GRADUATION PROJECT

Smart Shopping Cart

Supervised by:

Prof. Zaki Bassiony

Prepared by:

Amir Hany Georges

Mohamed Ashraf

Mohamed Nabil

Mahinour Ezz

Kirollos Maged

Cairo, Egypt

2021-2022

Abstract

This study is aimed to decrease the waiting queues at checkout counter to pay for products. The system performs a similar function by automatically detecting the products a user adds to the cart and displaying their prices, through a dynamic web-application. In this manner, the buyer may immediately pay the money online through the digital billing system, and simply depart with the goods purchased without waiting in long lines. It eradicates the conventional scanning of items at the cashier desk, which accelerates the overall shopping experience. Additionally, the system employs a recommendation system in order to provide customized recommendations to each user, in real time, as they are adding items to the cart.

The system promises users a frictionless, personalized shopping experience that is free from both waiting and human error.

Keywords

Bar code reader, ESP32, LCD Display, Weight Sensor, JavaScript, React JS, APIs, Django, Postman, Database, Recommendation System, Machine Learning, Artificial Intelligence, Data Mining, Collaborative Filtering.

Chapter 1. Introduction

1.1) Motivation

As our team believes that helping people and having a sense of mission in your pursuit of a greater life will make a substantial difference to your mindset and your work ethic, we decided to work together to save people a lot of time, all while saving the stores some expenses.

Many people are with disabilities or injuries and are not able to go to hyper markets and spend too much time shopping there or they might find it difficult to shop normally; also, some customers may also find it difficult to spend too much time at the market looking for a certain item or waiting in line to pay, so as a result we were thinking of an application that might help both the customers and the store.

Finally, one of the main motives and inspirations for this project was to minimize the human interaction, and dealing with money, since “according to the WHO” exchanging money was one of the leading causes for spreading the Covid-19 virus, and since our cart is fully automated and payment is done using virtual balance, this invention can help the fight against the virus.

1.2) Problem statement

Stores nowadays witness long lines of shoppers, especially in major cities. This congestion is amplified whenever significant deals or price reductions are present, or during festive occasions. A customer doing his/her shopping the traditional way would need to wait in long queues to pay for the items in his/her cart which wastes a lot of time and often causes the frustration of

customers. In addition to that, the newly discovered virus Covid-19 that was issued a global pandemic by the World Health Organization in 2020 has influenced people to avoid crowds altogether to decrease the probability of being contaminated by the virus. Therefore, developing a smart shopping cart, that can help in avoiding waiting in long cashier lines, is an important issue for shoppers as well as for shopping centers.

1.3) Project objective

The development of smart shopping carts will succeed in solving a real-life problem that intensified after the coronavirus pandemic. By implementing the suggested idea, any customer will be able to do his shopping and check himself out momentarily without wasting a second waiting in queues or crowds. In addition to that, the usage of the virtual billing system will dominate over the traditional exchange of cash-money which also serves as another measure of preventing the spread of coronavirus. Besides the customer satisfaction and the enhanced user experience as a result of the application of such systems. This project will save stores the salaries of cashiers, since every customer will be his own cashier. Another prospect to take in account is that this new technology will attract customers to the stores that use them since shoppers will be interested in trying them out.

1.4) Proposed solution

The solution suggested is developing a system that can be adopted by malls and big stores to solve the problems stated above “while having a reasonable cost”.

The system proposed is an intelligent shopping cart that offers customers the facility of logging into their account that contains their personal information, payment information and their current balance. Logging in is achieved easily through the website. The Barcode code reader will be used for identifying the items that lie inside the cart, such that the shopping cart will be able to add the price of the item to the total cart balance. In the same manner, any item removed from the cart, its price will also be removed from the total cart balance. Finally, after the list of items in the cart is confirmed, the total cart balance is deduced from the shopper’s balance.

To achieve such objectives, an embedded system will be implemented and the concept of IOT (Internet of Things) will be employed.

In summary, an embedded system will be developed that will employ the concepts of IOT to enrich the user's shopping experience.

Chapter 2. Embedded System

2.1) What is an Embedded System?

A microprocessor-based computer hardware and software system called an embedded system is created to carry out a particular task, either independently or as a component of a larger system. An integrated circuit built to do computing for real-time processes is at the heart of the system.

From a single microcontroller to a group of linked processors with networks and peripherals, complexity can range from having no user interface to having intricate graphical user interfaces. Depending on the job for which it is created, an embedded system's complexity varies greatly.

Applications for embedded systems include hybrid cars, avionics, digital watches, microwaves, and more. Embedded systems consume up to 98 percent of all produced microprocessors.

2.2) What is a Microcontroller?

A microcontroller is a compact integrated circuit designed to govern a specific operation in an embedded system. A typical microcontroller includes a processor, memory, and input/output (I/O) peripherals on a single chip.

Sometimes referred to as an embedded controller or microcontroller unit (MCU), microcontrollers are found in vehicles, robots, office machines, medical devices, mobile radio transceivers, vending machines, and home appliances, among other devices. They are essentially simple miniature personal computers (PCs) designed to control small features of a larger component, without a complex front-end operating system (OS).

2.3) How do microcontrollers work?

A microcontroller is embedded inside of a system to control a singular function in a device. It does this by interpreting data it receives from its I/O peripherals using its central processor. The

temporary information that the microcontroller receives is stored in its data memory, where the processor accesses it and uses instructions stored in its program memory to decipher and apply the incoming data. It then uses its I/O peripherals to communicate and enact the appropriate action.

Microcontrollers are used in a wide array of systems and devices. Devices often utilize multiple microcontrollers that work together within the device to handle their respective tasks.

For example, a car might have many microcontrollers that control various individual systems within, such as the anti-lock braking system, traction control, fuel injection or suspension control. All the microcontrollers communicate with each other to inform the correct actions. Some might communicate with a more complex central computer within the car, and others might only communicate with other microcontrollers. They send and receive data using their I/O peripherals and process that data to perform their designated tasks.

2.4) What are the elements of a microcontroller?

The core elements of a microcontroller are:

- The processor (CPU) -- A processor can be thought of as the brain of the device. It processes and responds to various instructions that direct the microcontroller's function. This involves performing basic arithmetic, logic, and I/O operations. It also performs data transfer operations, which communicate commands to other components in the larger embedded system.
- Memory -- A microcontroller's memory is used to store the data that the processor receives and uses to respond to instructions that it's been programmed to carry out. A microcontroller has two main memory types:
    1. Program memory, which stores long-term information about the instructions that the CPU carries out. Program memory is non-volatile memory, meaning it holds information over time without needing a power source.
    2. Data memory, which is required for temporary data storage while the instructions are being executed. Data memory is volatile, meaning the data it holds is temporary and is only maintained if the device is connected to a power source.
- I/O peripherals -- The input and output devices are the interface for the processor to the outside world. The input ports receive information and send it to the processor in the

form of binary data. The processor receives that data and sends the necessary instructions to output devices that execute tasks external to the microcontroller.

While the processor, memory and I/O peripherals are the defining elements of the microprocessor, there are other elements that are frequently included. The term *I/O peripherals* itself simply refers to supporting components that interface with the memory and processor. There are many supporting components that can be classified as peripherals. Having some manifestation of an I/O peripheral is elemental to a microprocessor because they are the mechanism through which the processor is applied.

Other supporting elements of a microcontroller include:

- Analog to Digital Converter (ADC) -- An ADC is a circuit that converts analog signals to digital signals. It allows the processor at the center of the microcontroller to interface with external analog devices, such as sensors.
- Digital to Analog Converter (DAC) -- A DAC performs the inverse function of an ADC and allows the processor at the center of the microcontroller to communicate its outgoing signals to external analog components.
- System bus -- The system bus is the connective wire that links all components of the microcontroller together.
- Serial port -- The serial port is one example of an I/O port that allows the microcontroller to connect to external components. It has a similar function to a USB or a parallel port but differs in the way it exchanges bits.

2.5) Software Tools:

- Eclipse IDE
- Arduino IDE

2.6) Hardware Tools:

- ATmega16
- ESP32 DEVKIT V1 DOIT
- Load Cell (Straight Bar)
- Load Cell Amplifier HX711

GM65 (Bar Code Reader Module)

2.7) ATmega16:

2.7.1) What is ATmega16?

Atmel Corporation manufactured the ATmega16 microcontroller which comes under Atmel's Advanced Virtual RISC family. It has an advanced RISC (Reduced Instruction Set Computing) system and a high-performance microcontroller. This is the advanced version of the 8051 microcontrollers which has the features beat the 8051 microcontroller features. It's a computer inbuilt with CPU, RAM, ROM, EEPROM, Timers, Counters, ADC and last four 8-bit ports like port A, port B, port C, port D. Each port has 8 input and output pins for extra performance. In the below section, we can observe the features of this microcontroller.

2.7.2) ATmega16 Features

• High-performance, Low-power AVR® 8-bit Microcontroller

• Advanced RISC Architecture

– 131 Powerful Instructions

– Most Single-clock Cycle Execution

– 32 x 8 General Purpose Working Registers

– Fully Static Operation

– Up to 16 MIPS Throughput at 16 MHz

– On-chip 2

- cycle Multiplier

• High Endurance Non-volatile Memory segments

– 16K Bytes of In-System Self programmable Flash program memory

– 512 Bytes EEPROM

– 1K Byte Internal SRAM

– Write/Erase Cycles: 10,000 Flash/100,000 EEPROM

– Data retention: 20 years at 85°C/100 years at 25°C (1)

– Optional Boot Code Section with Independent Lock Bits

In-System Programming by On-chip Boot Program

True Read-While-Write Operation

– Programming Lock for Software Security

• JTAG (IEEE std. 1149.1 Compliant) Interface

– Boundary-scan Capabilities According to the JTAG Standard

– Extensive On-chip Debug Support

– Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface

• Peripheral Features

– Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes

– One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode

– Real Time Counter with Separate Oscillator – Four PWM Channels

– 8-channel, 10-bit ADC

8 Single-ended Channels

7 Differential Channels in TQFP Package Only 2 Differential Channels with Programmable Gain at 1x, 10x, or 200x

– Byte-oriented Two-wire Serial Interface

– Programmable Serial USART

– Master/Slave SPI Serial Interface

– Programmable Watchdog Timer with Separate On-chip Oscillator

– On-chip Analog Comparator

• Special Microcontroller Features

– Power-on Reset and Programmable Brown-out Detection

– Internal Calibrated RC Oscillator

– External and Internal Interrupt Sources

– Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby

• I/O and Packages

– 32 Programmable I/O Lines

– 40-pin PDIP, 44-lead TQFP, and 44-pad QFN/MLF

• Operating Voltages

– 2.7 - 5.5V for ATmega16L

– 4.5 - 5.5V for ATmega16

• Speed Grades

– 0 - 8 MHz for ATmega16L

– 0 - 16 MHz for ATmega16

• Power Consumption @ 1 MHz, 3V, and 25·C for ATmega16L

– Active: 1.1 mA

– Idle Mode: 0.35 mA

– Power-down Mode: < 1 µA

2.5.3) ATmega16 Pin Configurations

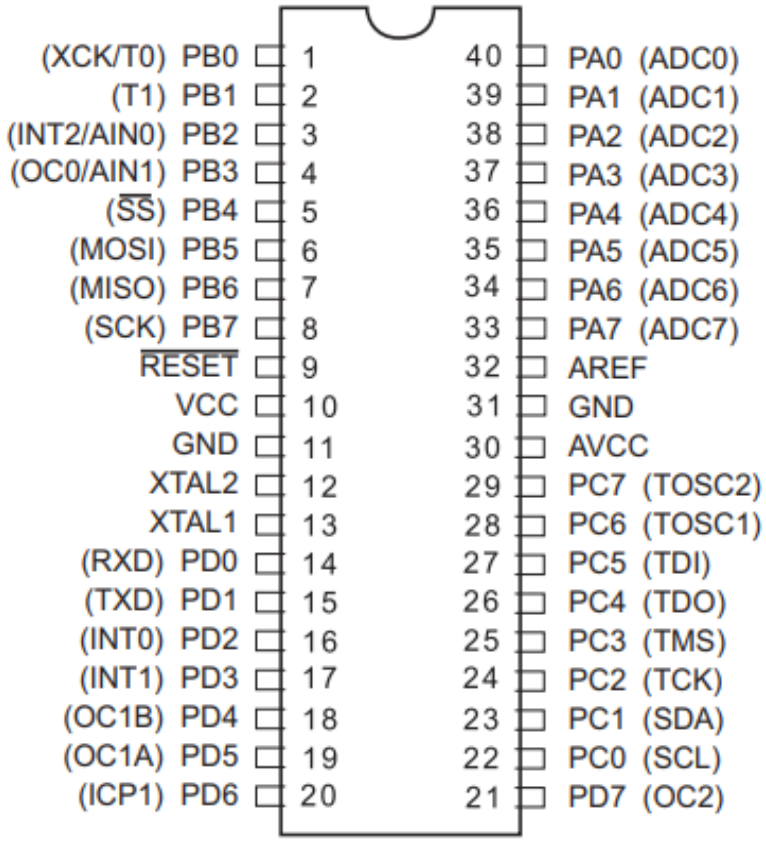


Figure 1. pin description of ATmega16

2.7.4) ATmega16 Block Diagram

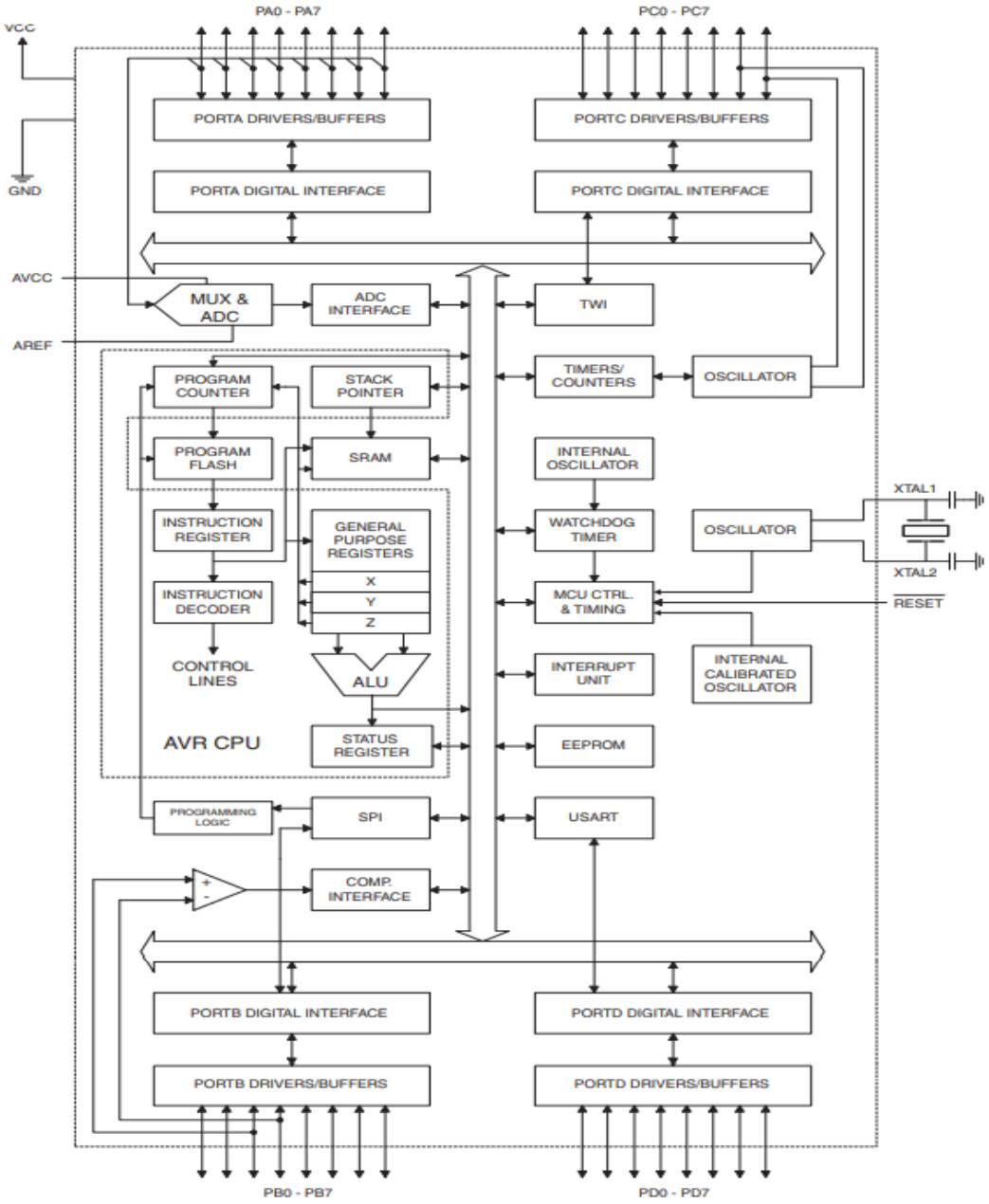


Figure 2. Block diagram of ATmega16

2.8) ESP32 Dev Kit V1

ESP32 is a single 2.4 GHz Wi-Fi-and-Bluetooth combo chip designed with the TSMC ultra-low-power 40 nm technology. It is designed to achieve the best power and RF performance, showing robustness, versatility, and reliability in a wide variety of applications and power scenarios.

ESP32 is designed for mobile, wearable electronics, and Internet-of-Things (IoT) applications. It features all the state-of-the-art characteristics of low-power chips, including fine-grained clock

gating, multiple power modes, and dynamic power scaling. For instance, in a low-power IoT sensor hub application scenario, ESP32 is woken up periodically only when a specified condition is detected. Low-duty cycle is used to minimize the amount of energy that the chip expends. The output of the power amplifier is also adjustable, thus contributing to an optimal trade-off between communication range, data rate and power consumption.

ESP32 is a highly integrated solution for Wi-Fi-and-Bluetooth IoT applications, with around 20 external components. ESP32 integrates an antenna switch, RF balun, power amplifier, low noise receive amplifier, filters, and power management modules. As such, the entire solution occupies minimal Printed Circuit Board (PCB) area. ESP32 uses CMOS for single-chip fully-integrated radio and baseband, while also integrating advanced calibration circuitries that allow the solution to remove external circuit imperfections or adjust to changes in external conditions. As such, the mass production of ESP32 solutions does not require expensive and specialized Wi-Fi testing equipment.

2.8.1) ESP32 Features

2.8.1.1) Wi-Fi Key Features

• 802.11 b/g/n

• 802.11 n (2.4 GHz), up to 150 Mbps

• WMM

• TX/RX A-MPDU, RX A-MSDU

• Immediate Block ACK

• Defragmentation

• Automatic Beacon monitoring (hardware TSF)

• 4 × virtual Wi-Fi interfaces

• Simultaneous support for Infrastructure Station, SoftAP, and Promiscuous modes Note that

when ESP32 is in Station mode, performing a scan, the SoftAP channel will be changed.

