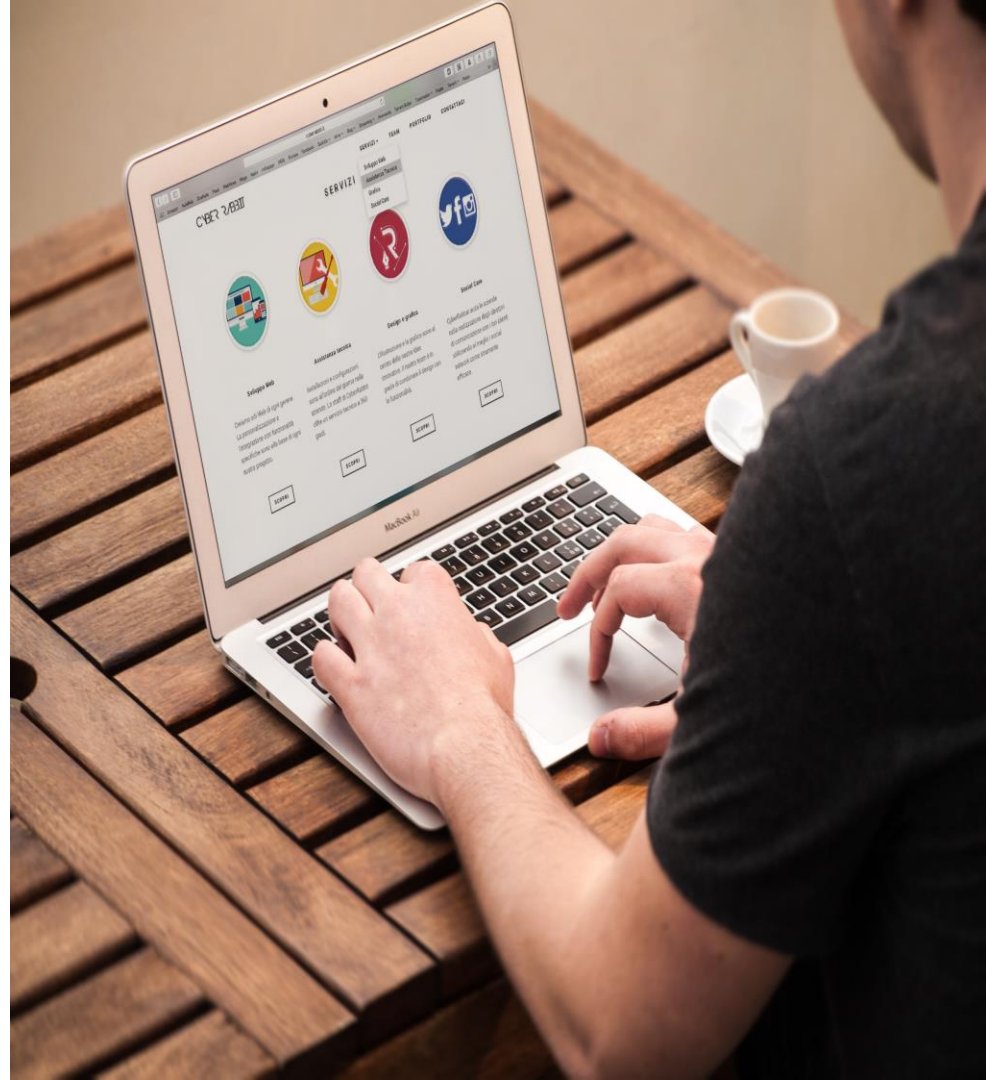


# DEVELOPMENT OF DYNAMIC WEBPAGE

## Introduction to server-side scripting



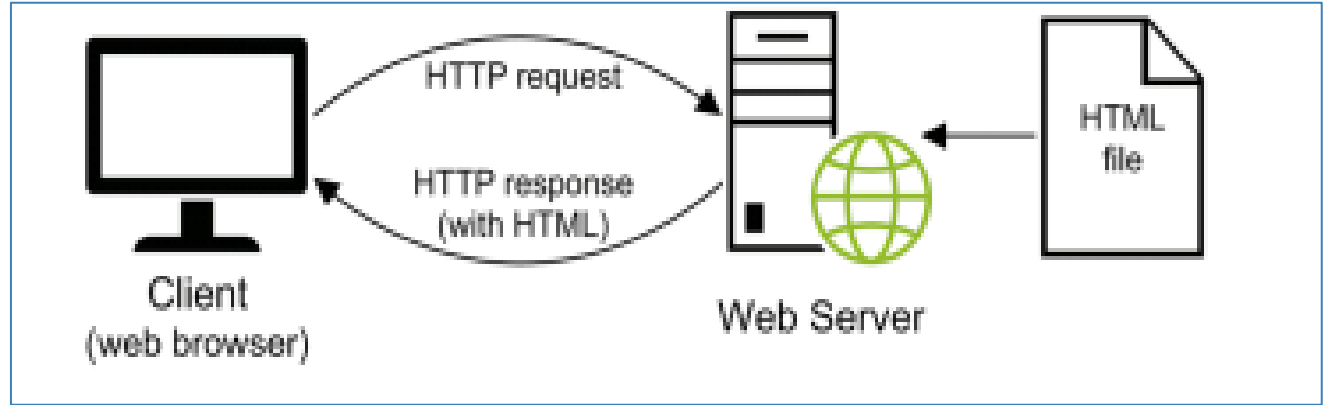
# What is Dynamic website design ?

