12.AWS S3

Procedure

1. **Login to AWS Management Console**
    - Open the AWS Management Console using your registered account.
    - From the list of services, navigate to **S3 (Simple Storage Service)**.
2. **Create a New S3 Bucket**
    - Click on **"Create bucket."**
    - Enter a **unique bucket name** and choose a **region** (keep other settings as default).
    - Click **"Create bucket."**
3. **Upload an Object (PDF File)**
    - Open the newly created bucket.
    - Click **"Upload" → "Add files"** and select a **PDF file** from your local system.
    - Keep all settings **default** and click **"Upload."**
4. **Verify the Uploaded Object**
    - Once uploaded, the PDF file will be visible as an **object** inside the bucket.
    - Click on the **object name** to open its details.
5. **Accessing the Object**
    - From the object details page, click **"Open"** or copy the **object URL**.
    - The file opens or **downloads automatically** to your system, confirming successful storage and access.

11.AWS EC2

Procedure

- Login to the **AWS Management Console** using your account credentials.
- Go to **Elastic Beanstalk** service from the AWS dashboard.
- Click **"Create Application."**
- Enter an **Application Name** (for example: *MyWebApp*).
- Under **Platform**, select a runtime environment such as **Python**, **Node.js**, or **Java**.
- Choose **Sample Application** option to deploy a default app.
- Keep all other settings as **default** and click **"Create Application."**
- AWS will automatically create an **environment**, **EC2 instance**, and **storage** needed for the app.
- Wait until the **Environment Health** status shows **Green**.
- Once ready, click on the **Application URL** provided on the dashboard.
- The sample web page will open in the browser showing that the application is deployed successfully.

13.AWS LAMBDA

CODE :

```
def lambda_handler(event, context):
  return {
    'statusCode': 200,
    'body': 'Welcome to my SaaS Application!'
  }
```

1. Open AWS Lambda service in the AWS Management Console.
2. Click Create Function → Author from Scratch.
3. Enter function name (e.g., SaaS_Lambda_App).
4. Choose Runtime: Python 3.x.
5. Click Create Function.
6. In the Code Source section, paste the above code.
7. Click Deploy.
8. Click Test, create a test event (keep default JSON), and click Test again.
9. You'll get an HTML output in the execution results — showing "Welcome to My SaaS Application".

1. **TRAFFIC LIGHT**

    a) **ARDUINO**

Components Required

- Arduino UNO
- 3 LEDs (Red, Yellow, Green)
- 3 × 220Ω resistors
- Breadboard
- Jumper wires

PROCEDURE

- Place **Red, Yellow, Green LEDs** on the breadboard.
- Connect **Red → pin 13**, **Yellow → pin 12**, **Green → pin 11** of Arduino.
- Connect each LED's **short leg (–)** through a **220Ω resistor** to **GND**.
- Connect **Arduino 5V → breadboard + rail**, and **GND → – rail**.
- Upload the traffic light code using Arduino IDE.
- LEDs will glow **Red → Yellow → Green → repeat** like a real signal.

CODE

```
int red = 13;
int yellow = 12;
int green = 11;

void setup() {
  pinMode(red, OUTPUT);
  pinMode(yellow, OUTPUT);
  pinMode(green, OUTPUT);
}
```

```
void loop() {
  digitalWrite(red, HIGH);    // Red ON
  delay(5000);              // 5 sec
  digitalWrite(red, LOW);

  digitalWrite(yellow, HIGH); // Yellow ON
  delay(2000);              // 2 sec
  digitalWrite(yellow, LOW);

  digitalWrite(green, HIGH);  // Green ON
  delay(5000);              // 5 sec
  digitalWrite(green, LOW);
}
```

b. RASPBERRY PI

Components Required

- Raspberry Pi
- 3 LEDs (Red, Yellow, Green)
- 3 × 220Ω resistors
- Breadboard
- Jumper wires

Procedure

- Place Red, Yellow, and Green LEDs on the breadboard.
- Connect Red → GPIO17, Yellow → GPIO27, Green → GPIO22.
- Connect each LED's short leg (–) through a 220Ω resistor to GND.
- Use 3.3V (Pin 1) and GND (Pin 6) pins from Raspberry Pi to power the circuit.
- Run the Python code below in Thonny or terminal.
- LEDs will glow Red → Yellow → Green → repeat like a real signal.

Code

```
import RPi.GPIO as GPIO
import time
```

```
red = 17
yellow = 27
green = 22

GPIO.setmode(GPIO.BCM)
GPIO.setup(red, GPIO.OUT)
GPIO.setup(yellow, GPIO.OUT)
GPIO.setup(green, GPIO.OUT)

while True:
   GPIO.output(red, True)
   time.sleep(5)
   GPIO.output(red, False)

   GPIO.output(yellow, True)
   time.sleep(2)
   GPIO.output(yellow, False)

   GPIO.output(green, True)
   time.sleep(5)
   GPIO.output(green, False)
```

2. WEB INTEGRATION

Components Required

• Raspberry Pi
• 3 LEDs (Red, Yellow, Green)
• 3 × 220Ω resistors
• Breadboard, Jumper wires

Procedure

• Connect **Red → GPIO17**, **Yellow → GPIO27**, **Green → GPIO22** (through 220Ω to GND).
• Power from **3.3V (Pin1)** and **GND (Pin6)** pins.