• Antenna diversity

2.8.1.2) Bluetooth Key Features

Compliant with Bluetooth v4.2 BR/EDR and Bluetooth LE specifications

• Class-1, class-2 and class-3 transmitter without external power amplifier

• Enhanced Power Control

• +9 dBm transmitting power

• NZIF receiver with –94 dBm Bluetooth LE sensitivity

• Adaptive Frequency Hopping (AFH)

• Standard HCI based on SDIO/SPI/UART

• High-speed UART HCI, up to 4 Mbps

• Bluetooth 4.2 BR/EDR Bluetooth LE dual mode controller

• Synchronous Connection-Oriented/Extended (SCO/eSCO)

• CVSD and SBC for audio codec

• Bluetooth Piconet and Scatternet

• Multi-connections in Classic Bluetooth and Bluetooth LE

• Simultaneous advertising and scanning

2.8.2) MCU and Advanced Features

2.8.2.1) CPU and Memory

• Xtensa® single-/dual-core 32-bit LX6 microprocessor(s)

• CoreMark® score:

  – 1 core at 240 MHz: 504.85 CoreMark; 2.10 CoreMark/MHz

  – 2 cores at 240 MHz: 994.26 CoreMark; 4.14 CoreMark/MHz

• 448 KB ROM

• 520 KB SRAM

• 16 KB SRAM in RTC

• QSPI supports multiple flash/SRAM

2.8.2.2) Clocks and Timers

• Internal 8 MHz oscillator with calibration

• Internal RC oscillator with calibration

• External 2 MHz <-> 60 MHz crystal oscillator (40 MHz only for Wi-Fi/Bluetooth functionality)

• External 32 kHz crystal oscillator for RTC with calibration

• Two timer groups, including 2 × 64-bit timers and 1 × main watchdog in each group

• One RTC timer

• RTC watchdog

2.8.2.3) Advanced Peripheral Interfaces

• 34 × programmable GPIOs

• 12-bit SAR ADC up to 18 channels

• 2 × 8-bit DAC

• 10 × touch sensors

• 4 × SPI

• 2 × I2S

• 2 × I2C

• 3 × UART

• 1 host (SD/eMMC/SDIO)

• 1 slave (SDIO/SPI)

• Ethernet MAC interface with dedicated DMA and IEEE 1588 support

• TWAI®, compatible with ISO 11898-1 (CAN Specification 2.0)

• RMT (TX/RX)

• Motor PWM

• LED PWM up to 16 channels

• Hall sensor

2.8.2.4) Security

• Secure boot

• Flash encryption

• 1024-bit OTP, up to 768-bit for customers

• Cryptographic hardware acceleration:

– AES

– Hash (SHA-2)

– RSA

– ECC

– Random Number Generator (RNG)

2.8.3) Hardware Specifications:

- Operating voltage/Power supply 3.0 V ~ 3.6 V

- Operating current Average: 80 mA
- Minimum current delivered by power supply 500 mA
- Recommended operating ambient temperature range –40 °C ~ +85 °C
- Package size 18 mm × 25.5 mm × 3.10 mm
- Moisture sensitivity level (MSL) Level 3

2.8.4) ESP32 Block Diagram

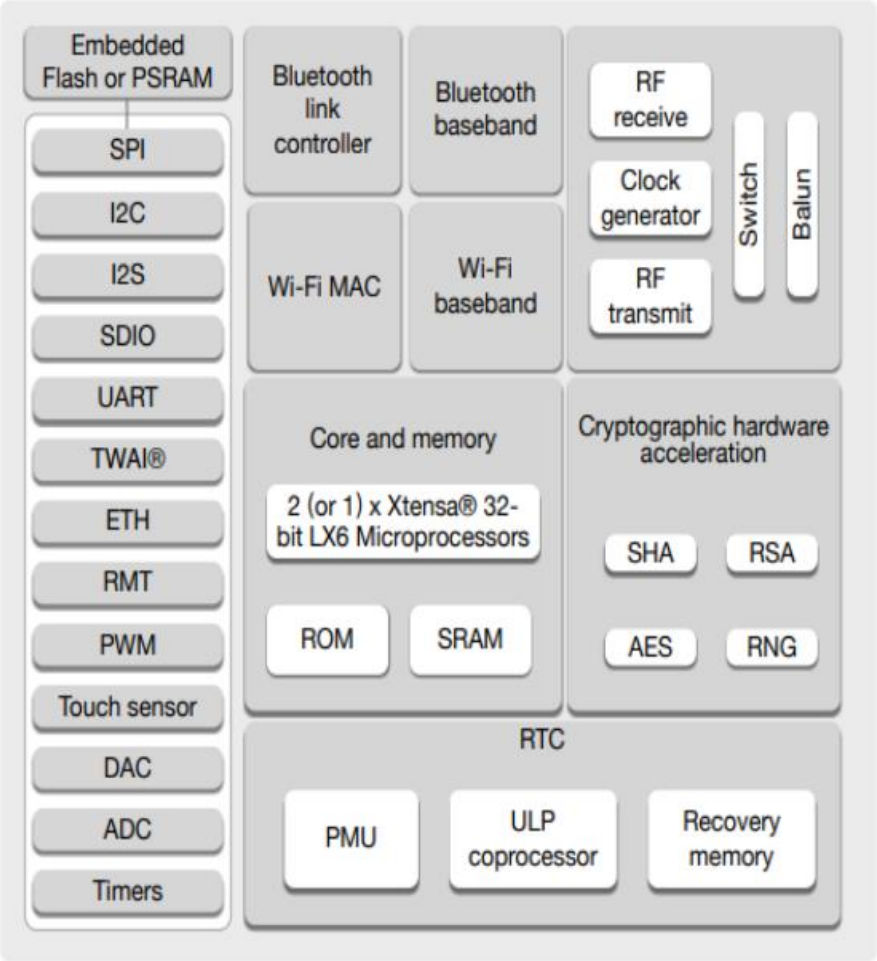


Figure 3. Block diagram of ESP32

2.8.5) ESP32 DevKit V1 Pin Configurations

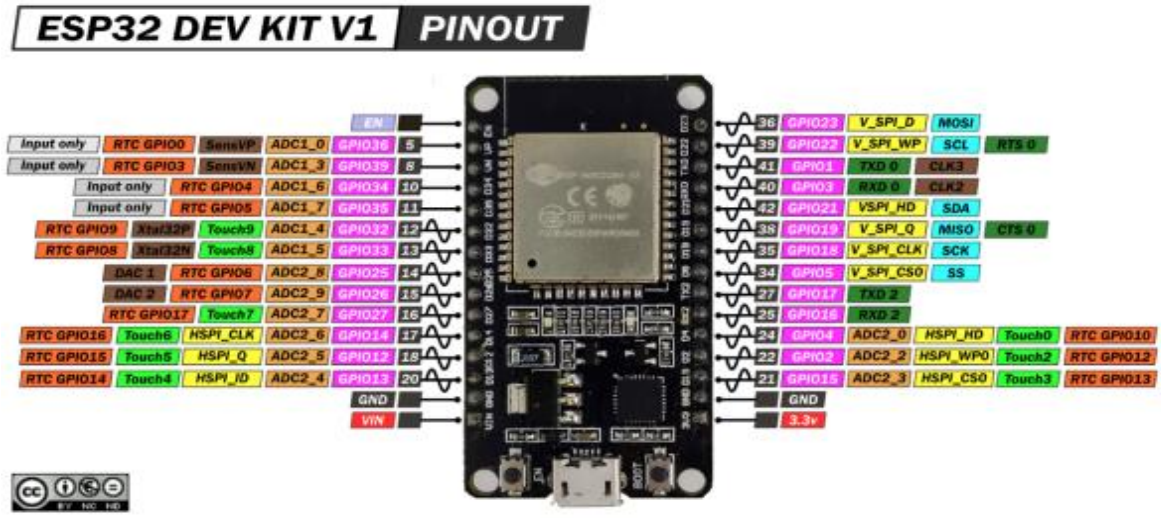


Figure 4. Pin description of ESP32 Dev Kit V1

2.9) Load Cell (Straight Bar)

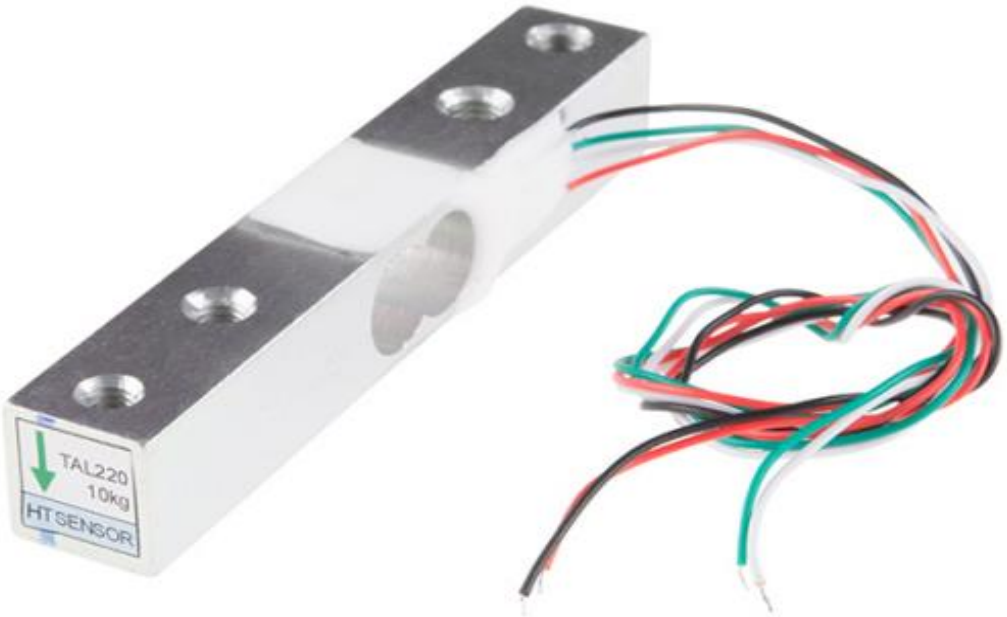


Figure 5. Load Cell (Straight Bar)

A load cell is a physical element (or transducer if you want to be technical) that can translate pressure (force) into an electrical signal.

So, what does that mean? There are three main ways a load cell can translate an applied force into a measurable reading.

2.9.1) Hydraulic Load Cells

Hydraulic load cells use a conventional piston and cylinder arrangement to convey a change in pressure by the movement of the piston and a diaphragm arrangement which produces a change in the pressure on a Bourdon tube connected with the load cells.

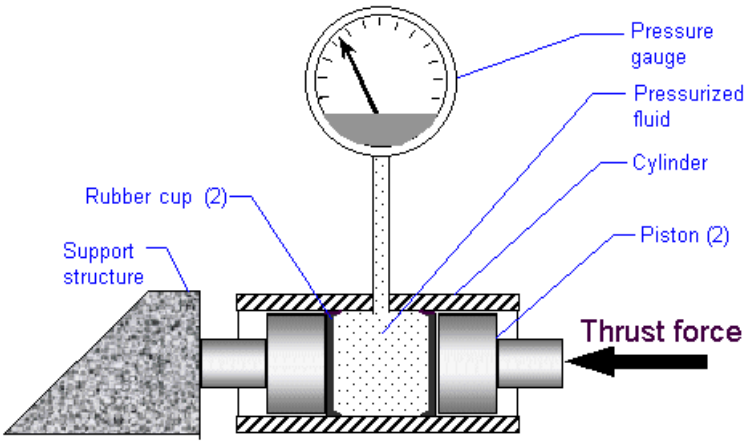


Figure 6. Hydraulic Load Cell

2.9.2) Pneumatic Load Cells

Pneumatic load cells use air pressure applied to one end of a diaphragm, and it escapes through the nozzle placed at the bottom of the load cell, which has a pressure gauge inside of the cell.

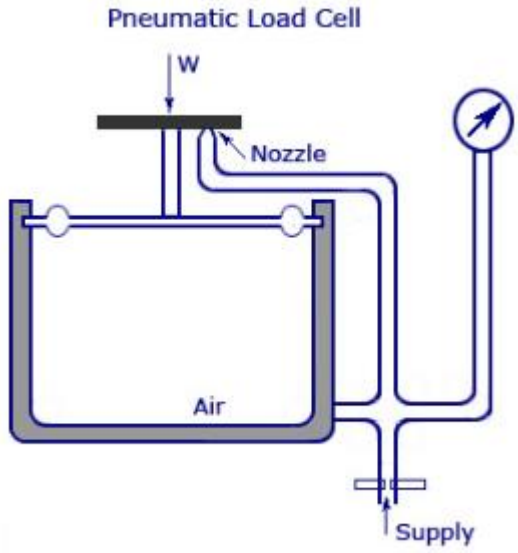


Figure 7. Pneumatic Load Cell

2.9.3) Strain Gauge Load Cells

And lastly (though there are many other less common load cell set ups), there is a strain gauge load cell, which is a mechanical element of which the force is being sensed by the deformation of a (or several) strain gauge(s) on the element.

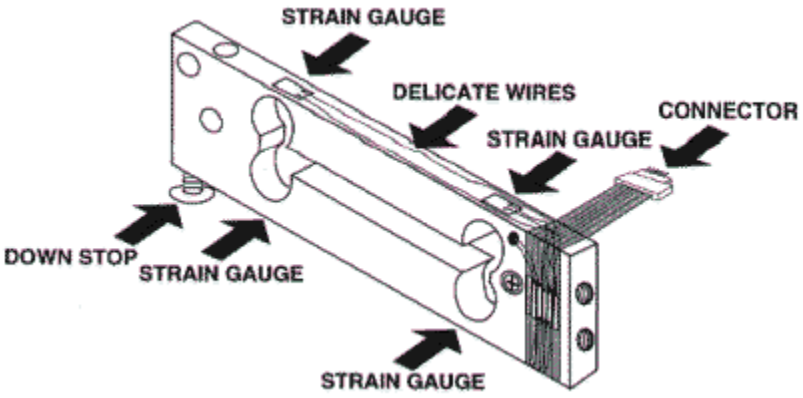


Figure 8. Strain Gauge Load Cell

In bar strain gauge load cells, the cell is set up in a "Z" formation so that torque is applied to the bar and the four strain gauges on the cell will measure the bending distortion, two measuring compression and two tension. When these four strain gauges are set up in a wheatstone bridge formation, it is easy to accurately measure the small changes in resistance from the strain gauges.

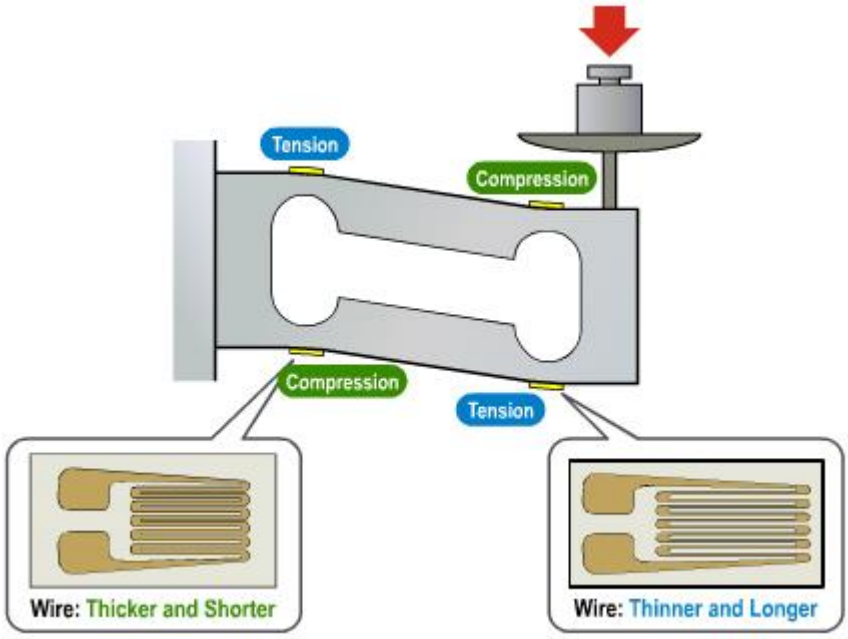


Figure 9. Strain Gauge Load Cell Z formation

Most strain gauge load cells work in very similar ways, but may vary in size, material, and mechanical setup, which can lead to each cell having different max loads and sensitivities that they can handle.

2.9.4) Strain Gauge Basics

A strain gauge is a device that measures electrical resistance changes in response to, and proportional of, strain (or pressure or force or whatever you so desire to call it) applied to the device. The most common strain gauge is made up of very fine wire, or foil, set up in a grid pattern in such a way that there is a linear change in electrical resistance when strain is applied in one specific direction, most found with a base resistance of 120Ω, 350Ω, and 1,000Ω.

2.9.5) Gauge Factor

Each strain gauge has a different sensitivity to strain, which is expressed quantitatively as the **gauge factor (GF)**. The gauge factor is defined as the ratio of fractional change in electrical resistance to the fractional change in length (strain). (The gauge factor for metallic strain gauges is typically around 2.)

2.9.6) Small Changes in Strain

We set up a strain gauge load cell and measure that change in resistance and all is good, right? Not so fast. Strain measurements rarely involve quantities larger than a few millistrain ($e.10^{-3}$) (fancy units for strain, but still very small).

So, let's take an example: suppose you put a strain of 500με. A strain gauge with a gauge factor of 2 will have a change in electrical resistance of only:

$$2 \cdot (500 \cdot 10^{-6}) = 0.1\%$$

For a 120Ω gauge, this is a change of only 0.12Ω. 0.12Ω is a very small change, and, for most devices, couldn't be detected, let alone detected accurately. So, we are going to need another device that can either accurately measure super small changes in resistance or a device that can take that very small change in resistance and turn it into something that we can measure accurately.

2.9.7) Amplifiers and Wheatstone Bridge

This is where an amplifier, such as the HX711 or the NAU7802 comes in handy.

A good way of taking small changes in resistance and turning it into something more measurable is using a wheatstone bridge. A wheatstone bridge is a configuration of four resistors with a known voltage applied like this:

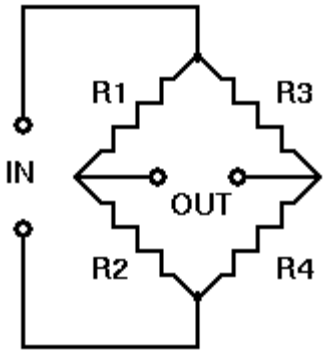


Figure 10. Wheatstone Bridge

where Vin is a known constant voltage, and the resulting Vout is measured. If $\frac{R1}{R2} = \frac{R3}{R4}$, then Vout is 0, but if there is a change to the value of one of the resistors, Vout will have a resulting change that can be measured and is governed by the following equation using Ohm's law:

$$Vout = \left[\left(R3/(R3+R4)\right) - R2/(R1+R2)\right]Vin$$

By replacing one of the resistors in a wheatstone bridge with a strain gauge, we can easily measure the change in Vout and use that to assess the force applied.

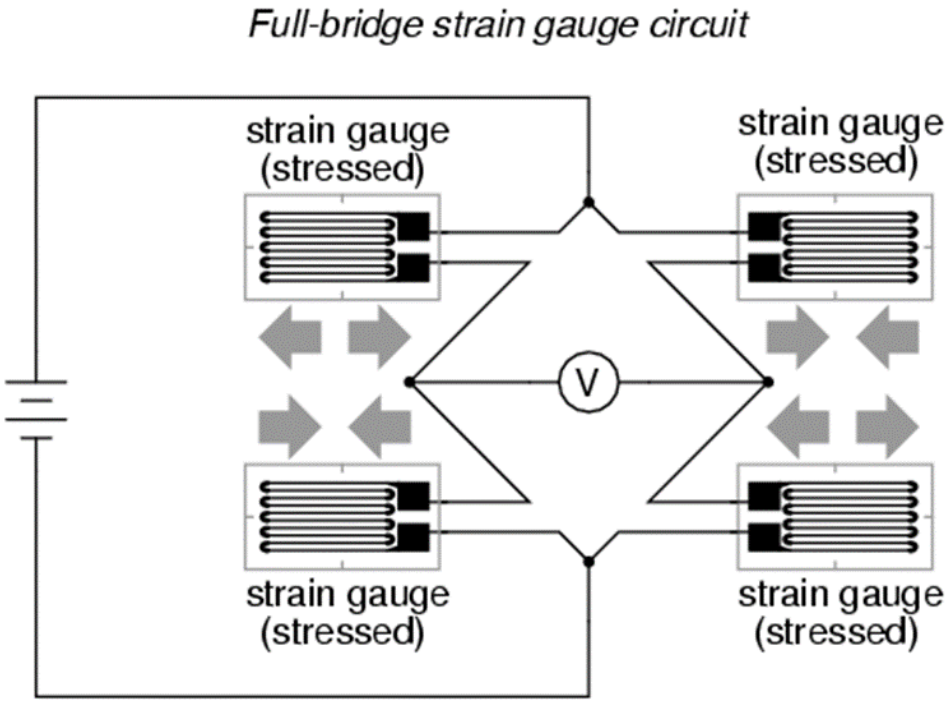


Figure 11. Full-bridge strain gauge circuit

2.10) Load Cell Amplifier HX711

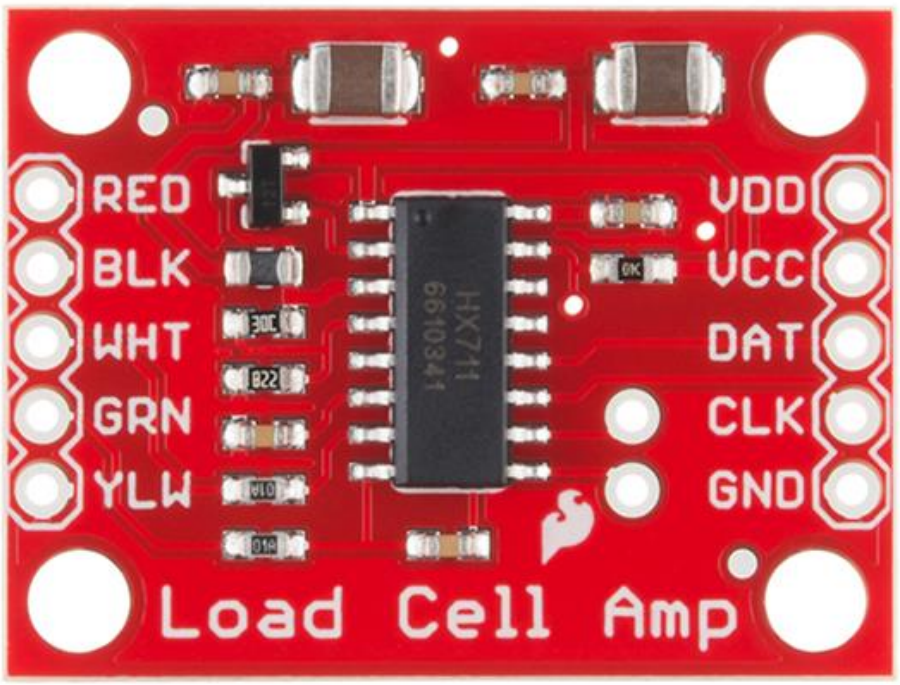


Figure 12. HX711 Integrated Circuit

Based on Avia Semiconductor's patented technology, HX711 is a precision 24-bit analog-to-digital converter (ADC) designed for weigh scales and industrial control applications to interface directly with a bridge sensor.

Load cells use a four-wire Wheatstone bridge configuration to connect to the HX711. These are commonly colored RED, BLK, WHT, GRN and YLW. Each color corresponds to the conventional color coding of load cells.

- Red (Excitation+ or VCC)
- Black (Excitation- or GND)
- White (Amplifier-, Signal- or Output-)
- Green (A+, S+ or O+)
- Yellow (Shield)

The YLW pin acts as an optional input that is not hooked up to the strain gauge but is utilized to ground and shield against outside EMI (electromagnetic interference).

