Technical Guide to IT Analysts and IT Managers

# Version Information

Draft version. Alpha, 1.1 (Satya Komatineni)

Hope to take it to a Beta soon.

Started: 10/12/23

Expected completion: On going.

Contents

[Version Information 1](#_Toc148105666)

[Goal/Note 1](#_Toc148105667)

[Target Audience 2](#_Toc148105668)

[Lesson 1: Basic Anatomy of a Modern Computer application 2](#_Toc148105669)

[UI 2](#_Toc148105670)

[Example browsers 3](#_Toc148105671)

[Technologies used to develop UI – Simple Ones 3](#_Toc148105672)

[Technologies used to develop UI – Programmatic Ones 3](#_Toc148105673)

[Databases 3](#_Toc148105674)

[How to read and write data to databases: 4](#_Toc148105675)

[APIs 5](#_Toc148105676)

[Anatomy of an API 6](#_Toc148105677)

[Format and Transport of an API 7](#_Toc148105678)

[Terminology in lesson 1 8](#_Toc148105679)

# Goal/Note

1. IT Analyst is an interesting and important role in IT.
2. Interestingly, key skills are not technical. They are instead:
   1. Leadership
   2. Communication
   3. Fluency with languages like English, Spanish, etc.
3. Recruitment often asks for a bachelor’s in computer science or an IT degree. This is not needed in my opinion.
4. This guide will help you with the technical basics needed to be an effective Analyst (assuming you have the primary skills of leadership and communication).

# Target Audience

1. IT Analysts
2. IT Managers
3. IT Testers

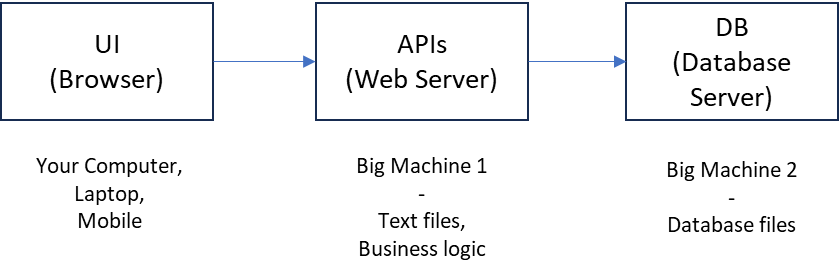
Suggestion: To get the best out of this document, make sure you have a comprehensive understanding of each of the items here. Ask someone enough questions until you are satisfied that you understand it. Yes. It is possible to do that even for a lay person with high-school level schooling. You can also use ChatGPT to further fill the gaps.

Hope to have more lessons as I find time, focusing on analysts. Some of them are

1. Reporting
2. ETL
3. Integration
4. Digital
5. AI/ML

# Lesson 1: Basic Anatomy of a Modern Computer application

Let’s start with a diagram.



There are 3 computers (Machines) involved in a modern application. Many modern applications are web-based internet applications.

With that, let me explain the parts of this application and the terminology used.

## UI

This is often referred to with the following names:

1. User interface
2. Front end
3. UX
4. User Experience

In a web application this part is what you see in the browser. The text, the documents, images, forms, buttons etc.

All this information that you see in the browser is kept on the web server or “machine 1” in the picture above.

However, the information that you enter is kept in the database server on machine 2. This information you entered is validated and cleaned up by the business logic on machine 1 (web server) before storing in the database on machine 2.

### Example browsers

1. Chrome
2. Ie
3. Mozilla
4. Safari
5. Etc.

### Technologies used to develop UI – Simple Ones

These technologies below are basic technologies developed originally for the web to display documents and files.

1. HTML
2. XML

Read up on ChatGPT what these are and what the files look like when displayed on a website.

Everything you see on the web page is finally translated to one of these two formats before displaying.

### Technologies used to develop UI – Programmatic Ones

When what is displayed is conditional and when the website is more complex like a Facebook or a Google, the HTML is finally generated using an underlying programming language coded in JavaScript.

Some popular libraries (or technologies) used to accomplish this are:

1. JavaScript
2. Angular
3. React

## Databases

Databases are where collected data is stored. This data is usually in rows and columns, similar to a spreadsheet. These are usually called tables.

Typically, the machines where these databases are located and managed are called database servers.

There are many types of databases with different capabilities.

Historically the most used type of database is called a “relational” database. Most databases fall into this category. In this type of database data is stored like a spreadsheet where each “tab” is called a “table”.

For a few years now there is a new kind of database type called “Document” based database. In this kind, data is stored as a document like MS word, where all related data is stored as a “chunk”.

Another approach to classification is whether the database is shared among multiple organizations and available in the cloud.

Few well known databases in the market are:

1. Relational databases
   1. MySQL
   2. **Postgres**
   3. **MS SQL Server**
   4. Oracle
   5. IBM DB2
2. Cloud data bases
   1. Google Firebase
   2. Supabase (Based on Postgres)
3. Document database
   1. Mongo
4. ..and many, many, more

### How to read and write data to databases:

As an analyst first need is how to see what is in the database. This is sometimes referred to as “querying a database”.

All databases come with an application where one can browse through the data. In relational databases such a tool is used for examining the various tables available in the database. For example, “parts” table in an inventory database will have many rows of parts where each describing what that part 17 is: its name, its manufacturer, its cost etc.

In addition to the user interface (UI), the tools allow us to use a language called “SQL” to examine and query the subsets of data based on any given criteria. This language also allows to insert data or modify tables and columns.

As an analyst you are expected to understand the following three topics in a fair amount of detail:

1. Applications that allow you to see and administer databases
2. SQL
3. Stored procedures

We already talked about 1 and 2. The 3rd one “stored procedures” is an advanced concept. These are a set of procedures or functions that reside in the same database that “processes” a collection of rows using some business logic to filter the rows or aggregate the rows based on business logic.