*A dynamic site content can change as per requirements provided by the computer program or the users.*

*The dynamic page may change with the time or as per user who use the site.*

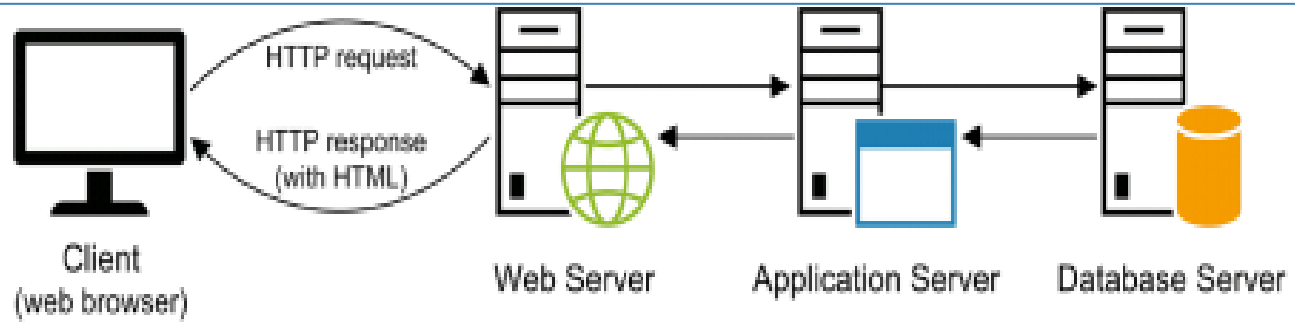
*Two types of Dynamic web pages that is Client side scripting and Server side scripting.*

# How static page is loaded?



1. You will add the url of the website
2. It will look for files
  - a. HTML
  - b. CSS
  - c. JS
3. It will return it back to the user's browser

# How dynamic site is loaded?



You will enter the url of the website in browser

2. It will look for files
  - a) HTML
  - b) CSS
  - c) JS
3. Inside the HTML file, it might call/retrieve data from the **database** through an application server
4. Once the data is retrieved from the database, the web server will return the website and required information to the user's browser

# Static website vs Dynamic website?

## The main differences between a static vs dynamic website

Content on a static website is stable and doesn't change. Content on a dynamic website can change according to how you want it to behave and what you want specific users to see.

Content on a static website is stored directly on the server and pulled as is. Content on a dynamic website is stored in a database or collection and delivered according to how it is organized or filtered.

Content changes on a static website need to be made page by page, on a dynamic website they can be made across hundreds of pages automatically.

A dynamic site can have its content displayed according to how a user interacts with the site, it can also have input from a user. This functionality is more limited with a static website.

Dynamic websites may take longer to initially setup but long term they can be more efficient to manage. Static websites conversely can be created fast but as they grow will require more intensive content management.

## Language to build a dynamic website

▪ *To develop a dynamic website, on top of HTML.,, CSS and Javascript you need to use a server side Scripting language*

▪ ***Example of Server side scripting language includes:***

▪ ***Python***

▪ ***PHP***

▪ ***Perl***

▪ ***.Net***

▪ ***Java***

▪ ***etc***

## Advantage of dynamic website?

*There are lots of the advantage of Dynamic website ...*

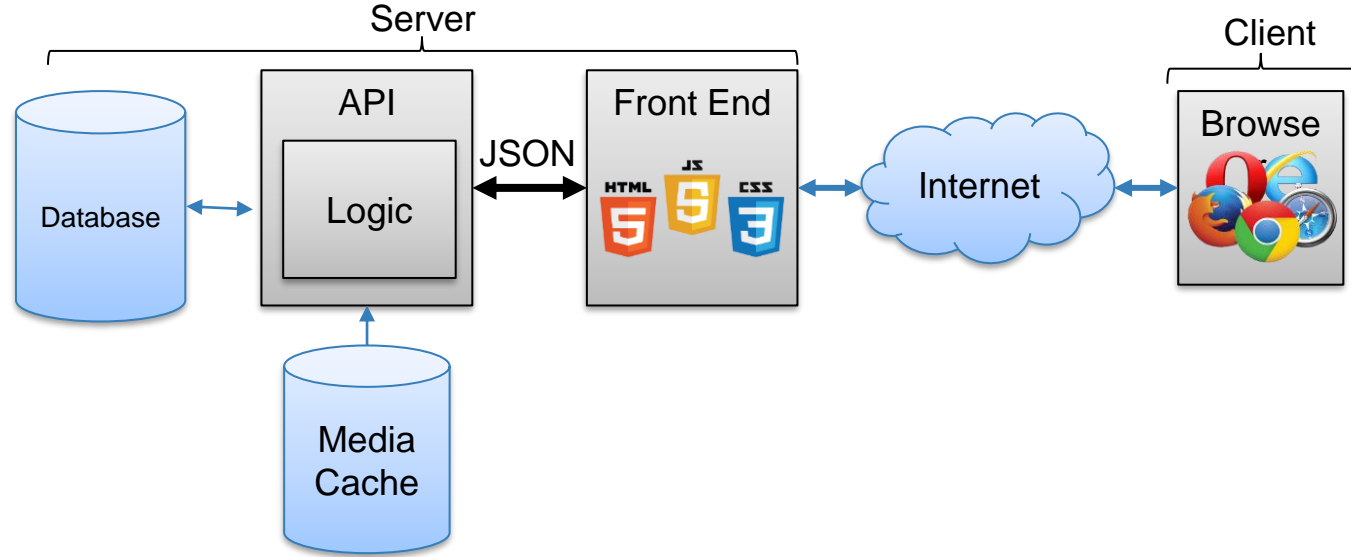
- *Data can be updated from Content Management System or from database*
- *Can be updated easily. For example, update catalog price, blog and news, announcement, pictures etc.*
- *Can be integrated with other system to create a full fledge engine: eg, payment provider to create an e-commerce or booking system.*