2.10.1) Features

• Two selectable differential input channels

• On-chip active low noise PGA with selectable gain of 32, 64 and 128

• On-chip power supply regulator for load-cell and ADC analog power supply

• On-chip oscillator requiring no external component with optional external crystal

• On-chip power-on-reset

• Simple digital control and serial interface: pin-driven controls, no programming needed

• Selectable 10SPS or 80SPS output data rate

• Simultaneous 50 and 60Hz supply rejection

• Current consumption including on-chip analog power supply regulator:

normal operation < 1.5mA, power down < 1uA

• Operation supply voltage range: 2.6 ~ 5.5V

• Operation temperature range: -40 ~ +85℃

• 16 pin SOP-16 package

2.10.2) HX711 Block Diagram

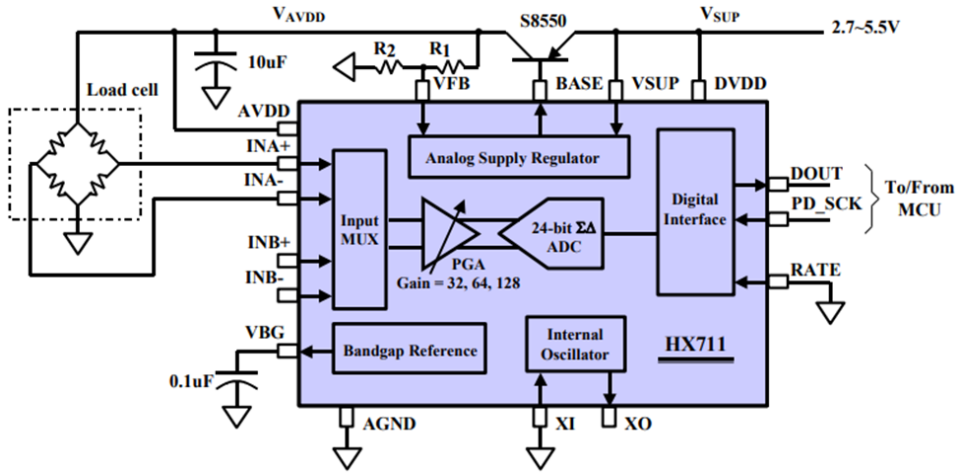


Figure 13. HX711 Block Diagram

2.10.3) HX711 Pin Description

Regulator Power	VSUP	1	16	DVDD	Digital Power
Regulator Control Output	BASE	2	15	RATE	Output Data Rate Control Input
Analog Power	AVDD	3	14	XI	Crystal I/O and External Clock Input
Regulator Control Input	VFB	4	13	XO	Crystal I/O
Analog Ground	AGND	5	12	DOUT	Serial Data Output
Reference Bypass	VBG	6	11	PD_SCK	Power Down and Serial Clock Input
Ch. A Negative Input	INNA	7	10	INPB	Ch. B Positive Input
Ch. A Positive Input	INPA	8	9	INNB	Ch. B Negative Input

Figure 14. HX711 Pins

Pin #	Name	Function	Description
1	VSUP	Power	Regulator supply: 2.7 ~ 5.5V
2	BASE	Analog Output	Regulator control output（NC when not used）
3	AVDD	Power	Analog supply: 2.6 ~ 5.5V
4	VFB	Analog Input	Regulator control input（connect to AGND when not used）
5	AGND	Ground	Analog Ground
6	VBG	Analog Output	Reference bypass output
7	INA-	Analog Input	Channel A negative input
8	INA+	Analog Input	Channel A positive input
9	INB-	Analog Input	Channel B negative input
10	INB+	Analog Input	Channel B positive input
11	PD_SCK	Digital Input	Power down control (high active) and serial clock input
12	DOUT	Digital Output	Serial data output
13	XO	Digital I/O	Crystal I/O (NC when not used）
14	XI	Digital Input	Crystal I/O or external clock input, 0: use on-chip oscillator
15	RATE	Digital Input	Output data rate control, 0: 10Hz; 1: 80Hz
16	DVDD	Power	Digital supply: 2.6 ~ 5.5V

Table 1. HX711 Pin Description

2.10.4) How does it work?

Pin PD_SCK and DOUT are used for data retrieval, input selection, gain selection and power down controls. When output data is not ready for retrieval, digital output pin DOUT is high. Serial clock input PD_SCK should be low. When DOUT goes to low, it indicates data is ready for retrieval. By applying 25~27 positive clock pulses at the PD_SCK pin, data is shifted out from the DOUT output pin. Each PD_SCK pulse shifts out one bit, starting with the MSB bit first, until all 24 bits are shifted out. The 25th pulse at PD_SCK input will pull DOUT pin back to high Input and gain selection are controlled by the number of the input PD_SCK pulses PD_SCK clock pulses should not be less than 25 or more than 27 within one conversion period, to avoid causing serial communication error.

PD_SCK Pulses	Input channel	Gain
25	A	128
26	B	32
27	A	64

Table 2. HX711 Gain configuration

Depending on the Gain we will send 25, 26 or 27 Pulses as specified from the table above.

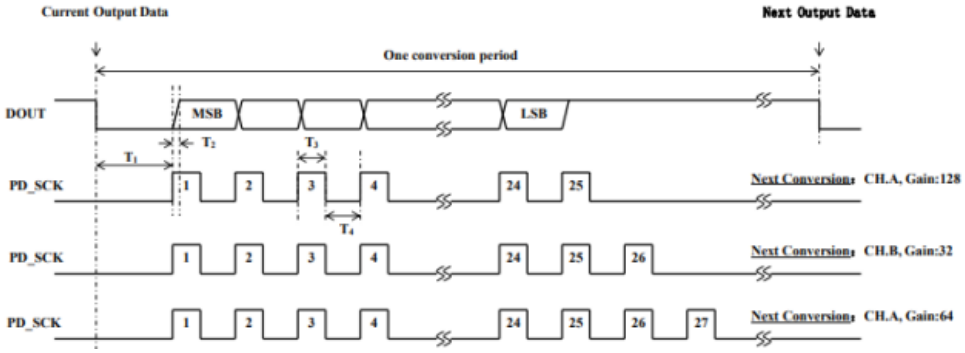


Figure 14. HX711 Data output, input and gain selection timing and control

Symbol	Note	MIN	TYP	MAX	Unit
T_1	DOUT falling edge to PD_SCK rising edge	0.1			μs
T_2	PD_SCK rising edge to DOUT data ready			0.1	μs
T_3	PD_SCK high time	0.2	1	50	μs
T_4	PD_SCK low time	0.2	1		μs

Table 3. HX711 function timings

2.10.5) To obtain a correct reading from the HX711

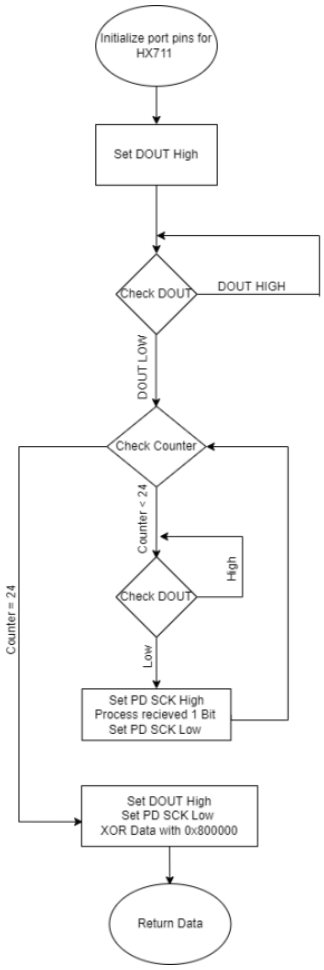


Figure 15. HX711 Data acquisition flow chart

Note: additional step of calibration is required to obtain human readable data.

The sensor has some offset value that needs to be subtracted and then it needs to be scaled with a factor to present human readable data.

```
#include "lcd.h"
#include "hx711.h"

/* Global variable to store weight reading */
uint32 Weight = 0;

/* Scale used for calibration process */
#define scale 3.275132275

//3.386243386
//3.275132275

int main(void){

    LCD_init();
    HX711_init(128);
    HX711_set_offset(75);
    LCD_displayString("Weight in grams:");
    LCD_moveCursor(1,0);

    while(1){
        Weight = HX711_get_value(50);
        Weight = (int)Weight/scale;
```

Additional Scaling factors

Figure 16. HX711 scaling results to a human readable data

2.11) Universal Synchronous Asynchronous Receiver/Transmitter (USART)

USART stands for Universal Synchronous Asynchronous Receiver Transmitter. It is sometimes called the Serial Communications Interface or SCI. Synchronous operation uses a clock and data line while there is no separate clock accompanying the data for Asynchronous transmission. Since there is no clock signal in asynchronous operation, one pin can be used for transmission and another pin can be used for reception. Both transmission and reception can occur at the same time — this is known as full duplex operation. Transmission and reception can be independently enabled. However, when the serial port is enabled, the USART will control both pins and one cannot be used for general purpose I/O when the other is being used for transmission or reception. The USART is most commonly used in the asynchronous mode.

The most common use of the USART in asynchronous mode is to communicate to a PC serial port using the RS-232 protocol.

The USART can both transmit and receive, and we will now briefly look at how this is implemented in the USART.

UARTs transmit data asynchronously, which means there is no clock signal to

synchronize the output of bits from the transmitting UART to the sampling of bits by

the receiving UART. Instead of a clock signal, the transmitting UART adds start and

stop bits to the data packet being transferred. These bits define the beginning and

end of the data packet, so the receiving UART knows when to start reading the bits. When the receiving UART detects a start bit, it starts to read the incoming bits at a

specific frequency known as the baud rate. Baud rate is a measure of the speed of data transfer, expressed in bits per second(bps). Both UARTs must operate at about the same baud rate. The baud rate between the transmitting and receiving UARTs can only differ by about 10% before the timing of bits gets too far off. Both UARTs must also be configured to transmit and receive the same data packet structure. for UART and most serial communications, the baud rate needs to be set the same on both the transmitting and receiving device. the set baud rate will serve as the maximum number of bits per second to be transferred.

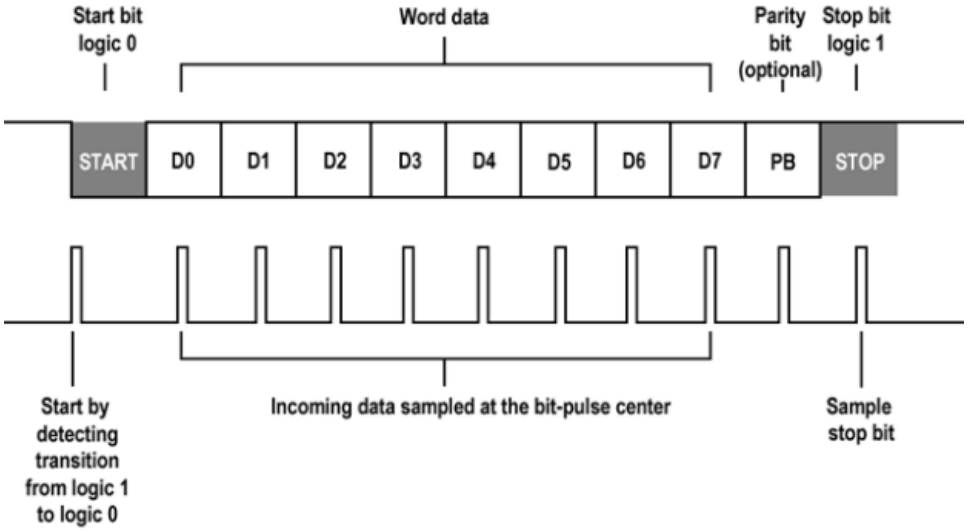


Figure 17. USART Data Frame

- Start bit: To declare the start of transmission, always low (0).
- Data bits: 5,6,7, 8 or 9 bits of useful data bits.
- Parity bit: To check for transmission errors.
- Stop bit: one or two stop bits to declare end of frame, always high (1).

2.11.1) USART Features

- Full Duplex Operation (Independent Serial Receive and Transmit Registers).
- Asynchronous or Synchronous Operation.
- Master or Slave Clocked Synchronous Operation.
- High Resolution Baud Rate Generator.
- Error Detection (Parity Bit).
- Supports Serial Frames with 5, 6, 7, 8, or 9 Data bits and 1 or 2 Stop Bits.

2.12) GM65 (Bar Code Reader Module)

GM65 Bar code reader module is a high-performance scanner, can read 1D bar code easily and read 2D bar code with high speed. It also wins high scan speed for linear code, even for bar code on paper or screen. GM65 bar code reader module is an advanced bar code decoding algorithm which developed on image recognition algorithm, can easily and accurately read bar code, simplify secondary development. MG65 works stable in dark and large temperature range.

2.12.1) Specifications

Default scan mode		Continuous scan	
Read code time for once		3s	Parameter：0.1-25.5s; step-size: 0.1s; 0 means no time limited
Reading interval		1S	Parameter：0.1-25.5s; step-size: 0.1s; 0 means no time limited
Output		GBK	GBK、UNICODE、 BIG5
Interface		USB KBW	USB KBW、serial port、USB VCom
Interface (TTL-232)	Serial Baud Rate	9600	adjustable，details at 2.1
	Verification	N	
	Data bit	8	
	Stop bit	1	
	CTSRTS	No	
serial mode	Read code time for once	5s	Parameter：0.1-25.5s; step-size: 0.1s; 0 means no time limited

Table 4. GM65 specifications

1	Opreating Voltage	DC 4.2 – 6.0
2	Standby Current	30 mA
3	Opreating Current	160 mA
4	Sleep Current	3 mA

Table 5. GM65 power specifications

1	Light	White light
2	Capture light	Red
3	Scan Angle	Roll:0-360°，Pitch:±65°，Yaw:±60°
4	Resolution	648x 488
5	Scanning angle	35°（Inclination），28°（Elevation）

Table 6. Hardware specifications

2.12.2) How can we program this module?

This module is best programmed by scanning the QR codes inside its user manual.

Example:

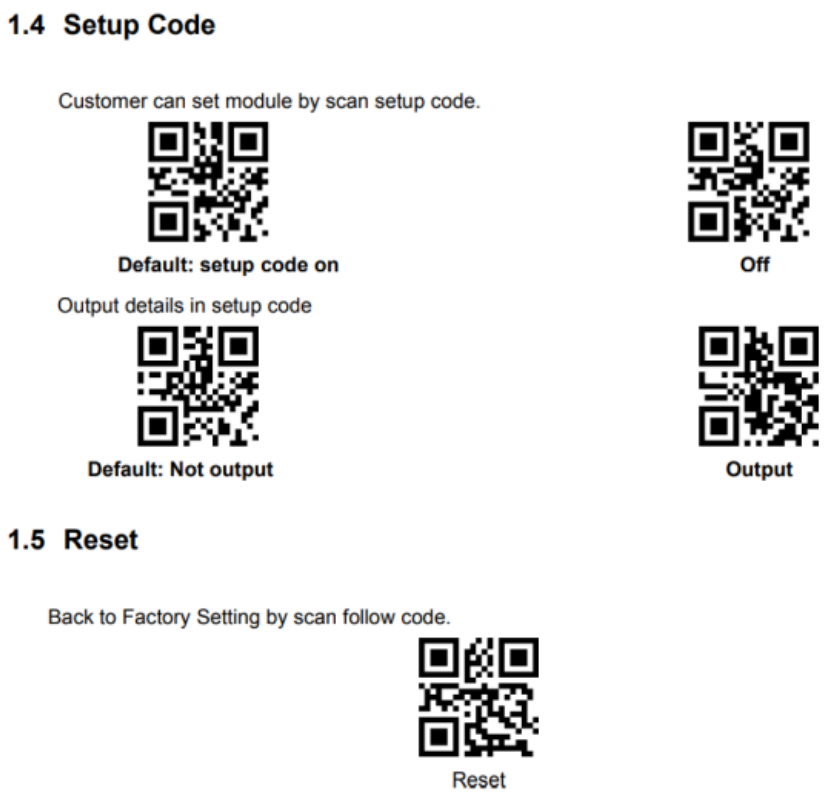


Figure 17. GM65 QR code programming

We can interface with this module easily. Because its physical layer based on TTL-232.

We will be able to receive data from this module by ATmegas16' USART module and send that code to our ESP32 which in its turn sends data to the server through API.

Default Parameter as form 2- 1. Only Baud Rate can be changed.

Form 2- 1 Default Parameters

Parameters	Default
Series communication interface	Standard TTL-232
Baud rate	9600
Verification	N
Data bit	8
Stop bit	1
CTSRTS	N

Table 7. GM65 default parameters

Since this module Baud rate is 9600 by default, know we know what number we place inside UBRR register inside ATmega16.

We will use double speed mode on ATmega16 to make the communication faster.

We can do that by setting U2X Bit inside USART Control and Status Register A – UCSRA.

Calculate that number using the equation from the ATmega16's Datasheet:

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRR Value
Asynchronous Normal Mode (U2X = 0)	$BAUD = \frac{f_{OSC}}{16(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous Double Speed Mode (U2X = 1)	$BAUD = \frac{f_{OSC}}{8(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{8BAUD} - 1$
Synchronous Master Mode	$BAUD = \frac{f_{OSC}}{2(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{2BAUD} - 1$

Table 8. UBRR value calculation

- Now we program the serial frame to match the GM65's frame 1 start bit, 1 stop bit, and 8 data bits.
- Set the USART to UART (Asynchronous mode).
- Disable Parity bits.

We can find these bits inside USART Control and Status Register C – UCSRC.

7	6	5	4	3	2	1	0
URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	0	0	0	0	1	1	0

Figure 17. USART Control and Status Register C – UCSRC

- Options were set based on these tables inside the ATmega16's Datasheet:

UCSZ2	UCSZ1	UCSZ0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

Table 9. USART frame character size

USBS	Stop Bit(s)
0	1-bit
1	2-bit

Table 10. USART stop bits

UMSEL	Mode
0	Asynchronous Operation
1	Synchronous Operation

Table 11. USART modes

UPM1	UPM0	Parity Mode
0	0	Disabled
0	1	Reserved
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

Table 12. USART frame parity bit configuration

We will be generating this type of Bar code:

Character Set	Length	Check Digit	Size, Module Width X, Print Ratio
ASCII (128 characters) ISO-8859-1	variable	Mod 103	H>=15% of L (H>=6.5 mm!); X>= 0.19 mms (max: 1.27 mms); Pr= 1:2:3:4
Applications	Widely used in all areas; modern compact symbology; introduced 1981 by "Computer Identics"; in conjunction with FNC1 used as GS1-128 or UCC/EAN-128 for retail product marking		
Notes	3 different code sets (A=upper case + ASCII control characters, B=upper + lower case characters, C=double density numeric characters); code set switching; function code characters (FNC1-4); high printing density (laser or thermo transfer printer recommended);		

Figure 18. Bar Code ASCII format

By default, GM65 Supports all types.

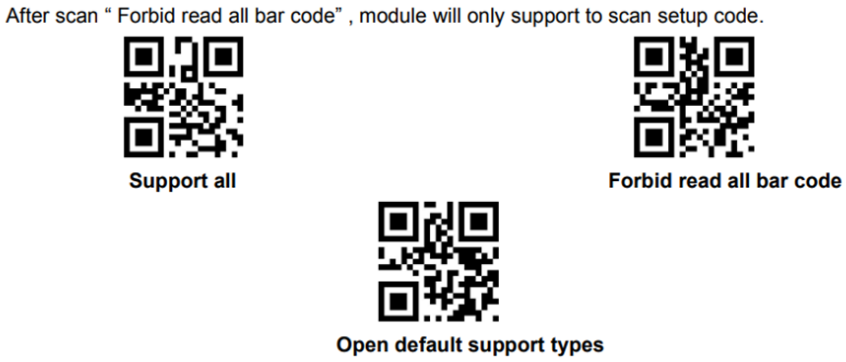


Figure 19. GM65 supports multiple ASCII formats

2.13) Objective of Embedded System

- Provide the smart cart user with a weight sensor to measure products.
- Send scanned Bar code from GM65 to server's database.
- Provide a security check routine on scanned item weight.
- Receive server's response and process it

2.14) Embedded System Block Diagram:

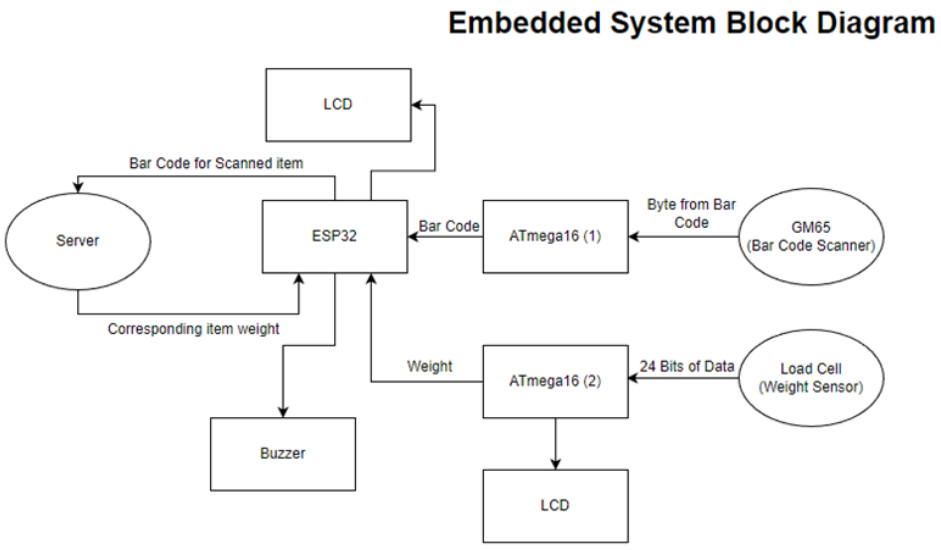


Figure 20. Embedded System Block Diagram

As the block diagram shows, the ESP32 will work as an interface between ATmega16(1), ATmega16(2) and the Server.

