAGER	IT	/
E004	JOHNATHAN STEVEN	WEB DEVELOPER	DEVELOPER	/
E005	SHANE MATHEW	CUSTOMER CARE AGENT	CUSTOMER CARE	/



EMPLOYEE DETAILS:

EMPID	EMPNAME	DESIGNATION	DEPT	Update/Delete
E001	ROHAN JOJO	INTERN	TECH SUPPORT	✎ / 🗑
E002	JOSEPH MATHEW	PRODUCT TESTER	PRODUCT MANAGEMENT	✎ / 🗑
E003	SAIRA JAMESON	IT MANAGER	IT	✎ / 🗑
E004	JOHNATHAN STEVEN	WEB DEVELOPER	DEVELOPER	✎ / 🗑
E005	SHANE MATHEW	CUSTOMER CARE AGENT	CUSTOMER CARE	✎ / 🗑

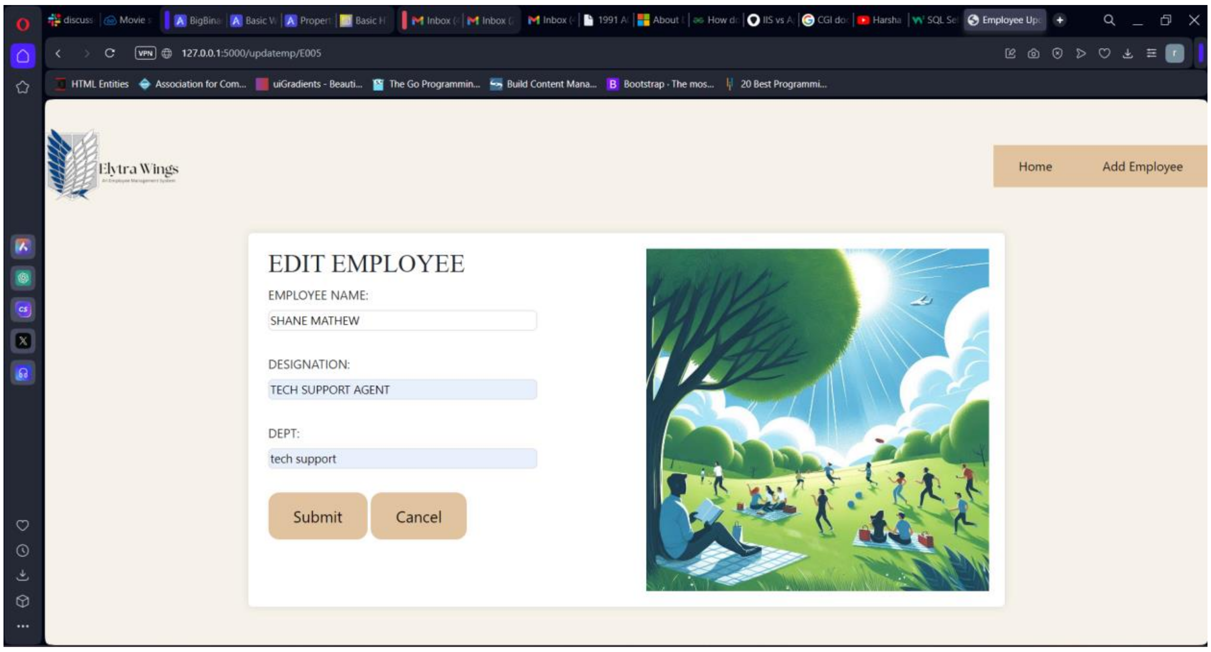