# List of static website & Dynamic website

STATIC	DYNAMIC
government.github.com	Youtube.com
christinavanessa.com	netflix
	Wikipedia.org



# Core Components of Web Applications



- *UI (Front End (DOM, Framework))*
- *Request Layer (Web API)*
- *Back End (Database, Logic)*

# What is a Web Framework?

- A Web Framework is designed to facilitate building a dynamic website easily.
- In a web framework, there are some concepts and architecture need to be adhered to:
  - Model-View-Controller (MVC)
  - Object Relational Modelling (ORM)
  - Routing

<https://shopee.com.my/search?keyword=exercise%20stupper>

<https://shopee.com.my/search?keyword=badminton%20onet%20portable>






<https://shopee.com.my/m/pasti-murah>

<https://shopee.com.my/50pcs-Full-Black-colour-3ply-kf94-Face-mask-Non-Medical-Disposable-Mask-full-Black-Face-Mask-Hitam-i.327979601.7362431580>

# Framework & Libraries



- *A framework normally linked to a particular language:*

Libraries & Framework	Language
 django	Framework using Python
 Spring Boot	Framework using Java
 Laravel	Framework using PHP
 RAILS	Framework using Ruby
 Flask	Framework using Python

*However, you may see the same principle in different languages*

# Types of Web Framework?

- *In web, there are two types of Frameworks*
  - *Server Side: Django, Ruby on Rails*
  - *Client Side: Angular, React, Vue*

*We will focus on Server Side framework, for Website Application Development (WAD)*

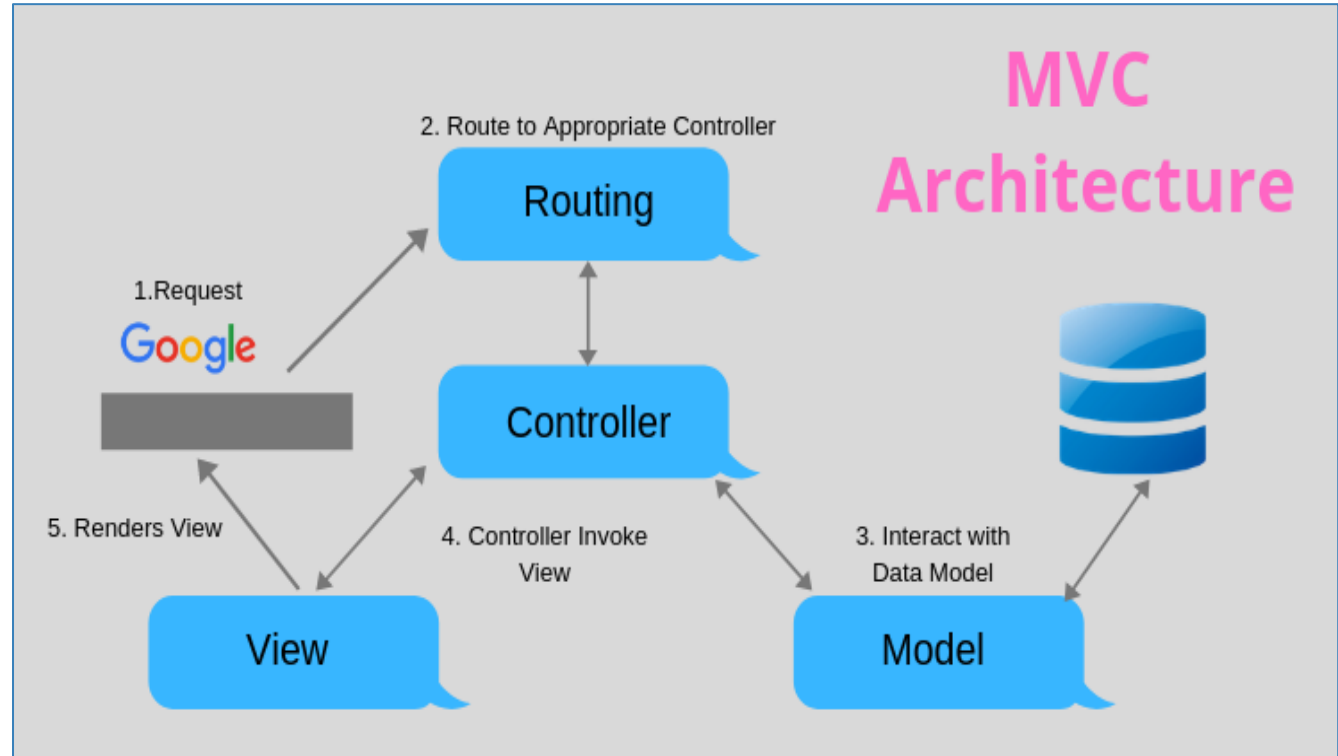
# What is MVC (Model View Controller)

- *A Web Application Development Framework*
- *Model (M):*
  - *Part of the web application that retrieve data from the database.*
  - *This is normally written in backend programming language using ORM principle*
- *View (V):*
  - *Think of the UI Representation of a website*
  - *You will normally see HTML, CSS and even JS here (WAD 1)*
  - *Some framework, eg Django will have server rendering language.*
- *Controller (C):*
  - *Handle the logic of our application,*
  - *It will link the UI (View) to the database (Model)*
  - *It also perform form handling, authentication, validation, integration with other application etc.*