2.15) Steps taken on ESP32:

1. Set up connection with ATmega16(1), ATmega16(2).
2. Set up Wi-Fi connection.

```
#include <WiFi.h> /* To use Wifi */
#include <HTTPClient.h> /* to send and recieve from internet */
#include <ArduinoJson.h> /* to use JSON */
#include <LiquidCrystal.h> /* to use LCD */
```

Figure 21. Necessary libraries for internet connection

```
const char* ssid = [redacted];
const char* password = [redacted];
```

Figure 22. Necessary libraries for internet connection

3. Define Server Name.

```
const char* serverName = "https://smartcart-helwan.herokuapp.com/api/add_orderItems"; /* API */
```

Figure 23. Necessary libraries for internet connection

4. Receive data from ATmega16(1), ATmega16(2).
5. Start connection with server. (Server requires Authorization token to be able to post to it.)
6. Format data for transmission.
7. Send data.
8. Receive response.
9. Process response.

```
String requestBody;                                  /* String to store post request */
StaticJsonDocument<200> doc;                         /* JSON document to store request */
doc["barcode"] = Bar_Code;   Assign Bar Code to JSON Document
serializeJson(doc, requestBody);  Send data as JSON as the server desired this specific format

// Send HTTP POST request
Serial.println(requestBody);
int httpResponseCode = http.POST(requestBody);   Send Request
Serial.print("HTTP Response code: ");
Serial.println(httpResponseCode);
if (httpResponseCode > 0) {
  payload = http.getString();   Recieve server response
```

Figure 24. Necessary libraries for internet connection

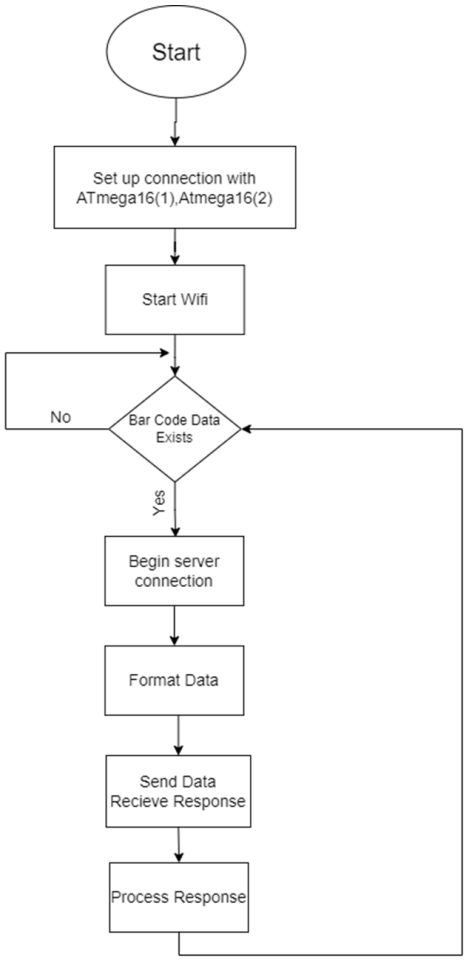


Figure 25. ESP32 program flow

2.16) Software Architecture

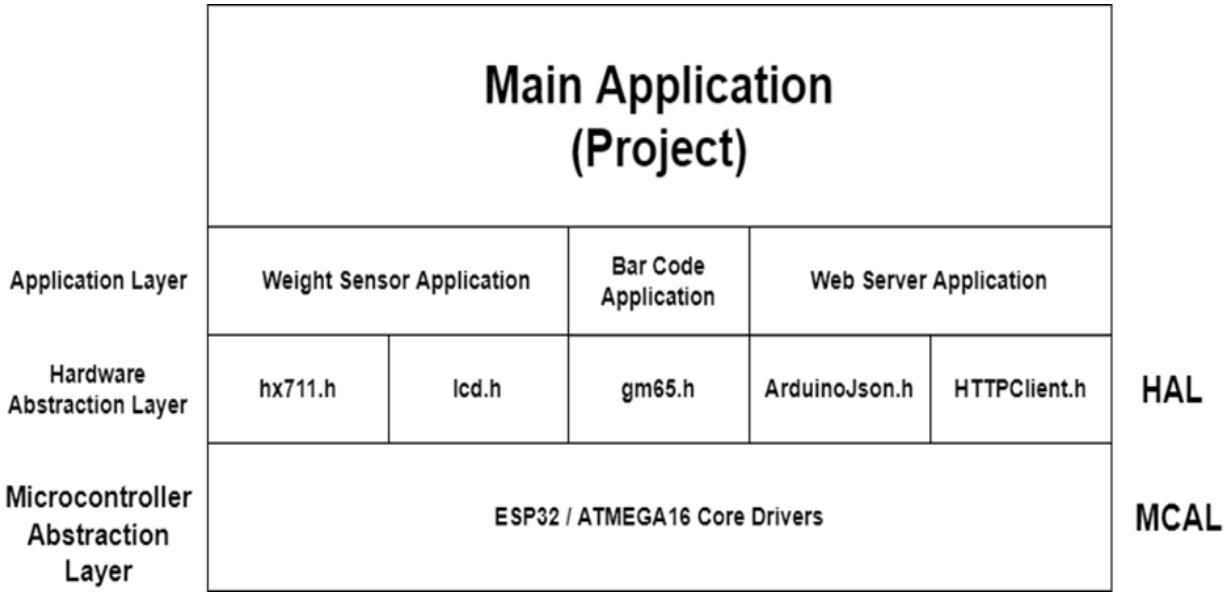


Figure 26. Software Architecture

Chapter 3. Front End

Front-End Web Development

3.1) What is Frontend?

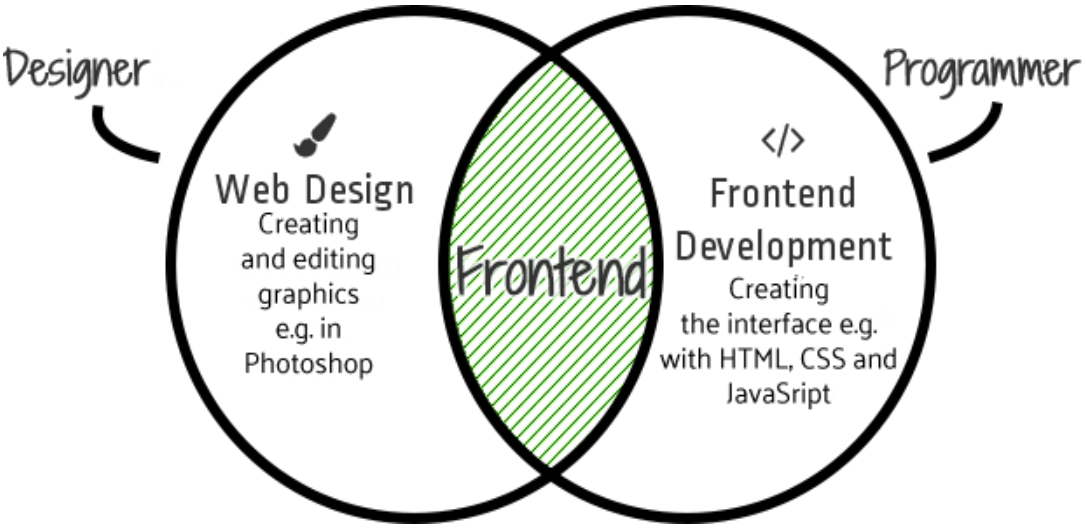


Figure 27. Front end

The frontend is the application's visible portion and is the presentation layer. Overall, it should be emphasized that when discussing a website, you must also discuss its front end. The fonts, colors, menus, text styles, buttons, links, tables, and so much more that you see on websites were all programmed and/or put together using html, CSS, and JavaScript (or any of its frameworks like react, angular or Vue) and are all components in the frontend of any website.

Frontend is sometimes referred to as "client side" and mainly consist of 2 parts, the user interface "UI" and the user experience "UX". When using a website, app, or other electronic device, you interact with screens, buttons, toggles, icons, and other visual components, which are referred to as user interfaces (UI). UX describes your overall experience using a product, including your feelings during that experience. Although UI may undoubtedly influence UX, the two are separate, as are the responsibilities that designers perform.

Strong UI and good UX are frequently necessary for creating products that people adore. For instance, you may have a banking app with a beautiful interface and simple navigation (UI). However, it doesn't matter how attractive the app is if it loads slowly or requires you to go

through multiple panels in order to transfer money (poor UX). Most likely, you won't want to deploy it.

On the other hand, a website may be stuffed full of original, useful information that is arranged logically and intuitively (perfect UX). However, you're more likely to leave a website if it appears outdated or if it's difficult for users to navigate between panels or browse through alternatives.

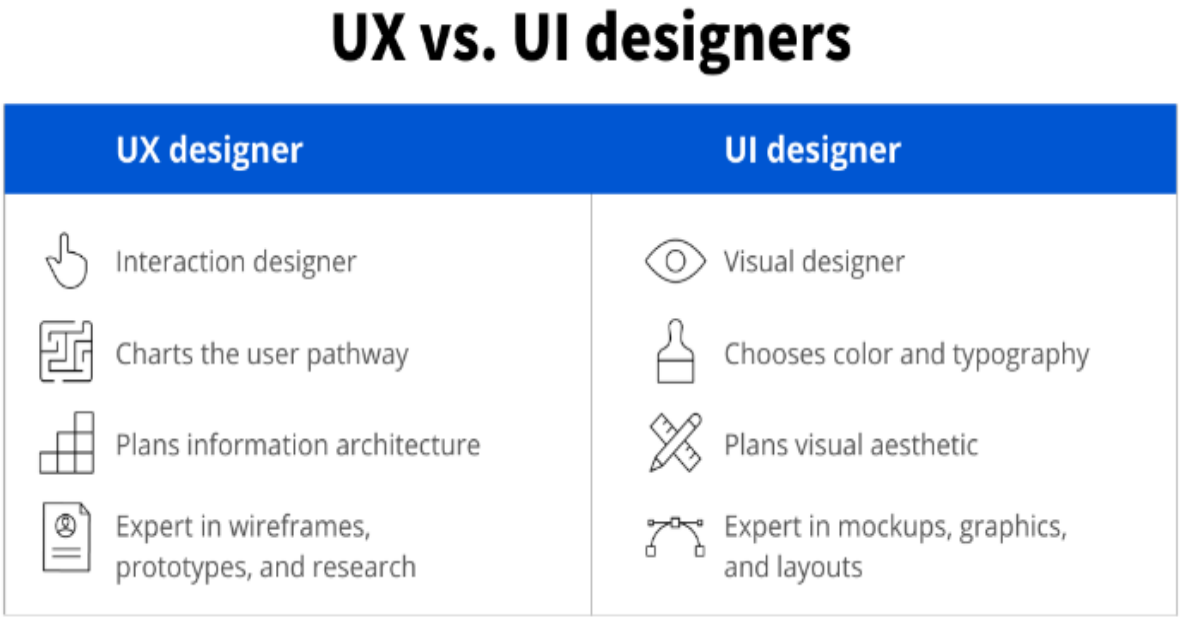


Figure 28. UI vs UX

3.2) Frontend and Backend

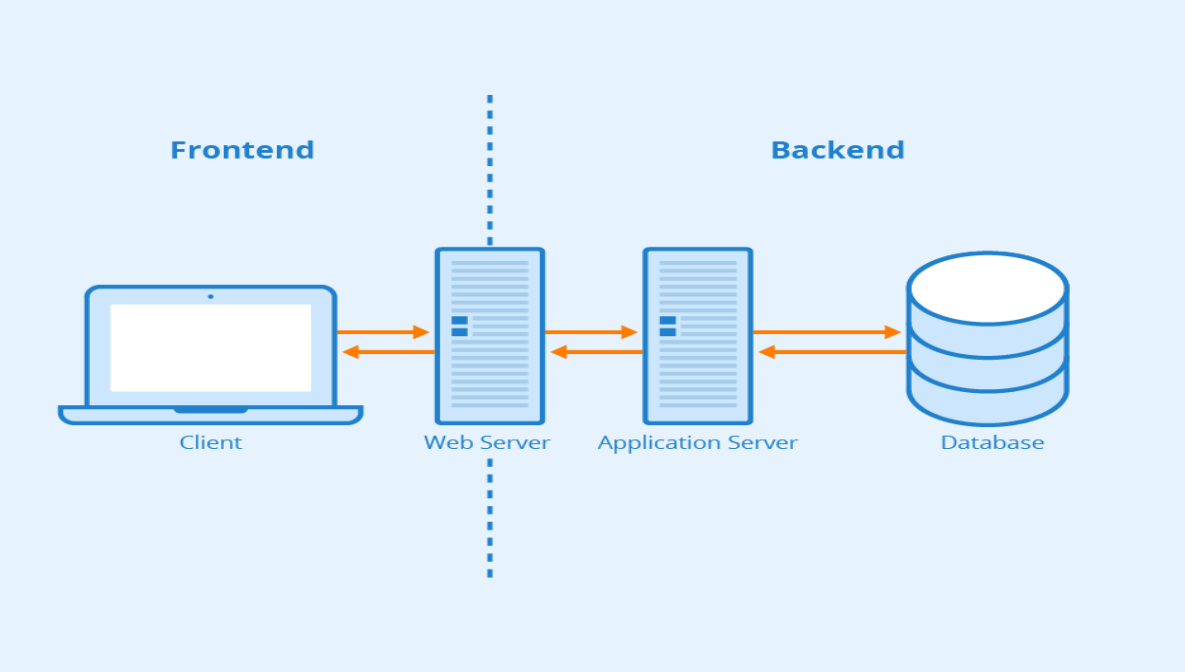


Figure 29. Frontend and Backend 1

In this thesis, there will be a whole chapter dedicated to explaining in detail the backend and server side, but this section serves as a small explanation of how backend can affect the frontend, and how this interaction will be used in our project.

Figure 30. Frontend and Backend 2

The data access layer is declared to be the backend. That indicates that it is the portion of a program that the user cannot see (unlike the frontend).

It includes the administration section and all of the application's code.

For instance, while visiting a website, the user interface, or frontend, is what one can see first. Users may communicate with the backend through the frontend. As soon as customers enter information, a database on a server receives it.

The backend, which handles the implementation of the functions, is necessary to make an application functioning. The code for the background-running functions is hidden from you, the user. (For more information in depth and better details please refer to the Backend Chapter).

Without backend, a website (composed only of frontend components) would be like a picture or a drawing, without real usable functionalities or user interfacing capabilities.

For example, a good-looking login or signup form needs to refer to the backend to either store new users or check the logging in users' credentials and get their data, otherwise (without the backend) it is simply a collection of input fields (username, password, email, ...) which does nothing when filled.

Our project will utilize this communication capabilities to provide users with a good smooth UI, with all the needed functionalities to ensure a good UX.

In our project, the connection between frontend (React JS) and backend is mainly done by using API calls, using the POST and GET methods to send and receive data from the website to the server (database) "more explanation on this topic will be given later on in this chapter".

3.3) HTML

Using elements, tags, and attributes, HTML, or Hypertext Markup Language, enables online users to design and organize sections, paragraphs, divisions, and connections. It's important to

remember, nevertheless, that HTML is not regarded as a programming language because it cannot develop dynamic functionality.

There are many applications for HTML, including:

To control how text, hyperlinks, and media files are displayed by browsers, developers employ HTML code.

Surfing the internet. Since HTML is widely used to include hyperlinks, users may explore and insert connections between relevant pages and websites with ease.

Web-based instruction. Similar to Microsoft Word, HTML allows for document organization and formatting.

It's also important to note that HTML is now regarded as a legitimate online standard. The HTML standards are developed and updated frequently by the World Wide Web Consortium (W3C).

The fundamentals of HTML will be covered in this essay, along with how it functions, its benefits and drawbacks, and how it interacts with CSS and JavaScript.

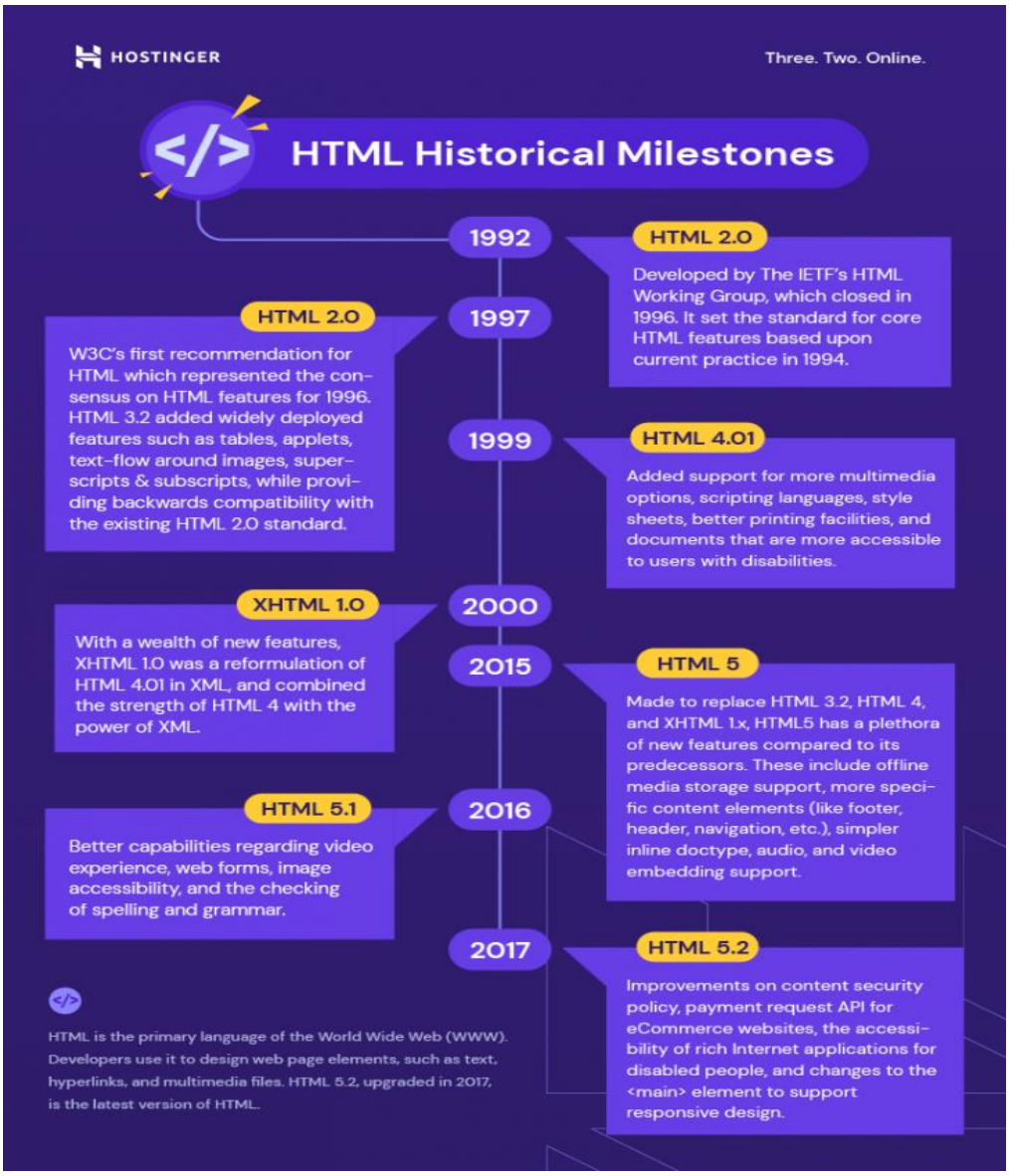


Figure 31. History of HTML

The typical website has a variety of HTML pages. A home page, an about page, and a contact page, for instance, would all have their own HTML files.

Files with the .html or .htm extension are HTML documents. Internet users may explore the content of an HTML file by reading the file in a web browser after it is rendered (by the browser).

HTML elements are a collection of tags and attributes that are present in every HTML page. The building components of a web page are HTML elements. While an attribute defines an element's properties, a tag instructs the web browser where an element starts and stops.

And when combining these components, we get an element, for example:

```
<p>This is how you add a paragraph in HTML. </p>
```

This is called a P or paragraph tag, which inserts a simple plain paragraph "which can be targeted and styled later using CSS (more on that later)"

Another example is:

```
<h1>This is how you add the largest heading in HTML. </h1>
<h6>This is how you add the smallest heading in HTML. </h6>
```

And this is how one can add an image:

`<img src="/" alt="Image">` "where src is the source of the image (which could be a local file or a URL) and alt is what shows if the image was failed to be loaded"

And finally, this is how to add a link tag:

```
<a href="https://example.com/">Click me! </a>
```

As it can be seen now, a HTML tag (element) is consisted of 3 main parts:

1) An opening tag " " indicating where an element starts.
2) The content, which is the output that a user sees.
3) A closing tag " " indicating where the element ends.