These “stored procedures” have commonalities with APIs (which we will discuss later). These are business rules that interprets the data in a database.

For example, consider the following stored procedures in a parts database:

1. getPartsFrom(“GM”) – This procedure will return all the parts manufactured by GM
2. getPartsFrom(“Ford”) – The same procedure will return all parts manufactured by Ford
3. insertPart(“Engine 5”,”GM”) – This procedure will insert a row in to the parts table with the part name as “Engine 5” and the manufacturer as “GM”

1. create procedure getPartsFrom(companyName)

2. {

3. return

4. “select \* from parts-table

5. where manufacturer = companyName”

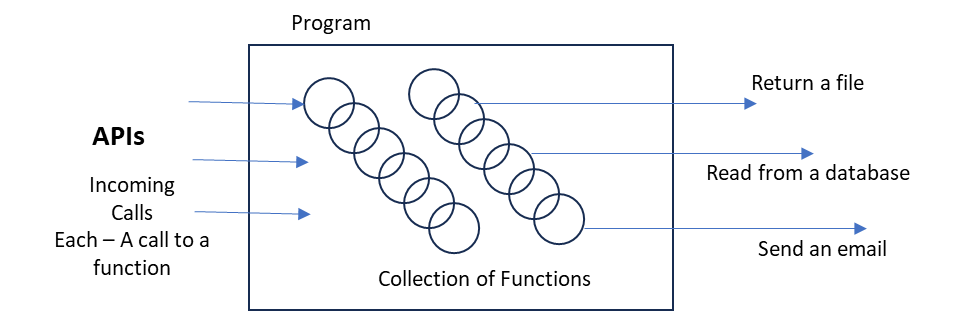
6. }

This is an example of how one will create a stored procedure. Here the code that starts with “select…” is in the SQL language. (Lines 4 and 5)

Based on the database this method and syntax of creating a stored procedure will differ. However the SQL (lines 4 and 5) typically don’t change

## APIs

Let’s cover now what are web servers and APIs.



A web server is a “program”. It runs on a dedicated machine. Many times on multiple machines.

A program is a collection of functions. Each function does something that is unique. Such as reading and returning a file, sending an email, updating a database etc.

These functions are written very similarly to the stored procedure you have seen before. It is a set of instructions written in a computer language.

There are many computer languages.

They all do the same thing. Write functions that does something.

Many times the language and the functions are preprogrammed and available as downloadable packages to developers (also called programmers). These preprogrammed packaged functions are called libraries.

Developers use those languages and the libraries to write these custom functions that make up a website logic and behavior.

Once the program is written in a language it is then installed on a computer and continues to run all the time. That is why this is called a “server” ready to serve anyone that is authorized and calls.

Typically it is the browser that you are using that is the caller.

Browser calls this website, program, by the address of the website and the “name of a function”.

These exposed names of functions are called “APIs”.

API stands for “Application Programming Interface”.

So APIs are basically functions written in a particular language and available for clients like the Browser to call.

Some popular languages to make these APIs are

1. Java
2. C#
3. Python
4. JavaScript
5. Rust
6. Go
7. ….many, many more

### Anatomy of an API

The following are the parts of an API

1. Name
2. Input arguments
3. Output

Ex:

1. SendMailAPI

2. {

3. Name: SendMailAPI

4. Input Arguments

5. {

6. To: Set of recipients

7. Subject: A text

8. Body: A long text

9. }

10. Output

11. {

12. SuccessOrFailureFlag: Yes or No

13. }

14. }

15.

Each language has its own syntax to specify the API.

In many languages this looks more like

1. Boolean SendMailAPI(to, subject, body)

The word “Boolean” means that this API returns a “true” or “False”.

### Format and Transport of an API

When a browser (which is your laptop) makes a call to one of these APIs on a web server, data is transmitted from your laptop to the Server.

This communication must follow one of several standards that is mutually agreed upon by the browser and the server.

There are a number of formats and protocols that have been used by the industry over time.

The most recent protocol is called “REST”.

This is why APIs are sometimes called REST APIs.

The information transmitted can be XML or HTML on top of the underlying network. Typically the information transmitted via a REST API is called JSON.

JSON is very similar to HTML and XML and it is in a shape that is easier to convert to a “language” that the server is written in.

Let’s look at how a message looks like in each of HTML, XML, and JSON

First in HTML

1. <html>

2. <head>

3. <title>Structured Message Example</title>

4. </head>

5. <body>

6. <h1>Structured Message</h1>

7. <p>This is a simple structured message represented in HTML.</p>

8. <ul>

9. <li>Name: John Doe</li>

10. <li>Email: johndoe@example.com</li>

11. <li>Message: Hello, World!</li>

12. </ul>

13. </body>

14. </html>

15.

Now in XML

1. <message>

2. <name>John Doe</name>

3. <email>johndoe@example.com</email>

4. <content>Hello, World!</content>

5. </message>

Now in JSON

1. {

2. "message": {

3. "name": "John Doe",

4. "email": "johndoe@example.com",

5. "content": "Hello, World!"

6. }

7. }

They are all the same. Just the way these are received by the web server depends on the language the server is written in.

Typically with REST JSON has been the preferred format.

When developers talk about APIs, REST APIs, JSON, this is what they are talking about.

# Terminology in lesson 1

1. UI
2. UX
3. User Interface
4. API
5. Input Arguments
6. Output Arguments
7. Database
8. SQL
9. Stored Procedure
10. Web server
11. Database Server
12. URL
13. HTML
14. XML
15. JSON
16. REST

Feedback/Note: If you have questions to strengthen your understanding reach out to me on LinkedIn.

https://www.linkedin.com/in/satyakomatineni/