# What is MVT(Model View Templates) in **django**

- *A Web Application Development Framework*
- *Model (M) (models.py):*
  - *Part of the web application that retrieve data from the database.*
  - *This is normally written in backend programming language using ORM principle*
- *View (V): - (views.py) -controller*
  - *Handle the logic of our application,*
  - *It will link the UI (View) to the database (Model)*
  - *It also perform form handling, authentication, validation, integration with other application etc.*
- *Templates (T) (templates folder) -view*
  - *Think of the User Interface Representation of a website*
  - *You will normally see HTML, CSS and even JS here (WAD 1)*
  - *Some framework, eg Django will have server rendering language.*

# MVC Model

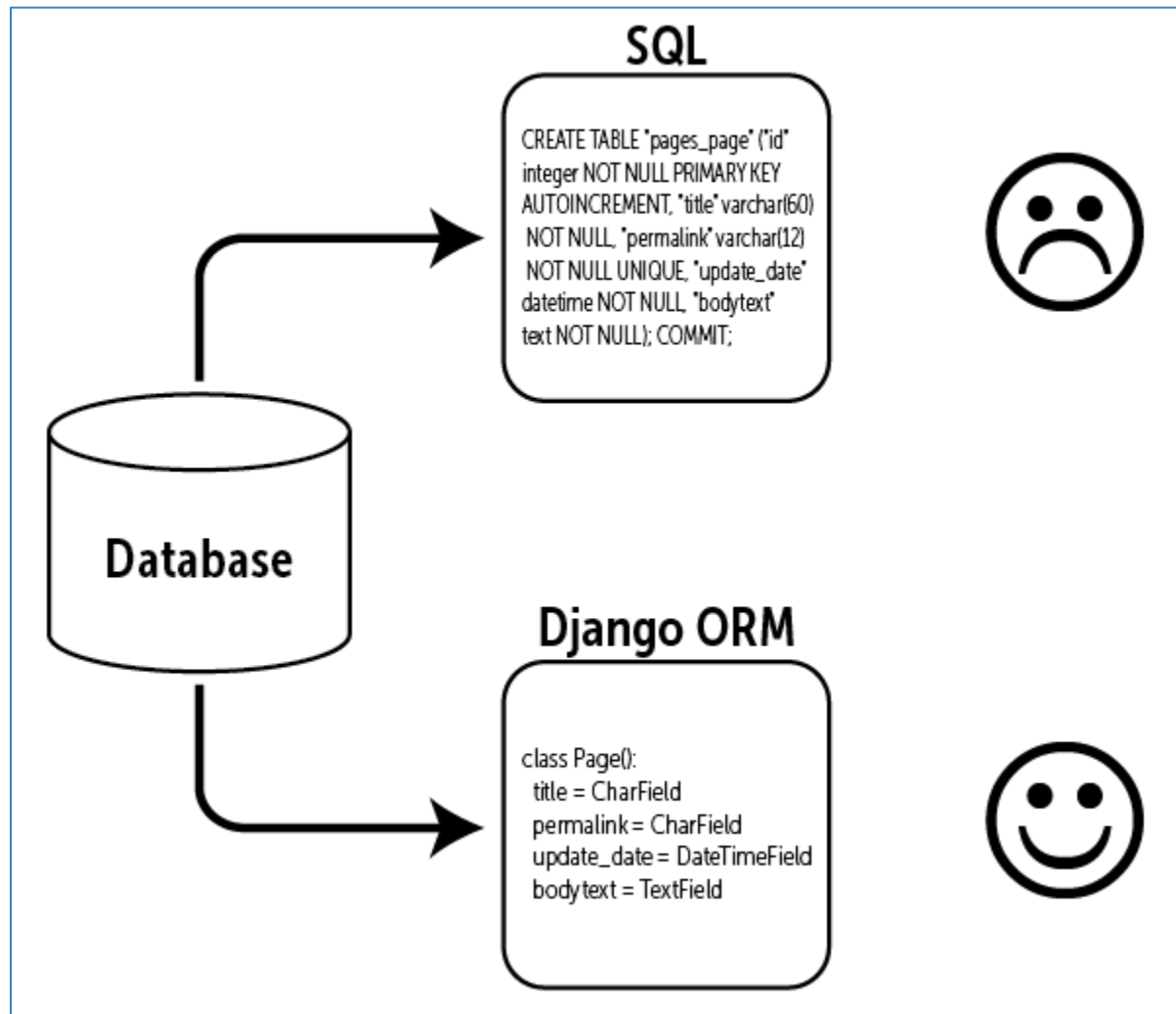


# What is Object Relational Modelling?

- **Object-Relational Mapping (ORM)** is a technique that lets you query and manipulate data from a database using an object-oriented paradigm.
- In ORM :
  - A table in database is represented as a Class
  - A column in a database is represented as a property
  - A row in a database is represented as an Object
  - A query in a database is represented as a method
- Most of modern web framework use ORM principle, including Django