Example:

Start tag	Element content	End tag
<h1>	My First Heading	</h1>
	My first paragraph.	
 	none	*none*

Figure 32. HTML tag components

Which will be written in this form:

```
<html>
  <head>
    <title>Page title</title>
  </head>
  <body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
    <p>This is another paragraph.</p>
  </body>
</html>
```

Figure 33. HTML file components

And will look like this in the browser:

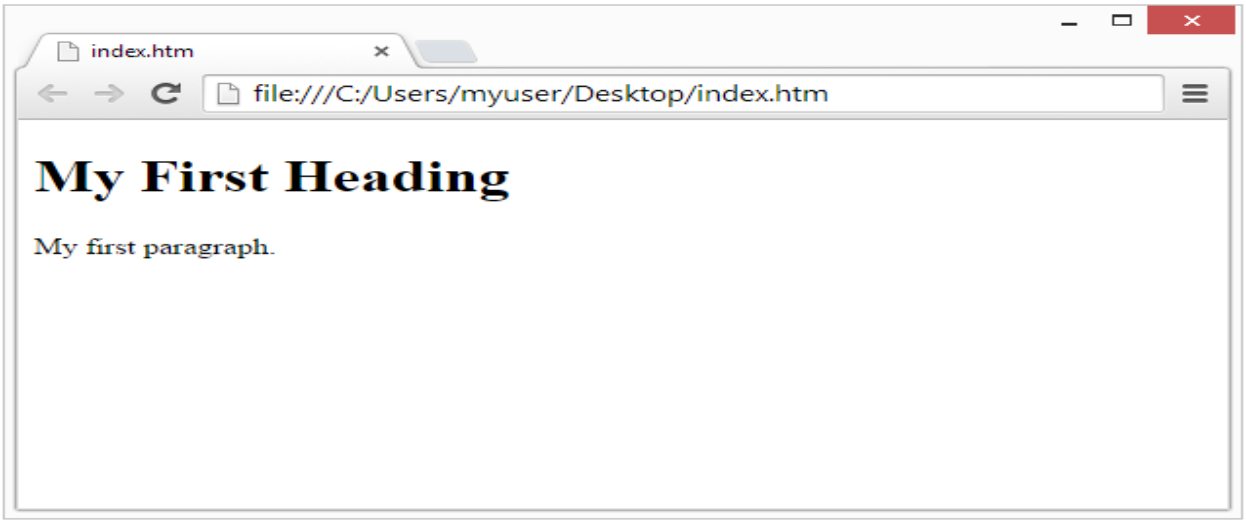


Figure 34. HTML file in a browser

There are now 142 HTML tags that may be used to create different types of elements.

So, by using a combination of these tags and mixing them and nesting them inside each other, one can create any website structure as required.

Just like any other language (or anything in life) the HTML has its Pros and Cons, some of its Pros are:

1) Beginner-friendly. HTML provides a short learning curve and simple, uniform markup.
2) Support. The language has a huge (helpful) community, a lot of resources, and is extensively utilized.
3) Accessibility. It is totally free and open-source. All web browsers support HTML properly.
4) Versatile. It is simple to integrate HTML with backend programming languages like PHP and Node.js.

While some of its drawbacks are:

1) Static. Static web pages are the main use for the language. You might need to employ JavaScript or a back-end language like PHP for dynamic capabilities.
2) Distinct HTML page. Even if the parts are the same, users still have to develop unique web pages for HTML.
3) compatibility with browsers. Some browsers take time to integrate new functionalities. Newer tags may not always be shown by older browsers.

In conclusion, HTML is the perfect language to construct the skeleton of a webpage, but in short, is not sufficient on its own, that's why in this project it was used alongside other languages to produce a well-designed, good looking and dynamic web page, and in the next sections, the other languages used in this project will be used to provide a better understanding of the final webpage.

3.4) CSS

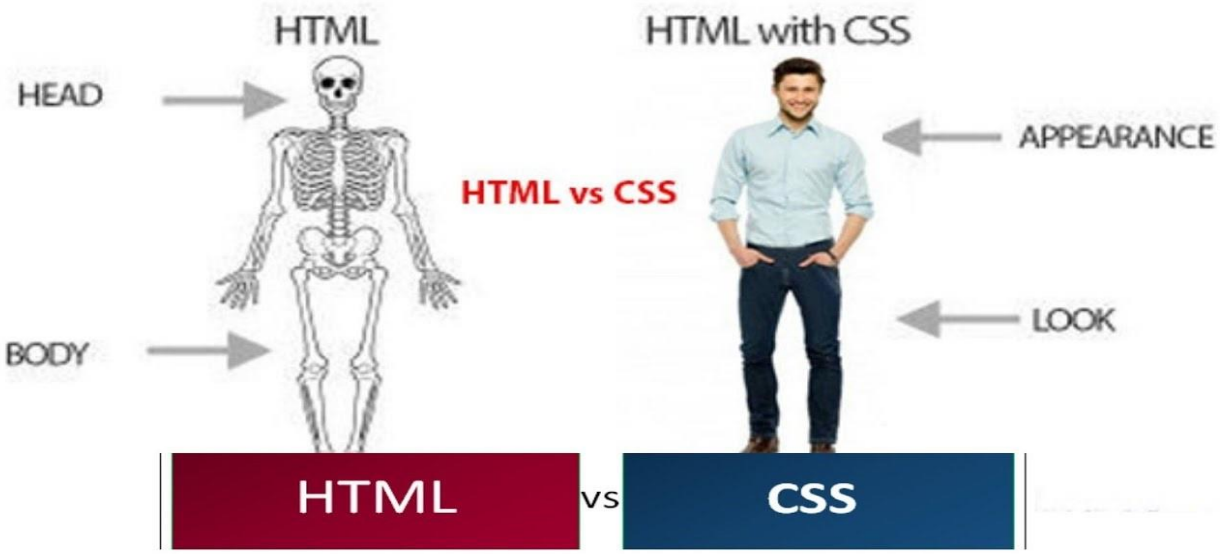


Figure 35. HTML and CSS

From the previous section, one can understand that HTML can be described as the skeleton of a website, and using the same description, CSS can be described as the skin (and looks) of a website, being on top of HTML and giving it a unique (good) look.

A programmer can Create attractive web pages with CSS (Cascading Style Sheets). A language called CSS is used to describe how documents are presented to users, including how they should be formatted and organized. Simple text styling for documents using CSS is possible, such as altering the size and color of headers and links. It is possible to utilize it to design a layout, such as transforming a single column of text into one with a primary content section and a sidebar for supplementary information. It may even be employed for animation effects.

CSS is a rule-based language; thus, you create the rules by naming sets of styles that should be used on certain web page components or sets of elements.

For instance, you may select to display your page's primary header as a block of huge red font.

HTML WAS NEVER Meant to have tags for web page layout.

HTML was developed to define a web page's contents. For web developers, a nightmare began when elements like "<font>" and "<color>" features were added to the HTML 3.2 specification.

The creation of huge websites became a time-consuming and expensive procedure since font and color information was added individually to every page.

The World Wide Web Consortium (W3C) developed CSS to address this issue.

The HTML page's style formatting was removed by CSS!

Normally, the style definitions are kept in separate .css files.

One may alter only one external CSS file to modify the appearance of a whole website!

Additionally, CSS can target specific elements or parts using their "ID" to give them unique separate styles.

For example, the following css code can be used to make all <h1> tags have red font and a specific font size, and all the <p> tags have black font color.

```
h1 {
   color: red;
   font-size: 5em;
}


p {
   color: black;
}
```

Additionally, CSS allows developers to add inline styles (inside the HTML opening tag), which will have higher priority and overrule any general css rule, for example:

```
<p style="color: purple; font-family: verdana">This is how you
add a paragraph in HTML. </p>
```

Making CSS very versatile and flexible, and favored by developers to this day.

Also, CSS can be used to add special event effect, like changing color on hover or making elements visible or not visible, it can even be used to add animation and changing effects.

Finally, the following figure shows how a website would look if it was implemented using only HTML (right) VS when it was implemented using HTML and CSS (left).

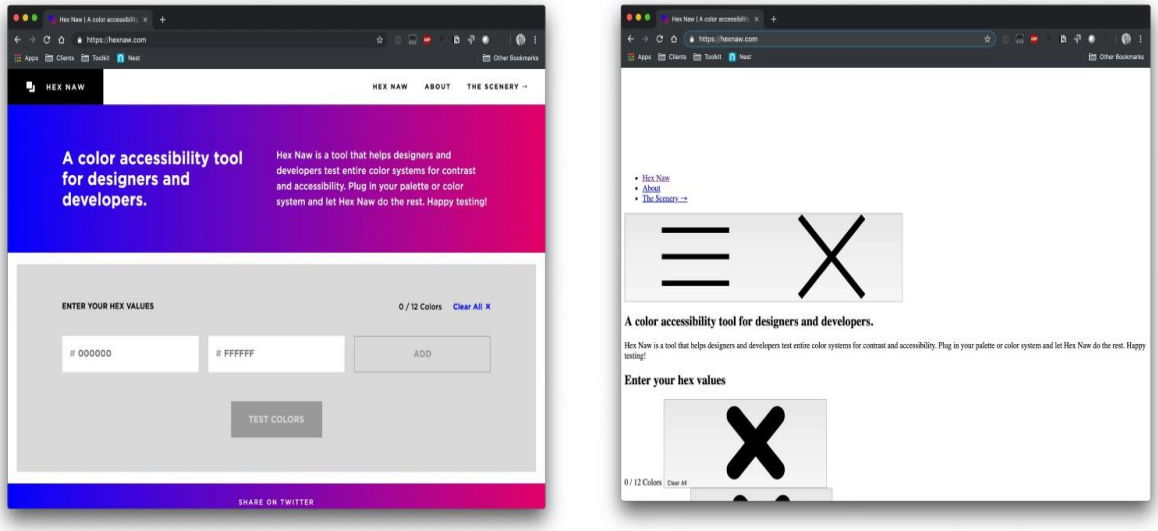


Figure 36. Site with vs without CSS

3.5) JavaScript

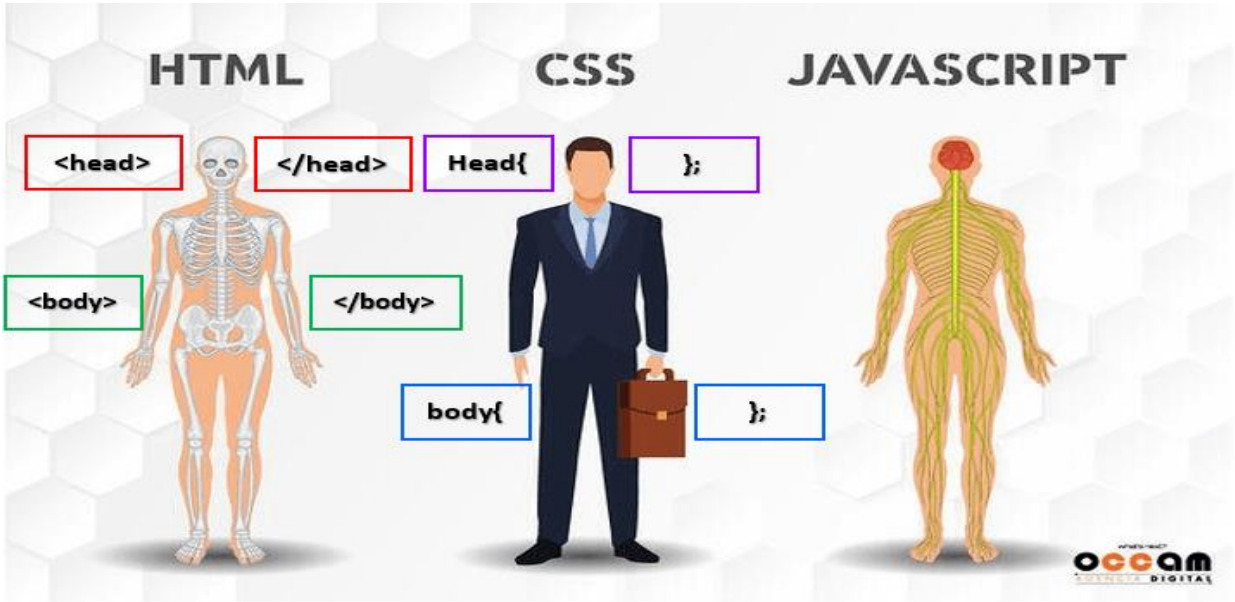


Figure 37. HTML, CSS, and JavaScript

Using the same descriptions as before, if HTML is the skeleton and CSS is the skin and looks, then JavaScript is undoubtedly the brain and nervous system of any website (a very common use of JavaScript is to dynamically modify HTML and CSS), without JavaScript a website (designed only with HTML and CSS) would be like a beautiful (yet lifeless) painting, having minimum to no functionality and user interactions, unable to respond or communicate with a server or a backend, or show alerts and messages to a user or deal with events (such as clicking on a submit button).

So, what is JavaScript?

A compact, interpreted, or just-in-time (Jit), mainly client-side, compiled programming language containing first-class functions is called JavaScript (JS)

“When functions in a programming language are regarded the same as variables, this is referred to as having first-class functions in that language. For instance, in such a framework, a function can be assigned as a value to a variable, supplied as an input to other functions, and returned by another function”

that allows developers to implement complex features.

Many non-browser contexts, like Node.js, Apache CouchDB, and Adobe Acrobat, employ it even though it is best recognized as the scripting language for Web pages. JavaScript is a dynamic, prototype-based language that supports object-oriented, declarative (like functional programming), and imperative programming paradigms.

The ECMAScript Language Definitions (ECMA-262) and the ECMAScript Internationalization API specification are the standards for JavaScript (ECMA-402). The most recent draught versions of ECMA-262 and ECMA-402 provide the foundation for the JavaScript documentation found throughout MDN.

JavaScript should not be confused with the Java programming language. Java and JavaScript are both registered trademarks or brands of Oracle in the United States and other countries. The syntax, semantics, and applications of the two programming languages, however, are significantly dissimilar.

JavaScript is a programming language that enables you to manipulate multimedia, animate graphics, and generate dynamically changing content. You can assume that JavaScript is probably used whenever a web page displays anything other than static information for you to look at, such as timely content updates, interactive maps, animated 2D/3D visuals, scrolling video jukeboxes, etc.

Generally, a JavaScript code is compiled in its written order, from top to bottom, so a programmer should be careful of when and where to declare or call functions and variables to avoid errors.

What can JavaScript do?

Store important values inside variables, Operations on pieces of text (referred to as "strings" in programming) “such as concatenating them”, Running code in response to certain events occurring on a web page (such as responding to a click event on a certain element).

However, the capabilities based on the client-side JavaScript language is much more intriguing. Developers have access to additional superpowers through so-called Application Programming Interfaces (APIs), which they may utilize in any JavaScript code.

APIs are pre-built collections of coding components that enable developers to create applications that would otherwise be challenging or impossible to create. It is much simpler to take pre-cut panels and screw them together to make a bookshelf than it is to work out the design yourself, cut all the boards to the correct size and shape, find matching screws, and put them together to make a bookshelf. They function similarly for programming as ready-made furniture kits do for home construction.

For example, APIs allow programmers to dynamically deploy new styles to the page, create, remove, and change HTML, as well as edit CSS. As well as any pop-up window on a page or new contents being shown, also they can retrieve geographical information (to keep google maps updated for example), create 2-D and even 3-D animation, and finally playing audio and video (any multimedia) right in a web page.

3.6) React JS

Our website was created using React JS, so this section aims to define React and clarify why it was chosen to build our website.

To begin, what is React JS?

React is a User Interface (UI) library, created by Facebook as a tool to build reusable UI components, and build single-page applications.

Today, React JS is regarded as the most popular front-end JavaScript library used to construct Web applications, React.js is used by most Fortune 500 firms.

React.js is an open-source JavaScript library, it manages the view layer for both online and mobile applications. Jordan Walke, a software developer for Facebook, developed React at first. In 2011 and 2012, Facebook's newsfeed and Instagram.com both used React for the first time.

With the aid of React, programmers may build substantial online apps that can modify data without refreshing the page. React's primary goals are to be quick, scalable, flexible, and easy to use. It only functions with the application's user interfaces. This relates to the MVC template's view. It may be used with other JavaScript frameworks or libraries, such as Angular JS in MVC, or using it alongside node.js (backend) as a full stacked web.

Here are some of the reasons for the huge (and growing) popularity of React:

1) React.js is declarative
2) React.js is easy to use.
3) React.js is flexible and versatile.
4) React.js is component based
5) React.js supports server side
6) React.js is a big library.
7) Learning React.js is simple.
8) React.js is fast.
9) React applications are very easy to test

React utilizes JSX for templating rather than standard JavaScript. JSX is a straightforward JavaScript that supports HTML quoting and displays subcomponents using this HTML tag

syntax. React Framework JavaScript calls are generated from HTML syntax. We may also use the original JavaScript language.

React employs one-way data binding, and the Flux application architecture manages the data flow to components through a single control point known as the dispatcher. Large ReactJS apps with independent components can be more easily fixed.

Due to all these reasons, and many more, after studying HTML, CSS and JavaScript, we decided to design our website using React.

In the next section, we will look more in depth to the code, reasoning and functionality, of our website, and how each element of the previously explained components was used to produce and good looking and well-functioning website, and how were these components integrated together.

Finally, as previously (and briefly) mentioned the connection between Frontend and backend is done by API calls, which was an additional reason for choosing React js, since react allows for a simple and smooth integration between the 2 sides, making it easier and faster to send and receive data, and further improving the user experience

3.7) The Project

As of this moment, our website is consisted of several pages

- Home page
- About us
- Purchase History
- Current Cart
- User Profile
- Login / Signup

With 2 permeant components visible in all pages

- Navbar
- Footer

3.7.1) File structure

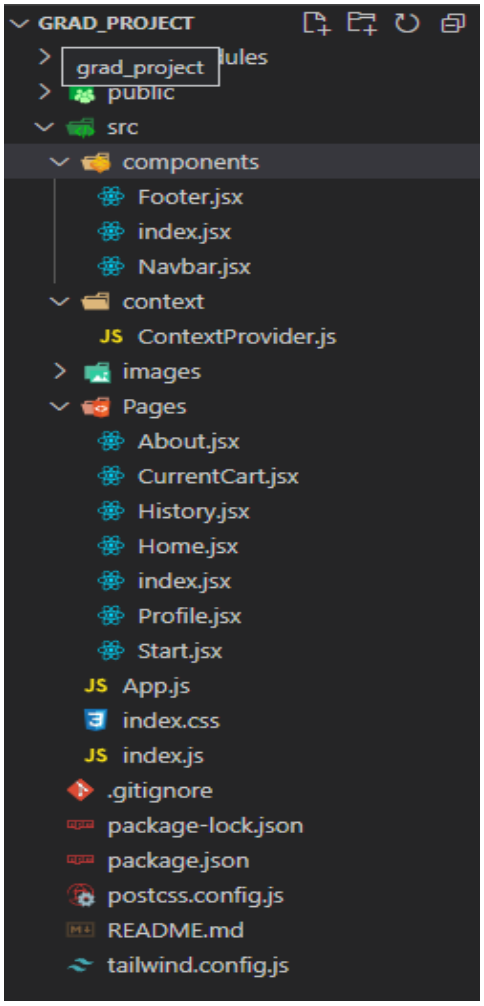


Figure 38. File structure

This is the folder structure, everything is neatly arranged in a relevant folder, the Navbar and footer in a components folder as they will be shared between all pages, and then every page having its own .jsx file in the pages folder, making it organized, and easy to call any components anywhere when needed.

The ContextProvider is a unique file (perks of using React JS) in the context folder that allows us to store all the states of certain variable to be shared between all pages and updated by anyone of them, for example a variable called “isLoggedIn” keeps the state of the current page, if the user is not logged in “a guest” then this variable carries the value (false) “by default” while when a user logs in successfully its value will change to (true), on itself this variable will do nothing, but when utilized with other components, the website can become dynamic, so for example the looks of the Navbar will changed based on the value in this variable, allowing logged in users to access additional features.

src > context > JS ContextProvider.js > ContextProvider

```
1   import React, { createContext, useContext, useState } from 'react';
2
3   const StateContext = createContext();
4
5   const initialUser = {
6     email:'',
7     token:''
8   };
9
10    export const ContextProvider = ({ children }) => {
11      const [screenSize, setScreenSize] = useState(undefined);
12
13      const [isloggedin, setIsLoggedIn] = useState(false);
14      const [user, setUser] = useState(initialUser);
15
16
17
18
19      return (
20        // eslint-disable-next-line react/jsx-no-constructed-context-values
21        <StateContext.Provider value={{ isloggedin, setIsLoggedIn, screenSize, setScreenSize, user, setUser }}>
22          {children}
23        </StateContext.Provider>
24      );
25    };
26
27    export const useStateContext = () => useContext(StateContext);
```

Figure 39. Context file

3.7.2) Navbar

As mentioned, the Navbar allows users to navigate between different pages, and it dynamically changes when the user loges in.

This first figure (40) shows how the Navbar looks when the user is not logged in:

Figure 40.Guest Navbar

And this figure (41) shows how the Navbar will look (additional functionalities) once the user loges in:

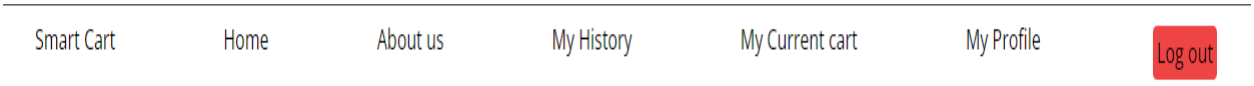


Figure 41. Client Navbar

3.7.3) Home page

At the moment, the homepage “or landing page” is used to provide the user with relative information about our store and products, this page will be available to all users “logged in or not” as it gives general information that any user can have access to, especially a new guest that wants to be familiar with the store.