EDIT EMPLOYEE

EMPLOYEE NAME: SHANE MATHEW

DESIGNATION: CUSTOMER CARE AGENT

DEPT: CUSTOMER CARE

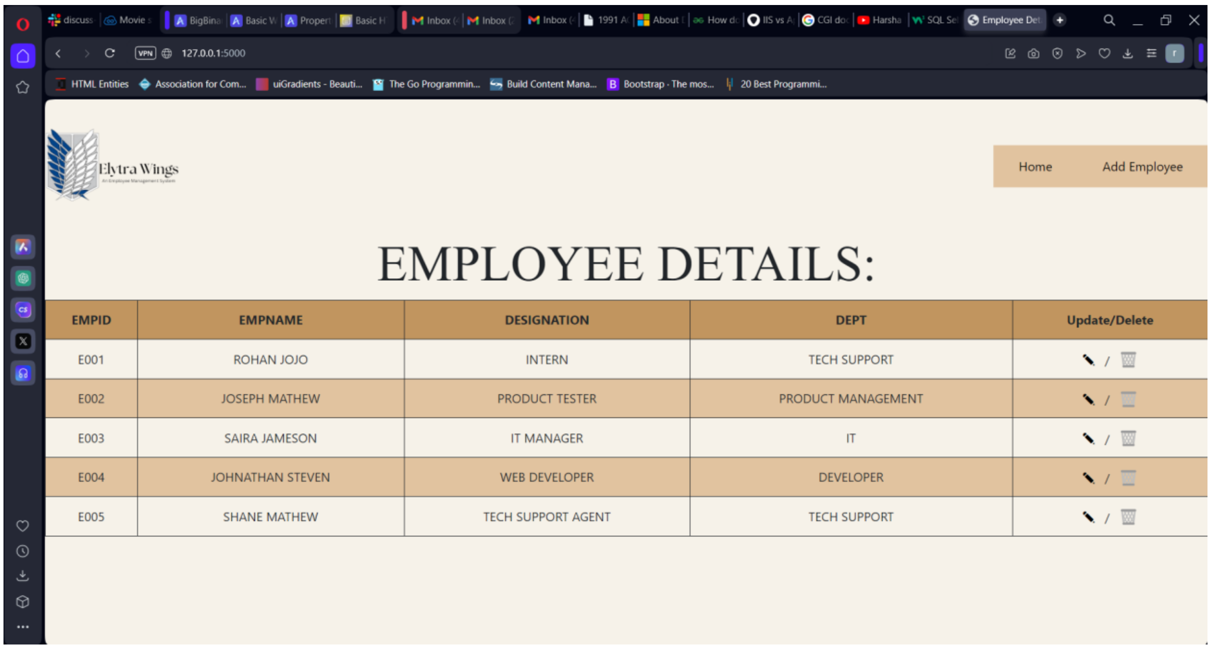


EDIT EMPLOYEE

EMPLOYEE NAME: SHANE MATHEW

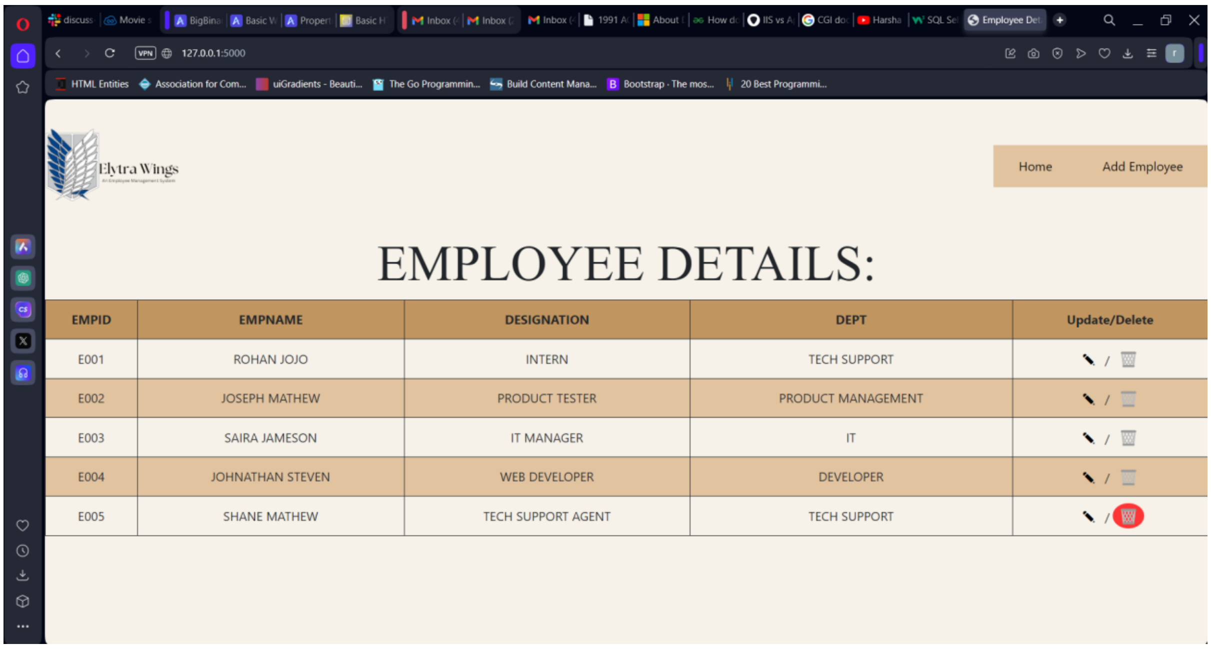
DESIGNATION: TECH SUPPORT AGENT

DEPT: tech support



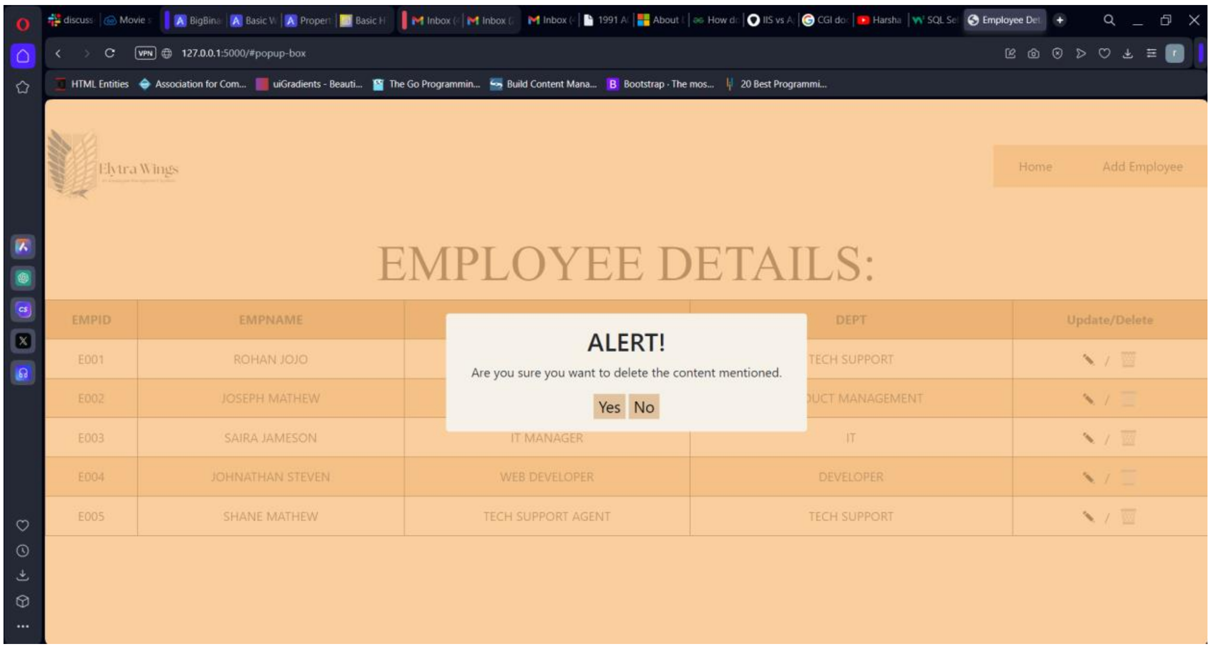
EMPLOYEE DETAILS:

EMPID	EMPNAME	DESIGNATION	DEPT	Update/Delete
E001	ROHAN JOJO	INTERN	TECH SUPPORT	/
E002	JOSEPH MATHEW	PRODUCT TESTER	PRODUCT MANAGEMENT	/
E003	SAIRA JAMESON	IT MANAGER	IT	/
E004	JOHNATHAN STEVEN	WEB DEVELOPER	DEVELOPER	/
E005	SHANE MATHEW	TECH SUPPORT AGENT	TECH SUPPORT	/



EMPLOYEE DETAILS:

EMPID	EMPNAME	DESIGNATION	DEPT	Update/Delete
E001	ROHAN JOJO	INTERN	TECH SUPPORT	/
E002	JOSEPH MATHEW	PRODUCT TESTER	PRODUCT MANAGEMENT	/
E003	SAIRA JAMESON	IT MANAGER	IT	/
E004	JOHNATHAN STEVEN	WEB DEVELOPER	DEVELOPER	/
E005	SHANE MATHEW	TECH SUPPORT AGENT	TECH SUPPORT	/