# ORM Model



# Installation

Create new folder S3WAD

Cd into S3WAD (open command prompt from the folder created)

python --version -> make sure that student is using python3

Install django - pip3 install django / pip install django

## Start a new django project

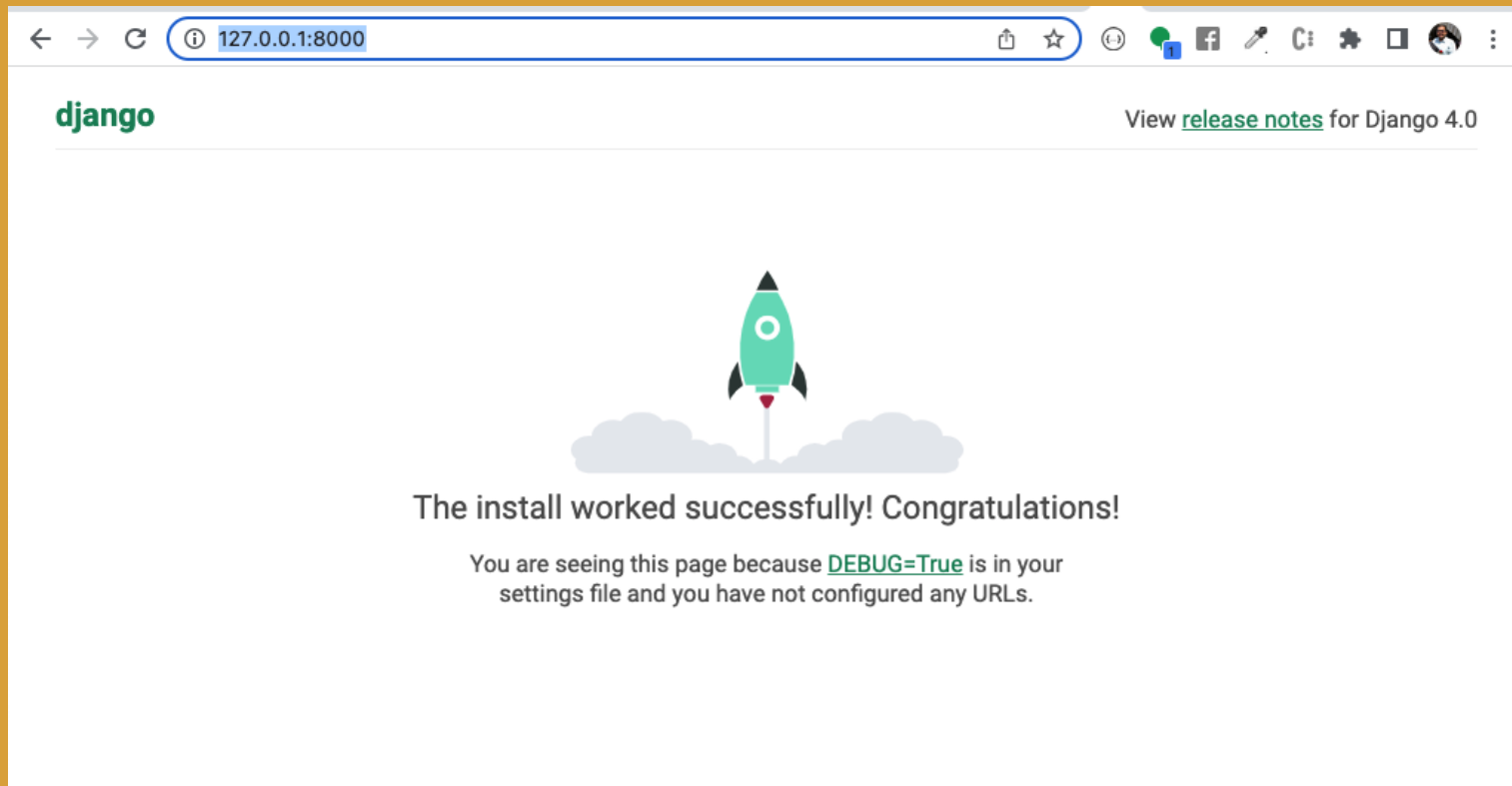
- 1) `django-admin startproject hello_django` -> Create a new project, project name hello\_django
- 2) `cd hello_django` -> Go inside hello\_django folder
- 1) `python manage.py runserver` -> run the server

# Open browser

```
June 15, 2022 - 06:48:48  
Django version 4.0.5, using settings 'hello_django.settings'  
Starting development server at http://127.0.0.1:8000/  
Quit the server with CONTROL-C.
```

Open this url on browser 127.0.0.1:8000

This page will come out if everything is ok



# HTTP

- *To communicate with the server, we use HTTP Protocol*
- *Invented for the Web - to retrieve HTML, Images, Documents, etc.*
- *Extended to handle data in addition to documents - RSS, Web Services, etc.*
- *Basic Concept: Make a connection - Request a document - Retrieve the document - Close the connection*
- *We can also communicate with the server to:*
- *Add new data in the database (Send data from a form) – CREATE (method: POST)*
- *Retrieve Data from the database – READ (method: GET)*
- *Update Information from the database from a form – UPDATE (method: POST)*
- *Delete Data from the database – Delete (method: DELETE)*
- *This operation is normally known as CRUD*

# Routing in URL in web

*What is Database operation?*

<b>METHOD</b>	<b>Database Operation</b>
POST	Create (collection)
GET	Read
POST/PUT	Update
DELETE	Delete

# Website Status Codes

<i><b>Status Code</b></i>	<i><b>Description</b></i>
200	<i>OK – things are great (return the item)</i>
201	<i>Created – after POST (HATEOAS – return location)</i>
204	<i>No Content (i.e. successful DELETE)</i>
400	<i>Bad Request (validation error, missing parms, etc.)</i>
401	<i>Unauthorized – Who are you?</i>
403	<i>Forbidden – No soup for you</i>
404	<i>Not Found</i>