The following figures (42, 43, 44, 45) show the current design of the homepage and the features it shows the user/guest.

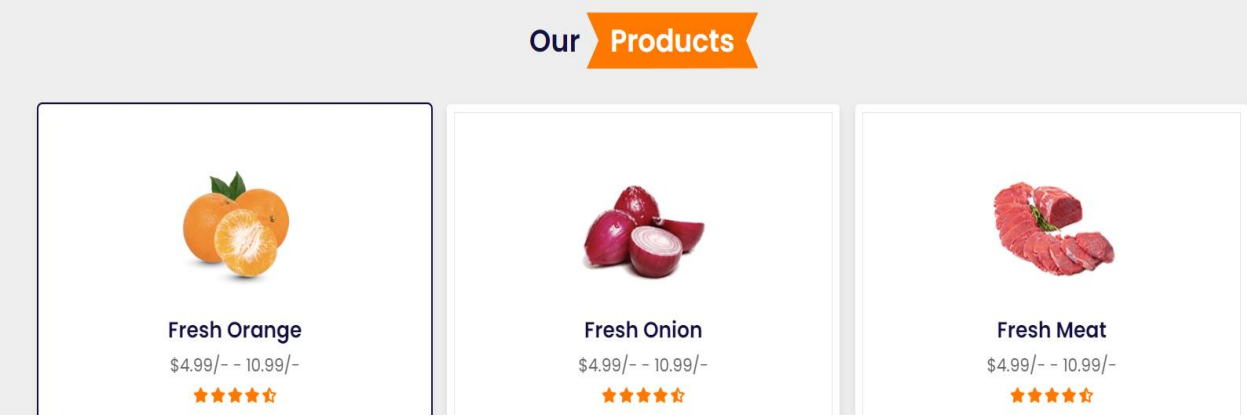


Figure 42. Homepage 1

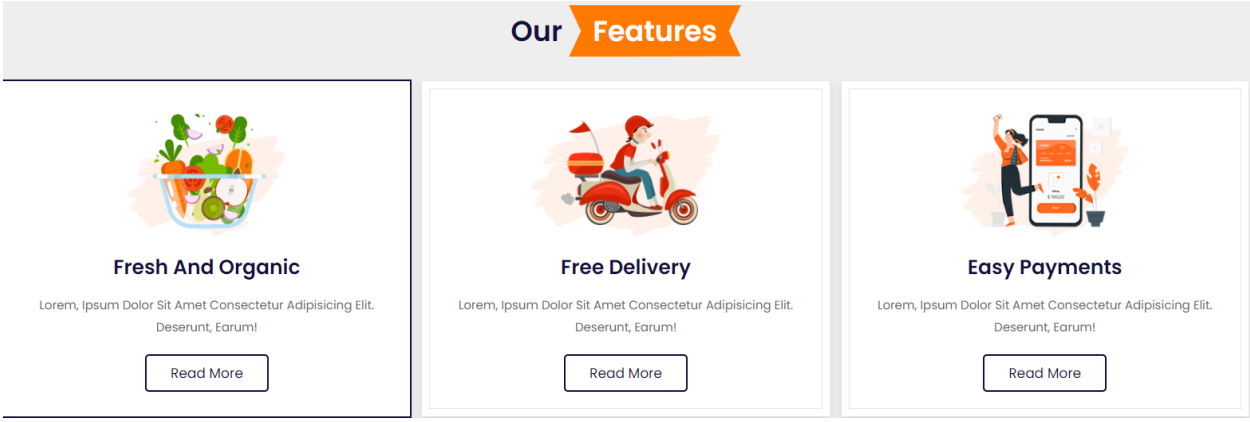


Figure 43. Homepage 2

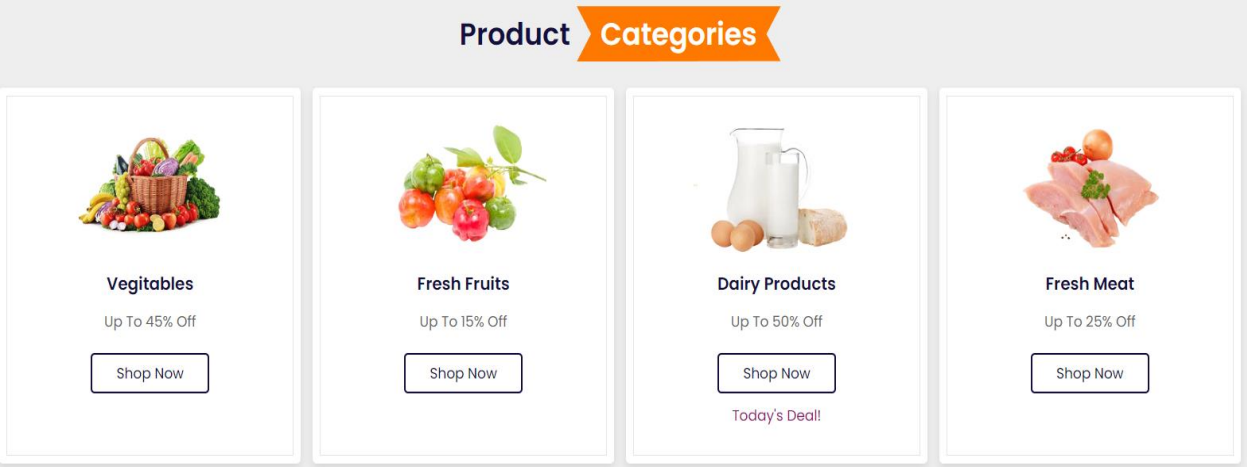


Figure 44. Homepage 3

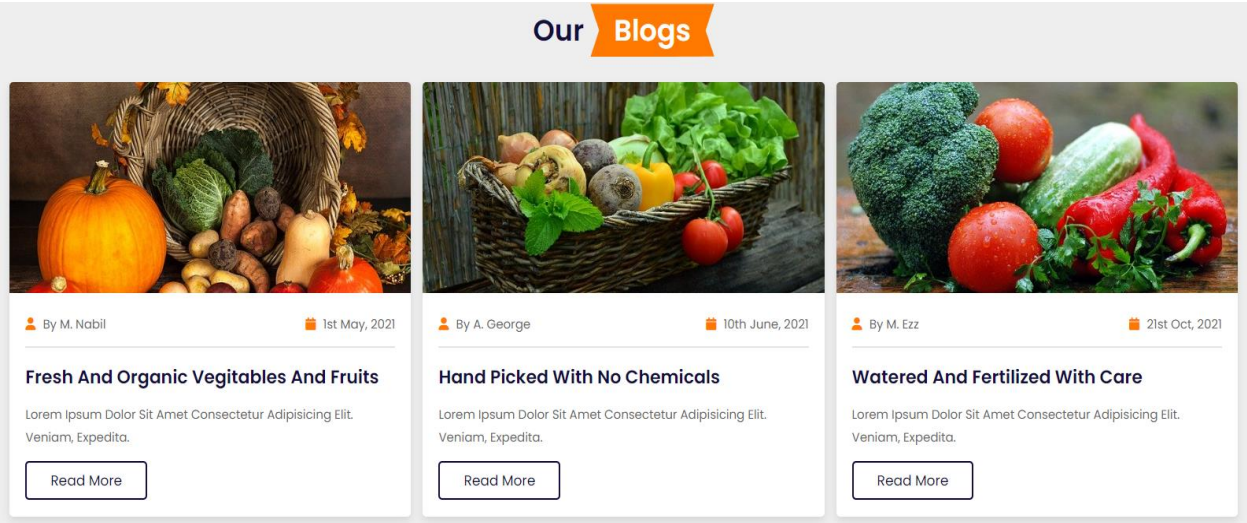


Figure 45. Homepage 4

3.7.4) About us

The about page “still under construction at the moment” will be also available to all visitors (logged in users and non-logged guests), and it will provide 2 main things

1) An explanation of how a customer “in the store” can connect and use a cart, and how this process works, explaining all the possible scenarios and how to deal with every one of them.
2) A brief introduction about the engineering team that worked on this project.

3.7.5) Purchase History

This page allows the user to see all the purchase history, so they are aware of all the transactions that happened, additionally they can access the purchase list of a specific day to repurchase these items without the need to make a new purchase list.

3.7.6) Current Cart

This page allows a user to do 2 things:

First, to reserve and connect to a cart to start shopping.

And second, after connecting to a cart it allows the user to see the items currently available in the cart.

3.7.7) User Profile

In this page a user can see all profile related data (including the current balance of this account) and edit them.

3.7.8) Login – Signup

When a user clicks on “Get Started” button on the navbar, they are taken to the form in figure 46 by default, it is the log in form as the user will log in on a regular basis but signup only once, a user enters his/her username or email and password, then an API call is made to the server to check these credentials, if they are correct then the user is redirected to the homepage, all the necessary variables are adjusted and he/she is now properly logged in.

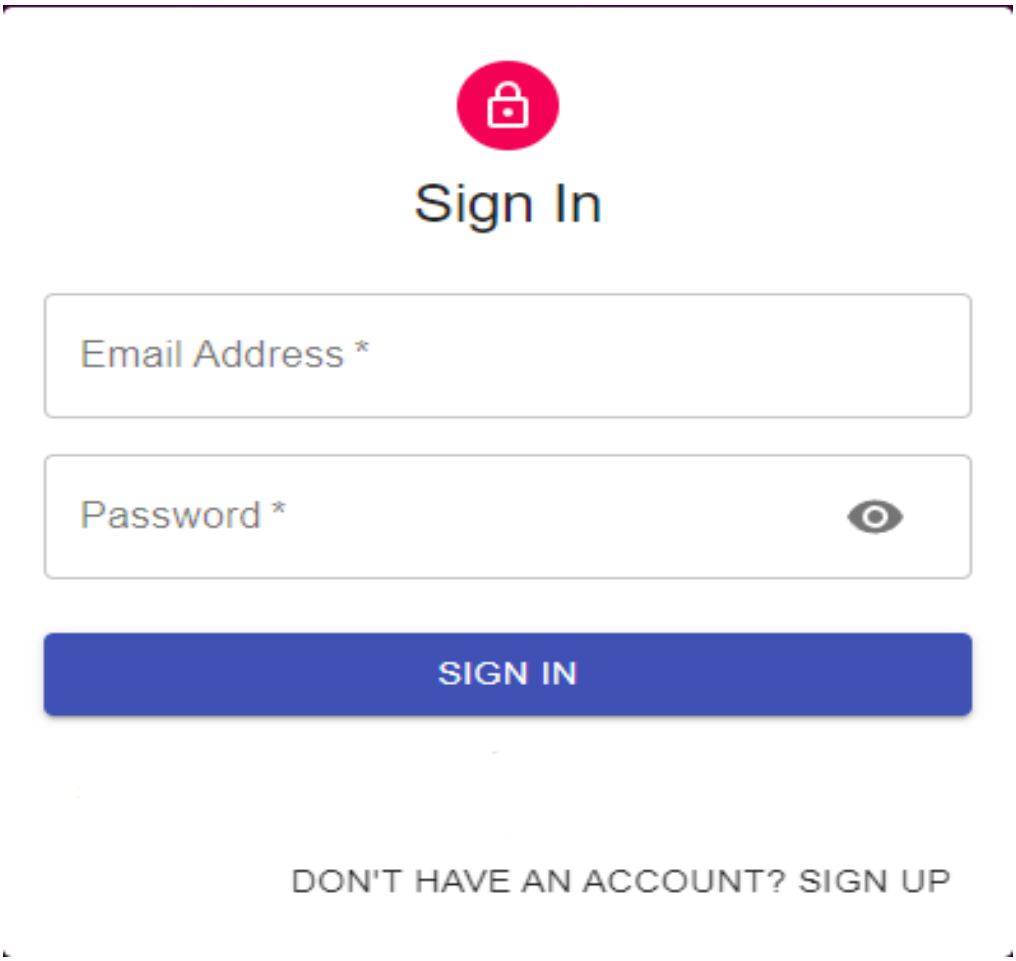


Figure 46. Sign in Form

If the user is new (which will normally happen once to each user), clicking on “Don’t have an account” will change the form to signup (as shown in figure 47), allowing the user to enter his information to create an account (add his data to the database), then use the newly created account to log in and access all available features.

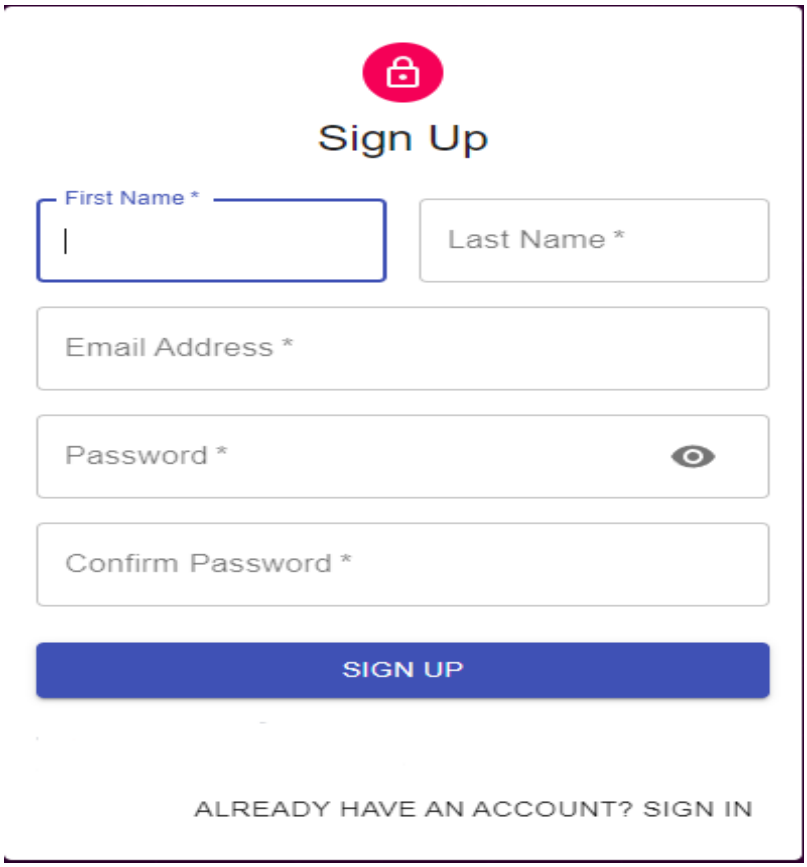


Figure 47. Signup Form

And finally, here is an example (figure 48) of a successful login then logout attempt, showing the different responses given by the server

The first line is the data (username and password) inserted by the user and sent to the backend, after being revised the backend responds with the user token and email “which are locally stored” indicating a successful login, then the user is redirected to the homepage. After being done and the user wants to logout, clicking on the logout button (in the Navbar) loges him/her out, and when this operation is done smoothly with no errors the user gets a message “last line” informing him that the logout was successful, and all the necessary safety procedures were made to protect his account and avoid any errors.

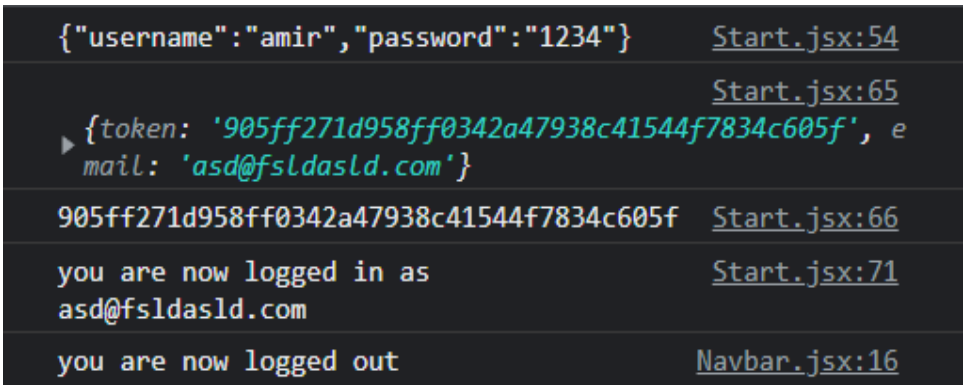


Figure 48. API response

3.7.9) Footer

As shown below (in figure 49), the footer provides users with some relevant additional information, like different links and components on our website, and also contact information, and a link to any external social media account, and finally the rights reserved at the bottom and the Logo and name of the project at the top.

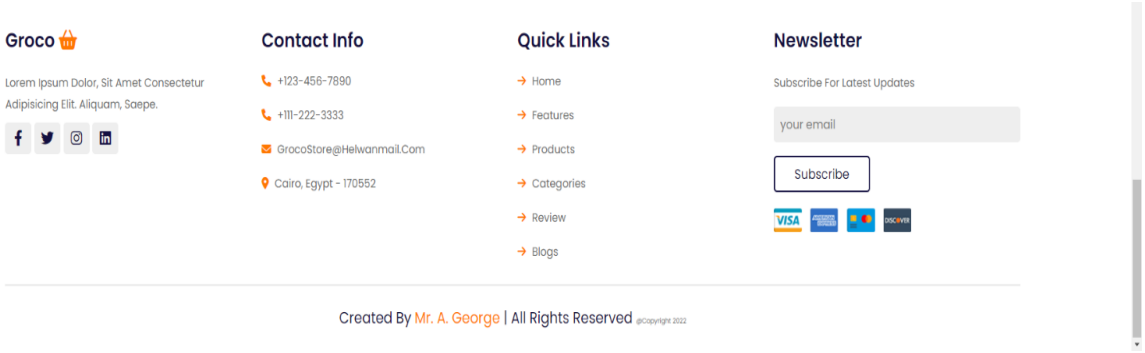


Figure 49. Footer

Chapter 4. Back End

4.1) Database

A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS). Together, the data and the DBMS, along with the applications that are associated with them, are referred to as a database system, often shortened to just database.

Data within the most common types of databases in operation today is typically modeled in rows and columns in a series of tables to make processing and data querying efficient. The data can then be easily accessed, managed, modified, updated, controlled, and organized. Most databases use structured query language (SQL)

4.1.1) Types of databases

There are many different types of databases. The best database for a specific organization depends on how the organization intends to use the data.

4.1.1.1) Relational databases

Relational databases became dominant in the 1980s. Items in a relational database are organized as a set of tables with columns and rows. Relational database technology provides the most efficient and flexible way to access structured information.

4.1.1.2) Object-oriented databases

- Information in an object-oriented database is represented in the form of objects, as in object-oriented programming.

4.1.1.3) Data warehouses

- A central repository for data, a data warehouse is a type of database specifically designed for fast query and analysis.

4.1.1.4) NoSQL

- NoSQL or nonrelational database, allows unstructured and semi structured data to be stored and manipulated (in contrast to a relational database, which defines how all data inserted into the database must be composed). NoSQL databases grew popular as web applications became more common and more complex.

We used in our project a relational database

4.2) Entity Relationship Diagram (ERD)

ERD stands for entity relationship diagram. People also call these types of diagrams ER diagrams and Entity Relationship Models. An ERD visulizes the relationships between entities like people, things, or concepts in a database. An ERD will also often visualize the attributes of these entities.

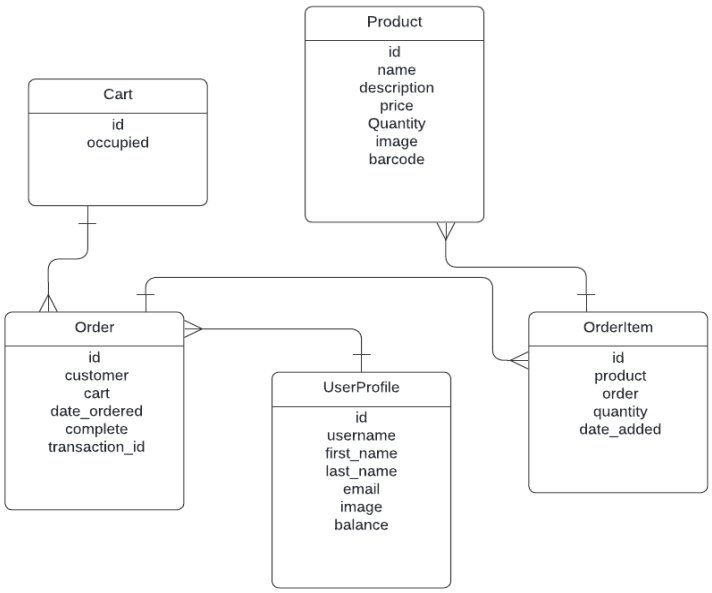


Figure 50.

4.3) Object-relational mapping

Object-relational mapping (ORM) is a programming technique in which a metadata descriptor is used to connect object code to a relational database. Object code is written in object-oriented programming (OOP) languages such as Java or C#. ORM converts data between type systems that are unable to coexist within relational databases and OOP languages.

ORM resolves the object code and relational database mismatch with three approaches: bottom up, top-down and meet in the middle. Each approach has its share of benefits and drawbacks. When selecting the best software solution, developers must fully understand the environment and design requirements.