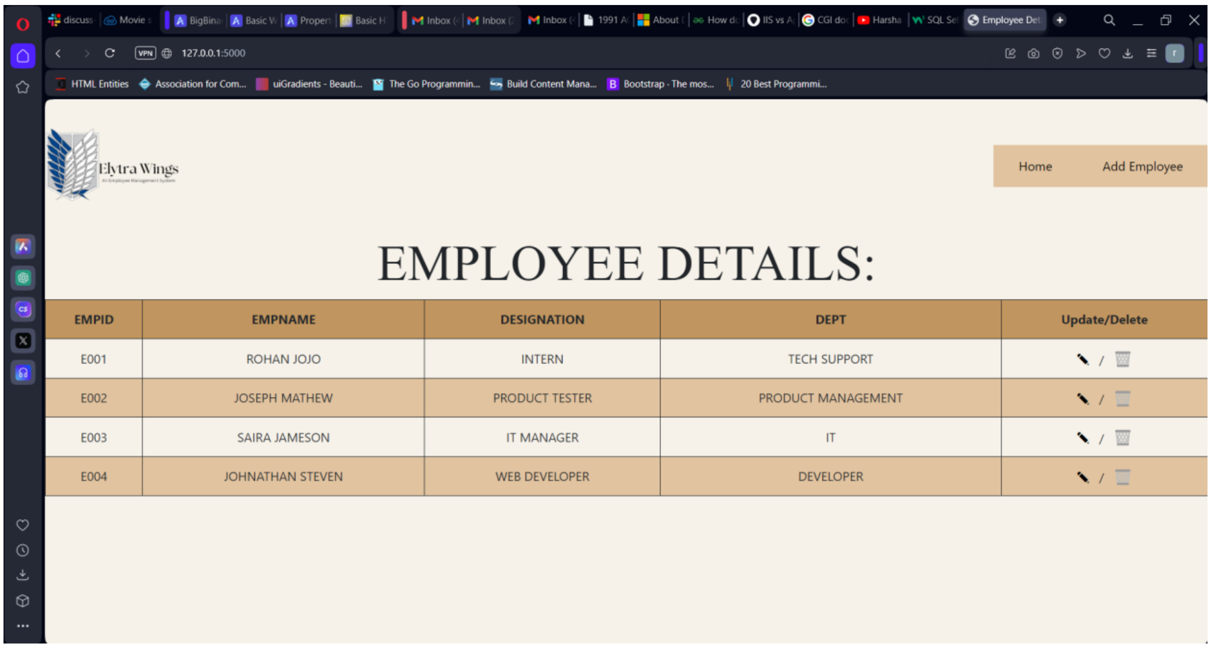
EMPLOYEE DETAILS:

ALERT!

Are you sure you want to delete the content mentioned.

Yes No

EMPID	EMPNAME		DEPT	Update/Delete
E001	ROHAN JOJO		TECH SUPPORT	/
E002	JOSEPH MATHEW		UCT MANAGEMENT	/
E003	SAIRA JAMESON	IT MANAGER	IT	/
E004	JOHNATHAN STEVEN	WEB DEVELOPER	DEVELOPER	/
E005	SHANE MATHEW	TECH SUPPORT AGENT	TECH SUPPORT	/



EMPLOYEE DETAILS:

EMPID	EMPNAME	DESIGNATION	DEPT	Update/Delete
E001	ROHAN JOJO	INTERN	TECH SUPPORT	✎ / 🗑
E002	JOSEPH MATHEW	PRODUCT TESTER	PRODUCT MANAGEMENT	✎ / 🗑
E003	SAIRA JAMESON	IT MANAGER	IT	✎ / 🗑
E004	JOHNATHAN STEVEN	WEB DEVELOPER	DEVELOPER	✎ / 🗑

REFERENCES:

https://howdns.works

https://www.browserstack.com/guide/what-is-browser

Definition of Functional Dependency from notes from class lecture section, Dr. Ray Hashemi, Database concepts, notes by Lance R. Olvey.

An Introduction to Database Systems, Addison-Wesley Publishing Company.

https://www.geeksforgeeks.org/types-of-functional-dependencies-in-dbms/

https://slideplayer.com/slide/7889701/

https://youtu.be/EGEwkad_llA

https://youtu.be/tkbAA--wKOc

https://youtu.be/mf_PbWPo7VM

https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/How_does_the_Internet_work

https://blog.hubspot.com/website/static-vs-dynamic-website

https://www.geeksforgeeks.org/difference-between-http-and-https/

https://www.geeksforgeeks.org/what-is-ssl-tls-handshake/

https://developer.mozilla.org/en-US/docs/Glossary/Thread https://developer.chrome.com/blog/inside-browser-part1

https://www.geeksforgeeks.org/difference-between-ram-and-cache/

https://www.geeksforgeeks.org/sram-full-form/

https://www.howtogeek.com/891526/l1-vs-l2-vs-l3-cache/

https://www.geeksforgeeks.org/difference-between-sram-and-dram/