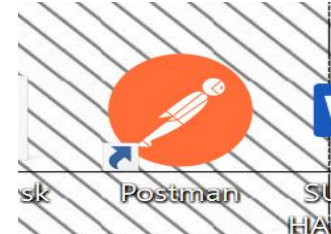
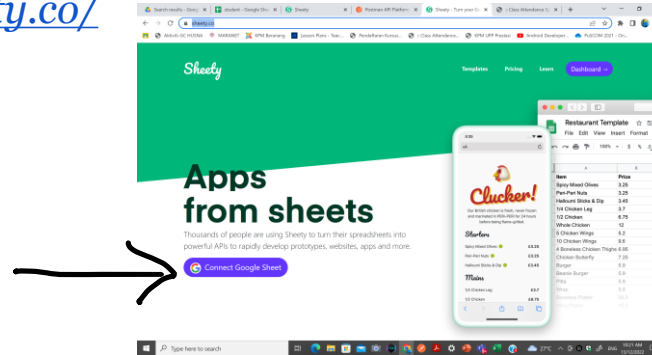


# Database operation

Database Operation	METHOD	url	Description	Need body or not?
Create	POST	/books	=add new data for any books.	YES
Read	GET	/books /books/1	= get all books = get a book of id 1	NO
Update	POST/PUT	/books/1	= update a book of id 1	YES
Delete	DELETE	/books/1	= delete a book of id 1	NO

# Database operation (POSTMAN ACTIVITY)

1. Create a google sheet from in Google Drive  
-Class Student  
-Property : studentid, studentname, studentprog →
2. Sign in and download postman (<https://www.postman.com/>)
3. <https://sheety.co/>



1. Paste link from google sheet in Google Drive
2. copy

# Database operation (POSTMAN ACTIVITY)

*Paste link from  
google sheet in  
Google Drive*

*Create project*

The screenshot shows the 'Create a new project' interface in the Postman API Platform. At the top, there are browser tabs for 'google Sh...', 'Sheety', 'Postman API Platform', 'Sheety', and 'Class Attendance S...'. Below the tabs, a navigation bar contains links for 'KPM Beranang', 'Lesson Plans - Teac...', 'Pendidikatan Kursus...', 'Class Attendance...', 'KPM UPP Prestasi', and 'Android Developer'. The main heading is 'Create a new project', followed by a description: 'A project represents a single spreadsheet in your Google Drive. The sheets within that spreadsheet will become individual API endpoints.' A text input field contains a Google Drive URL: 'https://docs.google.com/spreadsheets/d/1cUq4QKE77cblDr7G036KDdpKSofWIAp8Cq-p6Rx\_Jk/edit'. Below this, there is a table for project configuration:

Project Name	student
Sheet	URL Preview
Sheet1	/student/sheet1

Below the table, a note states: 'URLs are automatically generated from the names of your sheets. Rename or add sheets inside the spreadsheet to change the URLs. You can edit these later too. [Learn more](#) <>'. A green 'Create Project' button is highlighted with a hand-drawn circle. At the bottom, there are sections for 'About' (Pricing, Privacy Policy, Twitter @getpostman), 'Learn' (Getting Started, Making Requests, Templates), and 'Help' (Troubleshooting, Billing questions, Status). The Windows taskbar is visible at the very bottom.

# Database operation (POSTMAN ACTIVITY)

The screenshot shows the Sheety dashboard for a project named 'student'. The dashboard has tabs for 'API', 'Authentication', and 'Settings'. Under the 'API' tab, there is a section for '1 sheets' with a 'Refresh' button. Below this, there is a 'Sheet1' section with a description: 'To add or edit sheets, do so in the spreadsheet itself then push "Refresh" here. Learn more →'. To the right of the 'Sheet1' section, there is a list of API endpoints with their status: 'GET Retrieve rows from your sheet' (Disabled), 'POST Add a row to your sheet' (Disabled), 'PUT Edit a row in your sheet' (Disabled), and 'DELETE Delete a row in your sheet' (Disabled). The bottom of the dashboard has links for 'About', 'Learn', and 'Help'.

Search results - Google Drive X student - Google Sheets X Sheety X Postman API Platform | Sign X : Class Attendance System : X + -

dashboard.sheety.co/projects/6397d44c3129727edc02f39/sheets/sheet1

Aktiviti-GC HUSNA MARANET KPM Beranang Lesson Plans - Teac... Pendaftaran Kursus... : Class Attendance... KPM UPP Prestasi Android Developer... PuSCOM 2021 - On...

Sheety Projects Templates Usage Upgrade Learn nurhannanie01@gmail.com

## student

Open Spreadsheet →

API Authentication Settings

1 sheets Refresh

Sheet1

To add or edit sheets, do so in the spreadsheet itself then push "Refresh" here. Learn more →

### Sheet1

Enable or disable specific behaviours for this sheet.

- GET Retrieve rows from your sheet Disabled
- POST Add a row to your sheet Disabled
- PUT Edit a row in your sheet Disabled
- DELETE Delete a row in your sheet Disabled

Sheety © 2022

About Pricing Privacy Policy Twitter (@GetSheety)

Learn Getting Started Making Requests Templates

Help Troubleshooting Billing questions Status

This screenshot is a zoomed-in view of the 'POST Add a row to your sheet' endpoint. The status is 'Enabled'. Below the endpoint name, there is a text input field containing the URL: 'https://api.sheety.co/1a73274695f4ed0f0442c7c068315141/student/sheet1'. Below the URL field, there is a 'JavaScript Example' section with a code block. The code block contains a JavaScript snippet for making a POST request to the Sheety API. Below the code block, there are two more endpoints: 'PUT Edit a row in your sheet' (Disabled) and 'DELETE Delete a row in your sheet' (Disabled).

Sheet1

To add or edit sheets, do so in the spreadsheet itself then push "Refresh" here. Learn more →

### Sheet1

Enable or disable specific behaviours for this sheet.