In addition to the data access technique, ORM's benefits also include:

- Simplified development because it automates object-to-table and table-to-object conversion, resulting in lower development and maintenance costs
- Less code compared to embedded SQL and handwritten stored procedures
- Transparent object caching in the application tier, improving system performance
- An optimized solution making an application faster and easier to maintain

ORM's emergence in multiple application development has created disagreement among experts. Key concerns are that ORM does not perform well and that stored procedures might be a better solution. In addition, ORM dependence may result in poorly designed databases in certain circumstances.

```
class Product(models.Model):
    name = models.CharField(max_length=256)
    description = models.TextField()
    price = models.DecimalField(max_digits=7, decimal_places=2)
    Quantity = models.IntegerField(default=0)
    image = models.ImageField(upload_to="products")
    barcode = models.CharField(max_length=256, blank=True, null=False)

    def __str__(self):
        return self.name
```

```
class UserProfile(AbstractUser):
    image = models.ImageField(upload_to="users",
                              blank=True)
    balance = models.DecimalField(default=0.0,
                                  max_digits=99,
                                  decimal_places=3)
```

```
class Cart(models.Model):
    occupied = models.BooleanField(default=False)
```

```
class Order(models.Model):
    customer = models.ForeignKey(settings.AUTH_USER_MODEL,
                                 on_delete=models.CASCADE,
                                 null=True,
                                 blank=True)
    cart = models.ForeignKey(Cart,
                             on_delete=models.CASCADE,
                             null=True,
                             blank=True)
    date_ordered = models.DateTimeField(auto_now_add=True)
    complete = models.BooleanField(default=False)
    transaction_id = models.CharField(max_length=100, null=True)

    def __str__(self):
        return str(self.id)
```

```
class OrderItem(models.Model):
    product = models.ForeignKey(Product, on_delete=models.CASCADE,
                        related_name="product", null=True)
    order = models.ForeignKey(Order, related_name="orderItems",
                        on_delete=models.SET_NULL,null=True)
    quantity = models.IntegerField(default=0, null=True, blank=True)
    date_added = models.DateTimeField(auto_now_add=True)
```

```
@receiver(post_save, sender=settings.AUTH_USER_MODEL)
def create_auth_token(sender, instance=None, created=False, **kwargs):
    if created:
        Token.objects.create(user=instance)
```

4.4) What is an API?

Application Programming Interface (API) is a software interface that allows two applications to interact with each other without any user intervention. API is a collection of software functions and procedures. In simple terms, API means a software code that can be accessed or executed. API is defined as a code that helps two different software's to communicate and exchange data with each other.

It offers products or services to communicate with other products and services without having to know how they're implemented.

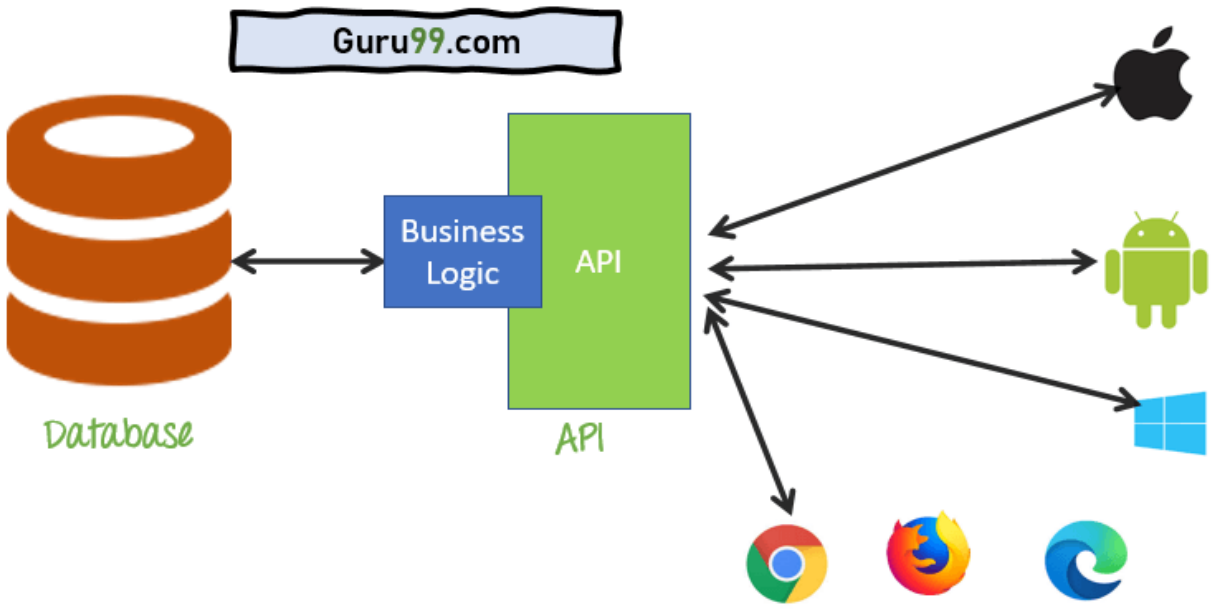


Figure 51.

Here are some reasons for using API:

- Application Programming Interface acronym API helps two different software's to communicate and exchange data with each other.
- It helps you to embed content from any site or application more efficiently.
- APIs can access app components. The delivery of services and information is more flexible.
- Content generated can be published automatically.
- It allows the user or a company to customize the content and services which they use the most.
- Software needs to change over time, and APIs help to anticipate changes.

Here are some key features of API:

- It offers a valuable service (data, function, audience...).
- It helps you to plan a business model.
- Simple, flexible, quickly adopted.
- Managed and measured.
- Offers great developer support.

Here comes Django and DjangoRestFramwork to make Apis

4.5) Django

Django is a free and open-source web application framework written in Python. A framework is nothing more than a collection of modules that make development easier. They are grouped together and allow you to create applications or websites from an existing source, instead of from scratch.

With the uses of the Django framework, we can develop and deploy web applications within hours as it takes care of much of the hassle of web development. Django is amazingly fast, fully loaded such as it takes care of user authentication, content administration, security as Django takes it very seriously and helps to avoid SQL injection, cross-site scripting etc. and scalable as applications can be scalable to meet high demands and used to build any type of applications that is why we call it as a versatile framework. We can build different applications from content management to social networking websites using the Django framework. It offers lots of resources and good documentation, which helps new learners to learn and experience

4.5.1) Django Rest framework

Django REST framework (DRF) is a powerful and flexible toolkit for building Web APIs. Its main benefit is that it makes serialization much easier. Django REST framework is based on Django's class-based views, so it is an excellent option if you're familiar with Django. It adopts implementations like class-based views, forms, model validator, QuerySet, and more.

and we used Postman to test our Apis

4.6) Postman

Postman is a plugin in Google Chrome, and it can be used for testing API services. It is a powerful HTTP client to check web services. For manual or exploratory testing, Postman is an excellent choice for testing API.

Figure 52.

Features:

- With Postman, nearly all modern web API data can be extracted
- Helps you to write Boolean tests within Postman Interface
- You can create a collection of REST calls and save each call as part of a collection for execution in future
- For transmitting and receiving REST information, Postman is more reliable.

The Apis of Smart-Cart project and its functions

Products list function

```python
@csrf_exempt
@api_view(['GET'])
def products_list(request):
    if request.method == 'GET':
        product = Product.objects.all()
        serializer = ProductSerializer(product, many=True)
        return JsonResponse(serializer.data, safe=False)
```

Figure 53.

Products list Api

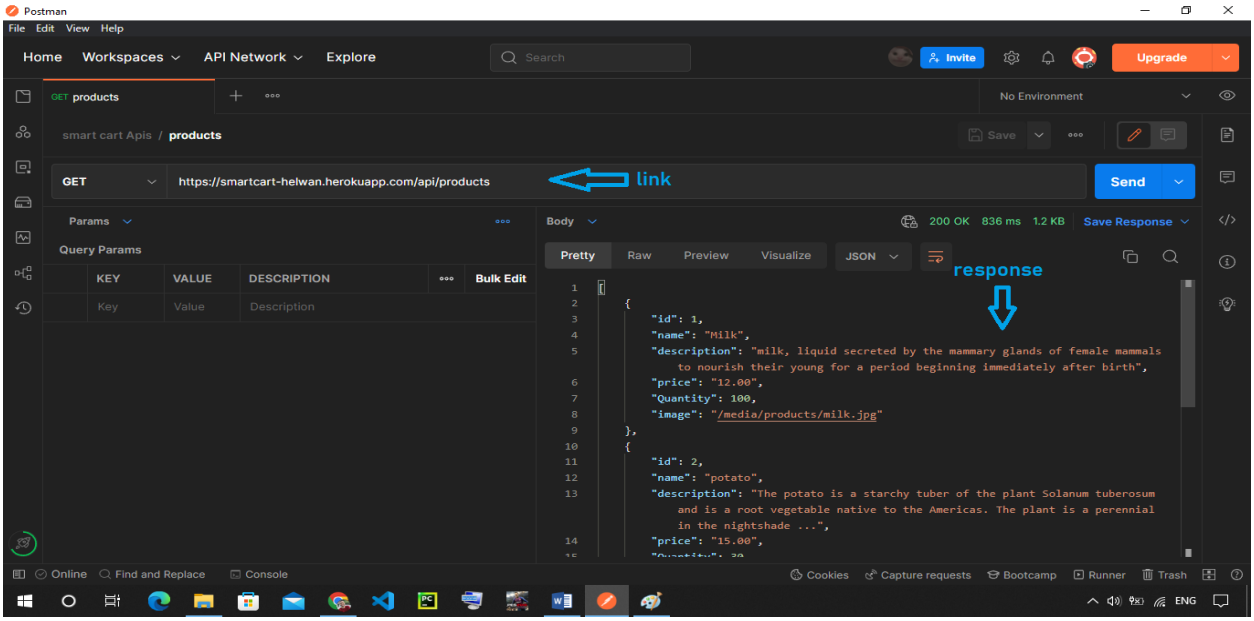


Figure 54.

Registration function

```python
@csrf_exempt
@api_view(['POST'])
def register_user(request):
    if request.method == 'POST':
        data = JSONParser().parse(request)
        serializer = UserProfileSerializer(data=data)
        if serializer.is_valid():
            serializer.save()
            return JsonResponse({
            'message': 'Succseccfully Registerd'}, status=201, safe=False)
        return JsonResponse(serializer.errors, status=400)
```

Figure 55.

Registration Api

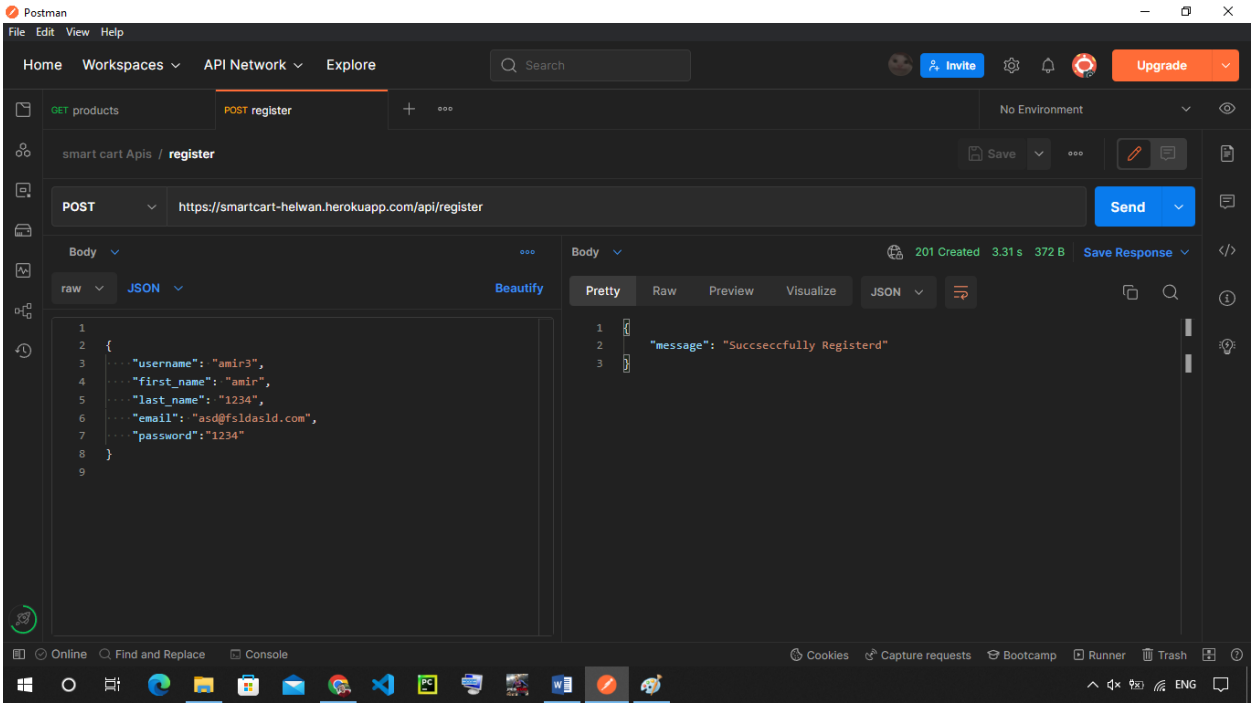


Figure 57.

Login function

```python
class CustomAuthToken(ObtainAuthToken):

    def post(self, request, *args, **kwargs):
        data = JSONParser().parse(request)
        serializer = self.serializer_class(data=data,
                                           context={'request': request})
        serializer.is_valid(raise_exception=True)
        user = serializer.validated_data['user']
        token, created = Token.objects.get_or_create(user=user)
        return Response({
            'token': token.key,
            'email': user.email
        })
```

Figure 58.

Login Api

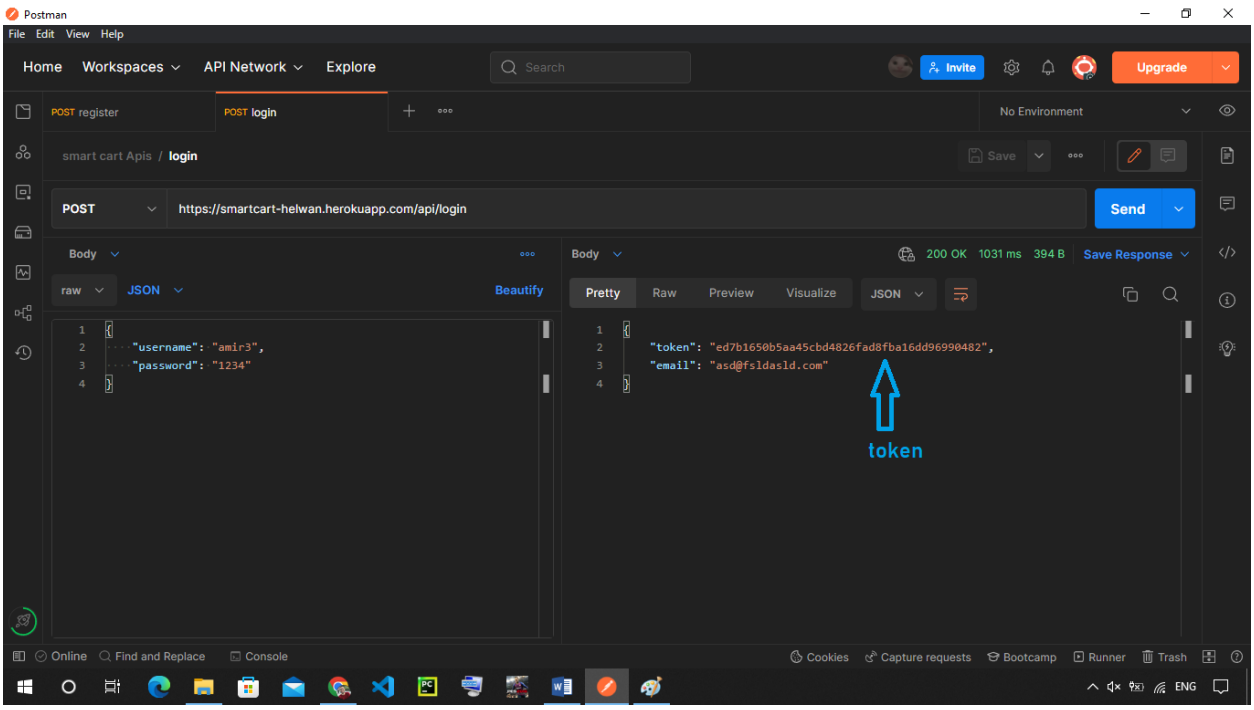


Figure 59.

User profile function

```python
@csrf_exempt
@api_view(['GET'])
def orders_list(request):
    if request.method == 'GET':
        orders = Order.objects.all()
        serializer = OrderSerializer(orders, many=True)
        return JsonResponse(serializer.data, safe=False)
```

Figure 60.

User profile API

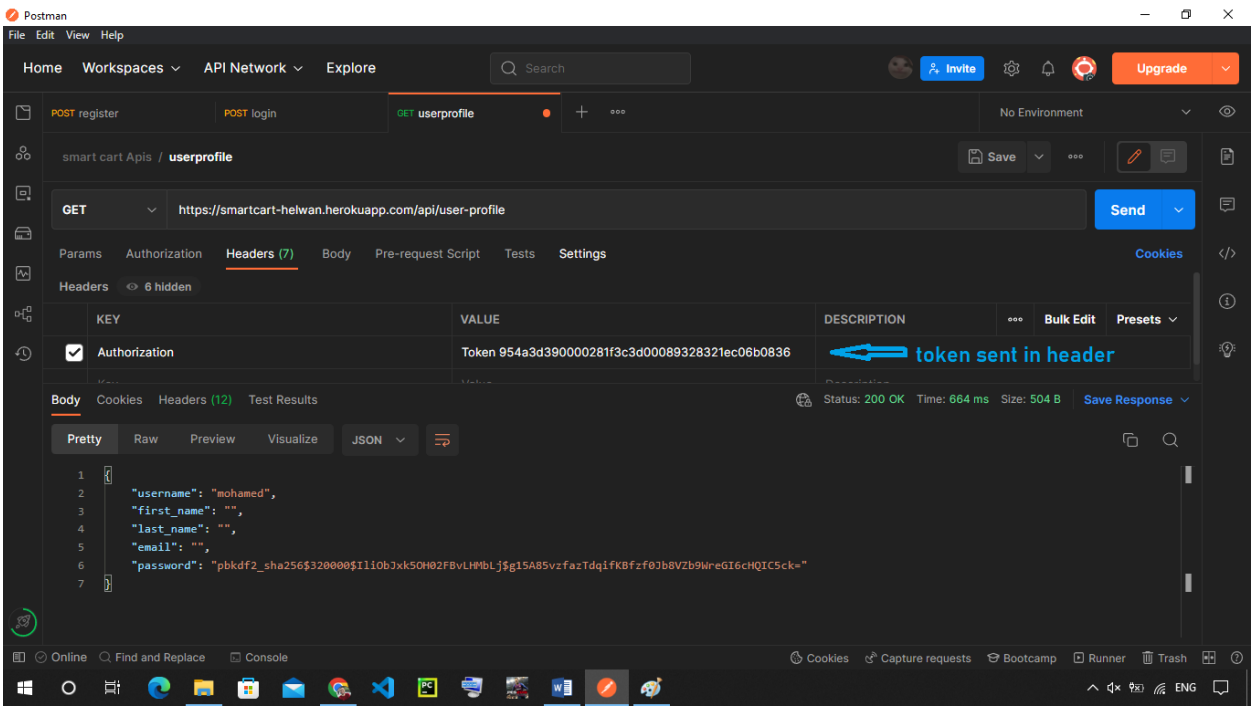


Figure 61.

Choose cart and create order function

```python
@csrf_exempt
@permission_classes([IsAuthenticated])
@api_view(['POST'])
def create_order(request):
    if request.method == 'POST':
        try :
            cart = Cart.objects.get(id = request.data.get("cart"))
        except Cart.DoesNotExist:
            return Response ( {"Error":" Cart dosn't exist "})
        if cart.occupied :
           return JsonResponse({"message":"cart not availabe"})
        else:
            order = Order.objects.create(customer=request.user,cart=cart)
            serializer = OrderSerializer(order)
            return JsonResponse({"order":serializer.data})
```

Figure 62.

Choose cart and create order API

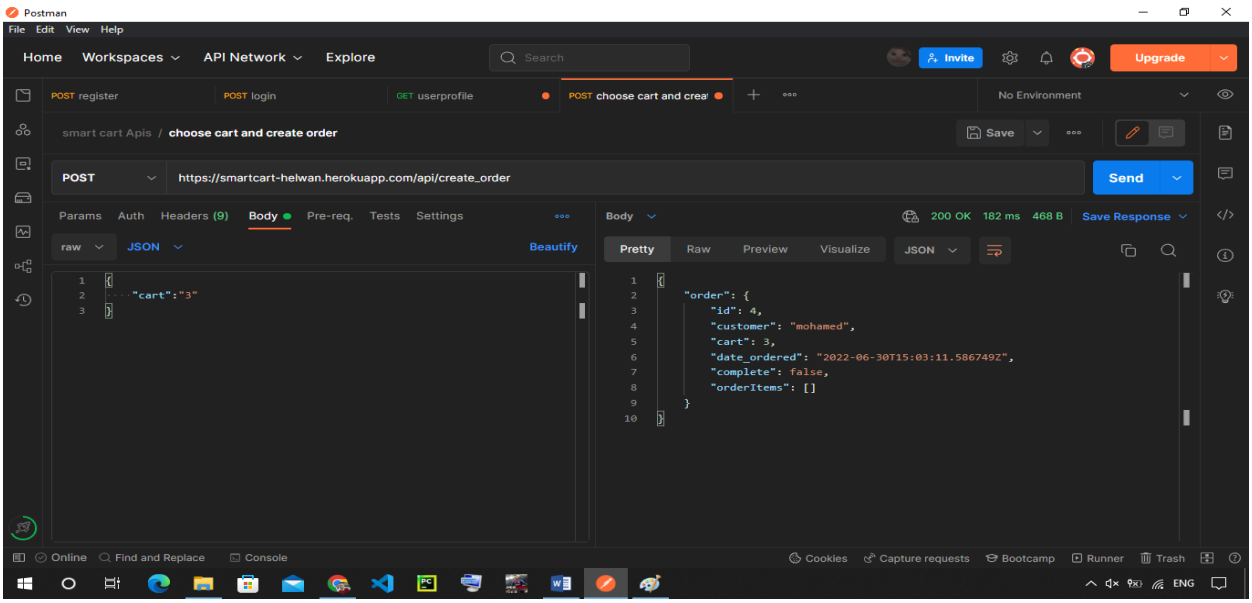


Figure 63.