/* IF NEEDED

• Install Apache & PHP:

```
sudo apt install apache2 php -y
*/
```

- Save PHP file in `/var/www/html/` as `traffic.php`.
- GET ip address by running "hostname -I " in terminal
  • Open in browser: `http://<Pi_IP>/traffic.php`

PHP Code

```
<?php
if(isset($_GET['led'])){
 $led=$_GET['led'];
 system("gpio -g mode 17 out");
 system("gpio -g mode 27 out");
 system("gpio -g mode 22 out");
 system("gpio -g write 17 0; gpio -g write 27 0; gpio -g write 22 0;");
 system("gpio -g write $led 1");
}
?>
<html>
<body>
<a href="?led=17">Red</a> |
```

```
<a href="?led=27">Yellow</a> |
<a href="?led=22">Green</a>
</body>
</html>
```

3. CAPTURE IMAGE/VIDEO AND SEND USING GMAIL (PI-CAM)

Components Required

- **Raspberry Pi (with internet)**
- **Pi Camera module**
- **Gmail account (enable "App Passwords")**

Procedure

1. Insert camera ribbon into **CSI port** near HDMI (contacts facing HDMI).
2. Lock the clip and power on.
3. **sudo raspi-config**

   → Interfacing Options → Enable Camera → Reboot
4. Install needed libraries:
5. **sudo apt install python3-picamera python3-smtplib -y**
6. **pip install yagmail**
7. Save below Python code as send_mail.py

Code

```
import yagmail, time
from picamera import PiCamera

camera = PiCamera()
camera.capture('/home/pi/image.jpg')
camera.start_recording('/home/pi/video.h264')
time.sleep(5)
camera.stop_recording()

yag = yagmail.SMTP('your@gmail.com', 'your-app-password')
yag.send(
  to='receiver@gmail.com',
  subject='Pi-Cam Capture',
  contents='Captured image & video from Raspberry Pi',
```

```
    attachments=['/home/pi/image.jpg', '/home/pi/video.h264']
)
print("Mail sent!")
```

4. TEMPERATURE SENSOR USING THINGSPEAK

COMPONENTS REQUIRED

- Raspberry Pi with internet
- DHT11 or DHT22 temperature sensor
- 10kΩ resistor (for DHT11 pull-up)
- Jumper wires & Breadboard

PROCEDURE

- Connect **VCC** of DHT11 to **3.3V (Pin 1)** on Raspberry Pi.
- Connect **GND** of DHT11 to **GND (Pin 6)**.
- Connect **DATA** of DHT11 to **GPIO4 (Pin 7)**.
- Add a **10kΩ resistor** between VCC and DATA as pull-up.
- Go to **ThingSpeak** → Sign Up / Log In.
- Click **Channels → My Channels → New Channel**.
- Enter **Name** (e.g., Temperature Monitor), enable **Field1**, save channel.
- Copy the **Write API Key** (needed to send data from Pi).
- Install Python libraries: sudo apt install python3-pip -y and pip3 install Adafruit_DHT requests.
- Run the Python code to send temperature data to ThingSpeak. C

CODE

```
import Adafruit_DHT

import requests

import time

DHT_PIN = 4

THINGSPEAK_API = "YOUR_WRITE_API_KEY"

while True:

    humidity, temp = Adafruit_DHT.read(Adafruit_DHT.DHT11, DHT_PIN)

    if temp is not None:


        requests.get(f"https://api.thingspeak.com/update?api_key={THINGSPEAK_API}&fi
        eld1={temp}")

        print(f"Temperature {temp}°C sent to ThingSpeak")

     time.sleep(15)  # send data every 15 seconds
```

5. Smart parking system

COMPONENTS REQUIRED

- Raspberry Pi with internet
- IR Obstacle Sensor
- LEDs (optional for indication)
- Jumper wires & Breadboard

PROCEDURE

- Connect VCC of IR sensor to 5V (Pin 2) on Raspberry Pi.
- Connect GND of IR sensor to GND (Pin 6).
- Connect OUT of IR sensor to GPIO17 (Pin 11).
- (Optional) Connect LEDs to indicate parking slot status.
- Go to ThingSpeak → Sign Up / Log In.
- Click Channels → My Channels → New Channel.

- Enable Field1 (for slot status) and save channel.
- Copy the Write API Key from your channel.
- Install Python libraries:

sudo apt install python3-pip -y

pip3 install requests

- Run the Python code to send parking slot status to ThingSpeak.

CODE

```
import RPi.GPIO as GPIO

import time

SENSOR = 17

LED = 27  # Optional LED pin

GPIO.setmode(GPIO.BCM)

GPIO.setup(SENSOR, GPIO.IN)

GPIO.setup(LED, GPIO.OUT)

while True:

    if GPIO.input(SENSOR):

        print("Parking Slot Empty")

        GPIO.output(LED, True)   # LED ON if empty

    else:

        print("Parking Slot Occupied")

        GPIO.output(LED, False)  # LED OFF if occupied

    time.sleep(1)
```

a) ARDUINO

Components Required

- **Arduino UNO**
- **IR Obstacle Sensor**
- **LED (optional)**
- **220Ω resistor (for LED)**
- **Jumper wires & Breadboard**

Procedure

- **Connect VCC of IR sensor to 5V on Arduino.**
- **Connect GND of IR sensor to GND on Arduino.**
- **Connect OUT of IR sensor to Digital Pin 7.**
- **(Optional) Connect LED + through 220Ω resistor → Pin 13, – → GND.**
- **Upload the Arduino code below.**

Code

```
int sensor = 7;

int led = 13; // optional

void setup() {

  pinMode(sensor, INPUT);0

  pinMode(led, OUTPUT);

  Serial.begin(9600);

}

void loop() {

  int status = digitalRead(sensor);

  if(status == HIGH){

    Serial.println("Parking Slot Empty");

    digitalWrite(led, HIGH); // LED ON if empty

  } else {

    Serial.println("Parking Slot Occupied");

    digitalWrite(led, LOW);  // LED OFF if occupied
```

```
  }
  delay(1000);
}
```