- GET Retrieve rows from your sheet Disabled
- POST Add a row to your sheet Enabled
- PUT Edit a row in your sheet Disabled
- DELETE Delete a row in your sheet Disabled

JavaScript Example

```
let url = "https://api.sheety.co/1a73274695f4ed0f0442c7c068315141/student/sheet1";
let body = {
  sheet1: {
    // ...
  }
}
fetch(url, {
  method: "POST",
  body: JSON.stringify(body)
})
.then(response => response.json())
.then(json => {
  // Do something with object
  console.log(json.sheet1);
});
```

# Database operation (POSTMAN ACTIVITY)

The screenshot displays the Sheety API dashboard in a web browser. The browser's address bar shows the URL: `dashboard.sheety.co/projects/6397d44c3129727edc022f39/sheets/sheet1`. The dashboard includes a green button labeled "Sheet1" and a note: "To add or edit sheets, do so in the spreadsheet itself then push 'Refresh' here. [Learn more](#) →".

Below this, a section titled "Enable or disable specific behaviours for this sheet." contains four toggle switches for different API methods:

- GET Retrieve rows from your sheet**: Disabled (toggle switch is off).
- POST Add a row to your sheet**: Enabled (toggle switch is on).
- PUT Edit a row in your sheet**: Disabled (toggle switch is off).
- DELETE Delete a row in your sheet**: Disabled (toggle switch is off).

The "POST Add a row to your sheet" section is expanded, showing a text input field with the URL: `https://api.sheety.co/1af3274eb954e8dfd4a25e2f06835343/student/sheet1`. Below the input field is a "Javascript Example" section containing the following code:

```
let url = 'https://api.sheety.co/1af3274eb954e8dfd4a25e2f06835343/student/sheet1';
let body = {
  sheet1: {
    ...
  }
}
fetch(url, {
  method: 'POST',
  body: JSON.stringify(body)
})
.then((response) => response.json())
.then(json => {
  // Do something with object
  console.log(json.sheet1);
});
```

The bottom of the image shows a Windows taskbar with various application icons and a system clock indicating 10:28 AM on 13/12/2022.

*METHOD : POST (BODY)*

```
{  
  "sheet1":{  
    "studentid": "bcs2211099",  
    "studentname": "SARIMAH",  
    "studentmentor": "MAZURA"  
  }  
}
```

# Database operation (POSTMAN ACTIVITY)

1. Create a google sheet from in Google Drive
  - Class Student
  - Property : studentid, studentname, studentprog
2. Sign in and download postman  
(<https://www.postman.com/>)
3. <https://sheety.co/>
4. Paste link from google sheet in Google Drive
5. Create project
6. Open postman app
7. Enable method from sheety.co
8. Copy & paste link generated from sheety to postman
9. In postman, choose body & JSON to create code to post and edit google sheet.
10. Click send.



# student1

[Open Spreadsheet →](#)

API

Authentication

Settings

1 sheets

Refresh

## Sheet1

Enable or disable specific behaviours for this sheet.

Sheet1

To add or edit sheets, do so in the spreadsheet itself then push "Refresh" here. [Learn more →](#)

**GET** Retrieve rows from your sheet

☒ Enabled

Home Workspaces API Network Explore

Search Postman

Invite



Upgrade



My Workspace

New

Import

Overview

DEL <https://api.sheety.co/1af3274eb954e8dfd4a25e2f06835343/student/sheet1/2>

No Environment

Collections



Environments



Mock Servers



Monitors



Flows



History

My first collection

- First folder inside collection
  - GET
  - POST
  - GET
- Second folder inside collection
  - GET
  - GET

### Create a collection for your requests

A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.

Create Collection

DELETE

GET

POST

PUT

PATCH

DELETE

COPY

HEAD

OPTIONS

LINK

UNLINK

PURGE

LOCK

UNLOCK

<https://api.sheety.co/1af3274eb954e8dfd4a25e2f06835343/student/sheet1/2>

Save



</>

Send

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

VALUE

DESCRIPTION

...

Bulk Edit

Value

Description



# Path and parameter

- *Parameter ->*

<https://www.example.com/index.html?name1=value1&name2=value2>

- *GET parameters always start with a question mark ? This is followed by the name of the variable and the corresponding value, separated by an =. If an URL contains more than one*
- *parameter, they are separated by an Ampersand &.*
  - *Example, it is used to filter content:*  
*Filtering content: ?type=green displays only green products on an e-commerce site.*
  - *Sorting contents: ?sort=price\_ascending sorts the displayed products by price, in this case ascending.*

# Practice exercise

▪ <https://api.openweathermap.org/data/2.5/weather?q=Dubai&apiKey=9fd7a449d055dba26a982a3220f32aa2>

▪ *What are the parameters in this URL?*  
[?q=Dubai&apiKey=9fd7a449d055dba26a982a3220f32aa2](https://api.openweathermap.org/data/2.5/weather?q=Dubai&apiKey=9fd7a449d055dba26a982a3220f32aa2)

▪ *Which parameter save the value of city name?*

q

▪ *Change the value of city name to Kuala Lumpur? What are the difference between temperature in Dubai and KL?*  
KL- 303.45    Dubai-310.92

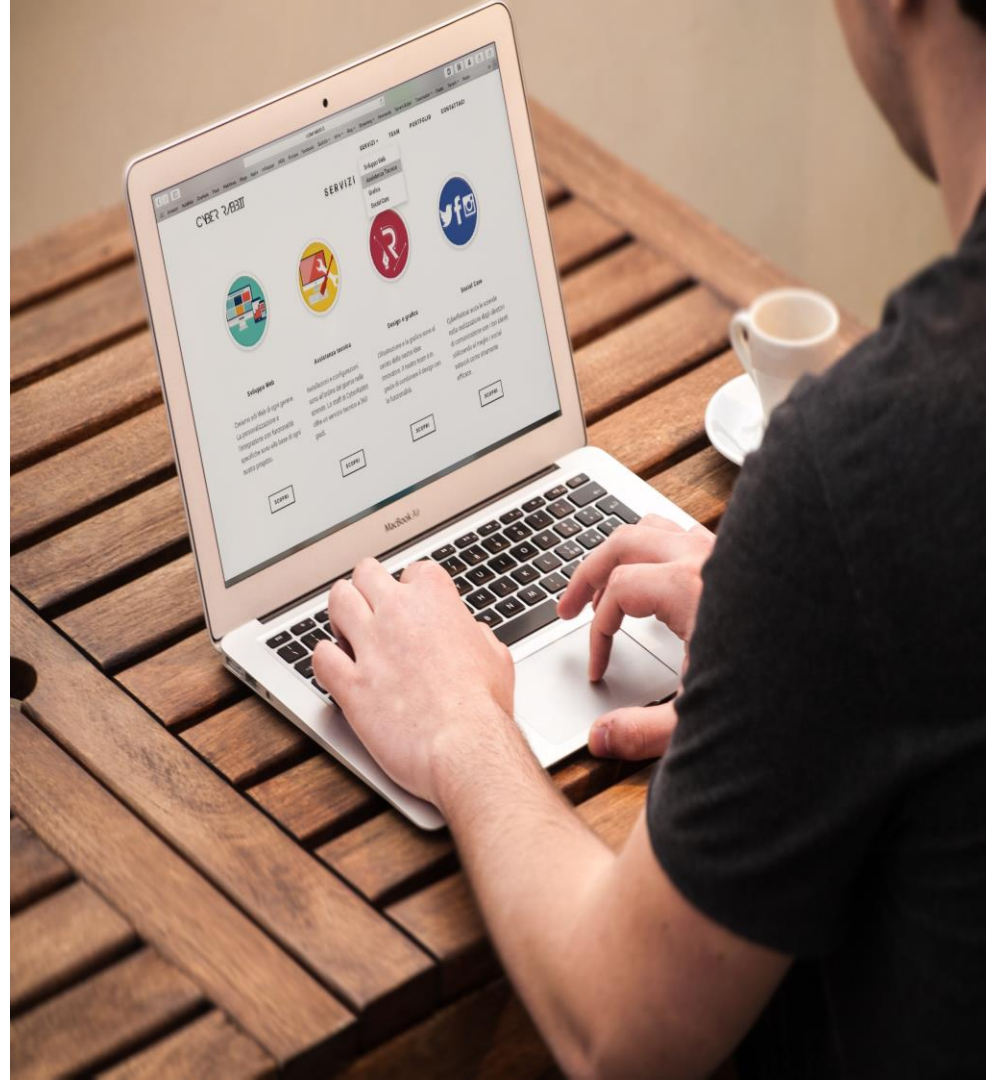
# Benefits of Dynamic website

*There are lots of the advantage of Dynamic website ...*

- *Data can be updated from Content Management System or from database*
- *Can be updated easily. For example, update catalog price, blog and news, announcement, pictures etc.*
- *Can be integrated with other system to create a full fledge engine: eg, payment provider to create an e-commerce or booking system.*

# DEVELOPMENT OF DYNAMIC WEBPAGE

## Connecting the Front & Backend



# What is a Backend?

- *All of the awesome that runs your application.*
- *Web API*
  - *Connection layer between the frontend and backend*
  - *Connected through API calls (POST, GET, PUT, etc. )*
  - *Transmit Content from the Backend to the Frontend commonly in JSON Blobs*
- *Service Architecture that drives everything (Where all the logic is)*

# What is a WebAPI?

*The intermediate layer between front end and back-end systems. A “must have” if your APIs will be consumed by third-party services*

- *Attention to details:*

- *How consumable is the API (signature, content negotiation)?*

- *Does it comply with standards (response codes, etc.)?*

- *Is it secure?*

- *How do you handle multiple versions?*

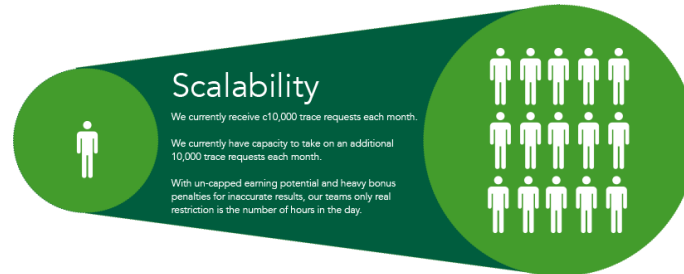
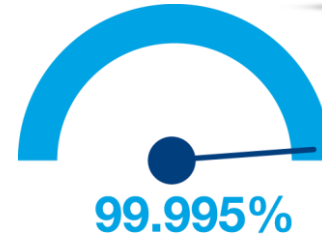
- *Is it truly RESTful? (Representational State Transfer)*

*<http://kpmb.com/API/books>*

# Principles of Web Design

1. Availability
2. Performance
3. Reliability
4. Scalability
5. Manageability
6. Cost

## Performance



# Popular Tools

## *Development Tools:*

1. *Chrome/Firefox Developer Tools*
2. *Postman (API)*
3. *Dreamweaver*
4. *Git / SourceTree*



Adobe  
Analytics

## *Analytics Tools:*

1. *Google/Adobe Analytics*



Google Analytics