Add item to order function

```python
@csrf_exempt
@permission_classes([IsAuthenticated])
@api_view(['POST'])
def adding_orderItem(request):
    if request.method == 'POST':
        barcode=request.data.get("barcode")
        try:
            product = Product.objects.get(barcode=barcode)
        except Product.DoesNotExist:
            return Response(status=status.HTTP_404_NOT_FOUND)
        order=request.user.order_set.last()
        print("0000000000000000000000000000000000000000")
        checker=0
        for i in order.orderItems.all():
            if product == i.product:
                checker=1
                break

        if checker == 1 :
            f =order.orderItems.get(product=product)
            setattr(f,"quantity",f.quantity+1)
            f.save()
        else:
            order.orderItems.create(product=product,quantity=1)
        order_serializer = OrderSerializer(order)
        return JsonResponse(order_serializer.data)
```

Figure 64.

Add item to order

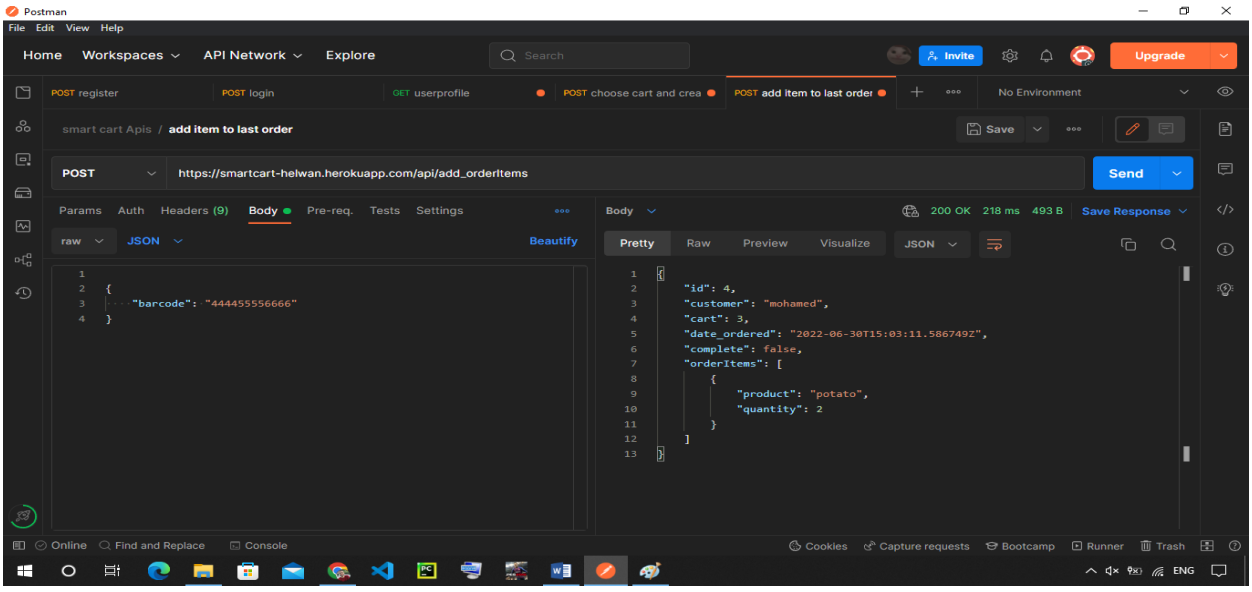


Figure 65.

All orders for specific user function

```python
@csrf_exempt
@permission_classes([IsAuthenticated])
@api_view(['GET'])
def user_orders(request,):
    if request.method == 'GET':
        id=request.user.id
        user_orders = Order.objects.filter(customer=id)
        serializer = OrderSerializer(user_orders, many=True)
        return JsonResponse(serializer.data, safe=False)
```

Figure 66.

All orders for specific user API

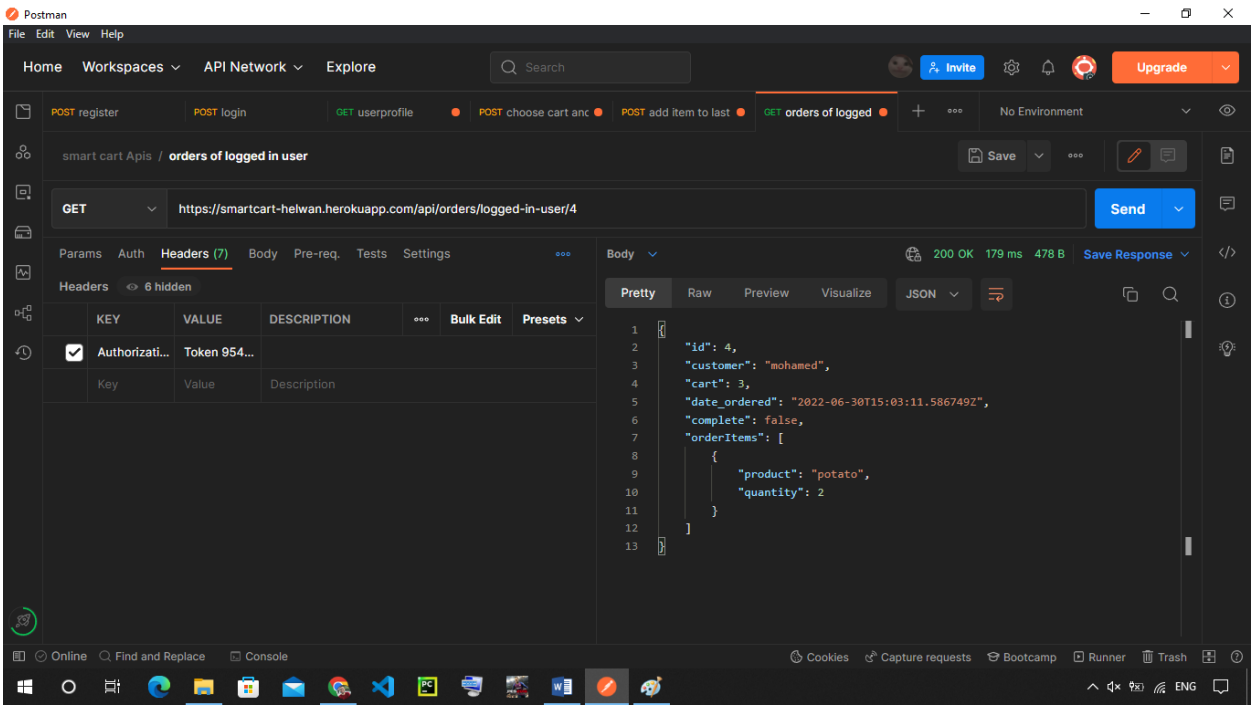


Figure 67.

Get Details of one order function

```python
@csrf_exempt
@permission_classes([IsAuthenticated])
@api_view(['GET'])
def get_order_details(request,id):
    try:
        order = Order.objects.get(id=id)
    except Order.DoesNotExist:
        return Response(status=status.HTTP_404_NOT_FOUND)
    serializer = OrderSerializer(order)
    return Response(serializer.data)
```

Figure 68.

Get Details of one order API

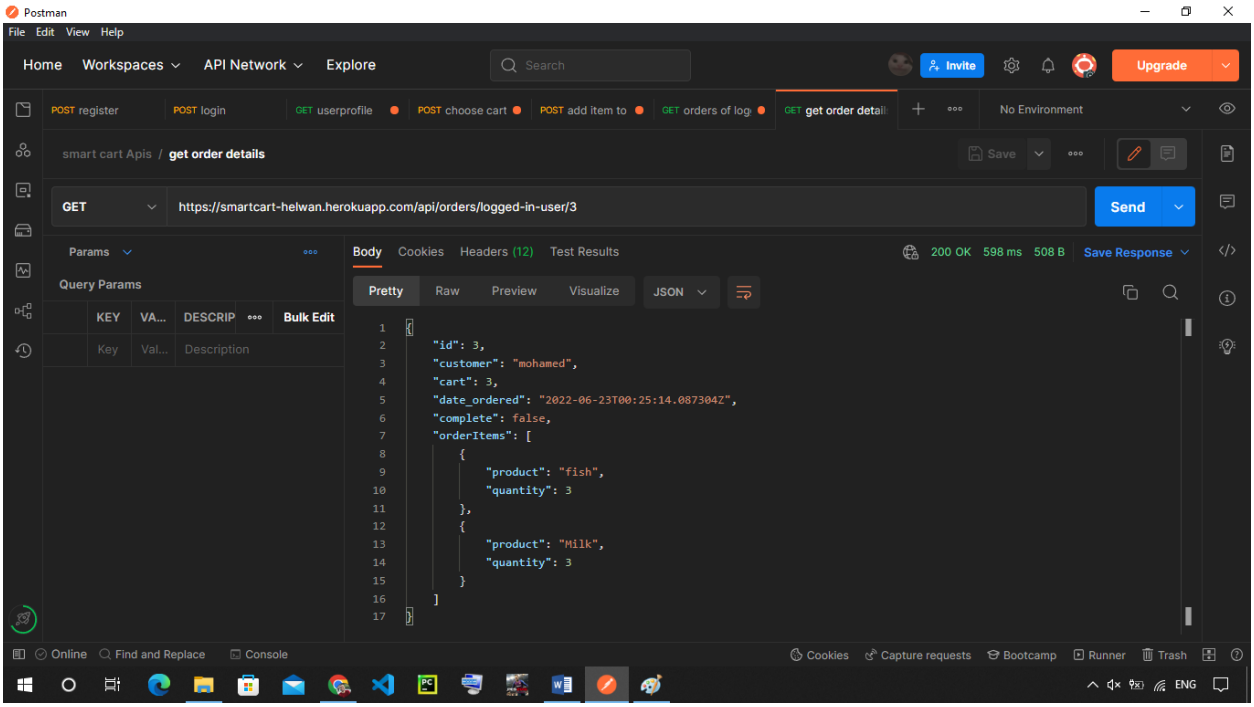


Figure 69.

Chapter 5. Recommendation Engine

5.1. Introduction

The approach users engage with webpages has been revolutionized thanks to recommendation engines, which are now becoming more and more significant. By considering the consumers' increased selections and unbiased actions, recommender systems improve user access and seize command of making smart, customized product recommendations to consumers. Presently, recommendation systems are essential whether they are used on e-commerce platforms or as online advertisements. Each and every business is attempting to harness the potential of recommendation systems. Such technologies have a wide range of applications in the fields of business, research, and education, among many others. In order to intelligently match customers with products that suit their interests, nearly all well-known websites incorporate recommendation systems. For one, online news outlets review readers' reading preferences to suggest articles that are relevant to their likes while online social networking websites suggest new friend connections to users relying on their present acquaintances. Online sellers frequently use client purchasing histories to influence product recommendations and boost their own revenues. These instances highlight the tremendously important part recommendation systems play in guiding our everyday decisions.

Because information technology is developing quickly, the pace at which digitized data is growing is overwhelmingly quick. The recommendation engines have been quite successful in addressing the issue of large volumes of data. Presently, there is a huge number of items advertised on e-commerce platforms like Amazon and Flipkart. When faced with a vast variety of options of potential products, recommendation systems assist users. Over 12 million products are marketed to be sold on Amazon right now. As a result, buyers find it difficult to take a decision choosing their most preferred product. The recommendation system analyzes the large amount of available data by prioritizing the most significant information according to every user's history, taking into account the user's own taste and areas of interest. Depending on the user's profile as well as his existing records of past transactions, a recommendation system is able to determine if a certain user will favour a product or not. Both service providers and consumers benefit from recommendation systems. This type of solution has also enhanced the

accuracy of the decision-making procedure. The following are the primary outcomes of recommendation systems:

1. Helps people locate products that interest them.
2. Assist vendors in connecting their products to the most appropriate buyer.
3. Identify the goods the consumer will find particularly valuable.
4. Tailored insights on each customer, which enhances the overall personalized consumer experience.
5. User interaction is increased.

On Amazon, there were two books advertised. "Touching the Void," the first book, was penned by a mountain climber in 1988. Amazon had just begun selling books online at that period. Nevertheless, that book did not receive any recognition from the public, and accordingly, few copies were sold. A couple of years later, in 1996, another publication titled "Into thin Air" appeared. It had a comparable plot, became well-known, and had massive purchases. At that point, Amazon began deploying its recommendation engine. As a result, Amazon began suggesting "Touching the void" to anybody who had purchased the book "Into thin Air," which led to the old book's purchases exceeding those of the newer one. Due to a simple insight, a valuable recommendation has been proposed, and this wizardry occurred. Because it is a tried-and-true method for increasing and expanding purchases as never before, this technological innovation has proven to be a blessing for e-commerce retailing companies. (Son, J. et al., 2018)

5.2. Literature Review

In the realm of product suggestion, a substantial volume of studies have been conducted. Collaborative filtering has been applied in a variety of applications in the past that are exclusively dependent on user-user and product-product interactions. Films, publications, and product recommendations are a few of the domains where collaborative filtering is used. Due to the restrictions on collaborative filtering approaches, web data mining strategies centered on web-use mining have been explored.

Web Mining can be described as the practice of implementing data mining strategies to identify various trends in digital content. Association rules, webpage clusters, and other various pattern detection techniques are among the trends discovered. On the basis of web data mining, a recommendation engine was developed, as was thoroughly explained in a research paper by (Shani et al., 2015). The strategy for such recommendation engine involved making product

recommendations based on online usage statistics while also incorporating information about product purchases into account. The approaches' purpose was to suggest products that consumers would find interesting. For this purpose, the developers had the recommendation system suggest a selection of the top-N items for various customers at a specific moment. For the online retailing's planned recommendation system to be validated, various experiments involving web activity were conducted. The proper level of product and consumer are taken, it was discovered during assessment, to improve the accuracy of suggestions (Shani et al., 2015).

Another approach of merchandise recommendation was developed, combining methods for collective decision-making and the previously mentioned recommendation system which was based on web data mining. To quantify the comparative values of the frequency and monetary (RFM) variables in their study of customer loyalty, they used the analytical hierarchy process (AHP). Next, based on the scaled RFM value, they created a grouping of consumers using clustering algorithms. Finally, they used association rule mining to propose products to each consumer segment. The result reached was that, while promoting a large variety of products does not improve the effectiveness of the recommendation system for less loyal clients, it does for those who are loyal to the business.

In a paper written by (Zhang et al., 2012), the development of an engine for individually tailored product recommendations was explored. The recommendation algorithm tracked each user's purchase patterns by using data mining approaches. Each consumer pick and merchandise association were obtained automatically from user's clock feeds. Secondly, an algorithm combined consumer preferences and commercial merchandise's associations to rate each particular product. Therefore, a mechanism is employed to create product catalogs for each consumer so they may give recommendations based on. They managed to demonstrate by thorough testing that their technology saves clients time when they purchase online and provides helpful recommendations.

5.3. Types of Recommendation Systems

Nowadays, each e-commerce platform, including Flipkart and Amazon, incorporates a recommendation system. Additionally, firms like Netflix and YouTube employ this predictive analytical solution to increase customer engagement. Recommendation engines come in numerous variations and are implemented in accordance with the architectural structure of the

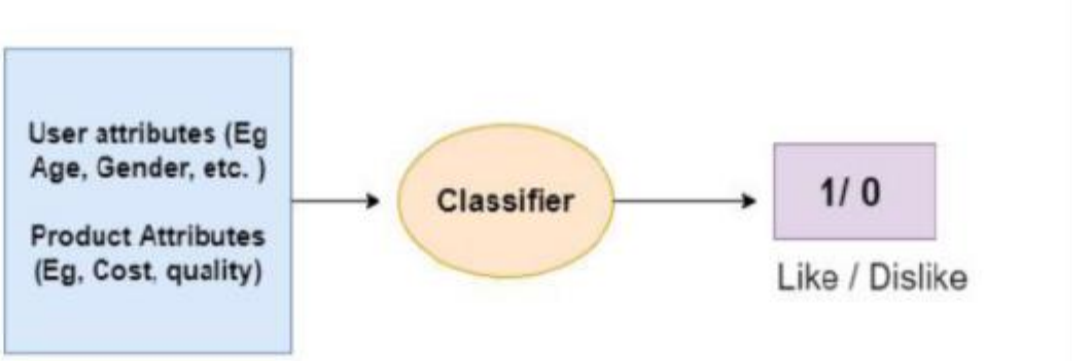
many venues that employ them. Following is an overview of several recommendation system classifications:

5.3.1. Popularity Based Recommendation System

This kind of recommendation system is one which is set up to provide users recommendations for the things that are currently highest in demand. It evaluates the goods that are popular at the moment. For instance, it goes after the chance that the customer who had just subscribed will acquire the same thing if it is an item which is typically bought by all novel customers. Broadly speaking, the most well-liked items may be located using a variety of filters including customer reviews, various places, etc. Since such types of recommendation systems are abe to suggest the highest trending goods to each registered consumer depending on various filters, popularity based recommendation systems do not experience the cold start challenge. The customer's prior information is not required. Such systems, nevertheless, still suffer from a number of issues, one being the lack of personalization. Each new customer who came to the platform for the first time will receive the exact same recommendations. The most well-known examples of the popularity-based recommendation systems are YouTube's viral videos and Google News, where news is sorted by major headlines (Schreiber et al., 2008).

5.3.2. Classification Model

This kind of recommendation system is a form of predictive model which determines whether a given customer would buy a certain item or not by taking into consideration both consumer and product characteristics. In other words, the output of the classification model will always be 1, or 0; where 1 simply signifies that the algorithm is predicting a specific user will buy a certain product and 0 means he won't. Since there must be a substantial volume of data about both items and various consumers, classification models are thought to be particularly complicated tasks. Additionally, it gets even harder to make a generalization if the past challenge was overcame. The agility of the classification algorithm is also of concern.



5.3.3. Content Based Recommendation System

It is a kind of recommendation system that functions according to the recommend-similar-content concept. When a customer searches for an item, the algorithm automatically detects items based on various criteria and suggests them to the customer.

Such recommenders are regarded as somewhat personalized solutions since they only suggest items with comparable content, which therefore implies that customers can only be shown things that attract them. Take the One Plus 7 smartphone for instance. Flipkart website lists two further variants, the One Plus 7T and One Plus 7T pro. Flipkart begins promoting the 7T and 7Pro versions of the smartphones only if the customer is searching for One plus 7 variations. This is obtained by calculating how comparable the items are based on several criteria. Let's use the cameras and RAMs as possible parameters to calculate the resemblance between the two variants of smartphones.

ONE PLUS 7	ONE PLUS 7T	ONE PLUS 7T PRO
6GB ram	6GB ram	8GB ram
32MP camera	32MP camera	32MP camera
64GB storage	64GB storage	64GB storage

Utilizing Euclidean distance metric will allow us to compare all one plus seven smartphone variations. Two objects are said to be alike if the Euclidean distance equals 0, otherwise they are not. If the distance between the One Plus 7 and One Plus 7T is determined based on RAM and Camera, it leads to the conclusion that they are comparable smartphones. However, if you

analyze the resemblance across One Plus 7T and One Plus 7T Pro using both RAMs and storage, the result won't be zero; nevertheless, if the proximity is measured using just storage, the result will be zero. These fundamental metrics are performed to evaluate the resemblance of the goods, and the consumer is therefore encouraged to buy the product which best fits their needs. The proximity measurement may be computed about any item aspect. Euclidean distances are applied to assess similarities between numerical values, cosine similarity is employed to compare similarities across textual documents, and Jaccard similarity is performed to compare similarities amongst categorical attributes. The following discourse covers an overview on various proximity measures.

i. Euclidean Distance:

Calculates the distance between two points x and y. As the distance between such points approach zero, it means that the two objects being analyzed are comparable. As the distance increases, it suggests that the two objects are not very alike.

$$d(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

ii. Cosine Distance:

In order to detect similarity, the cosine of the angle formed by the two product vectors of A and B is determined. If such angle is small, the cosine will be considerably high (close to 1) and this will only take place if the vectors are close to each other together. There is no such thing as a perfectly matched product. An individual with a 40K budget is searching for a computer with a 1 TB HDD, 4GB RAM, and an i5 CPU. There'll never arise a situation in which he is referred to a laptop with the precise same characteristics. The suggested computer will resemble the one the user is searching for in certain ways, such as having a 40K price tag. A recommendation algorithm cannot suggest identical goods.

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|}$$

5.3.4. Collaborative Filtering Recommendation System

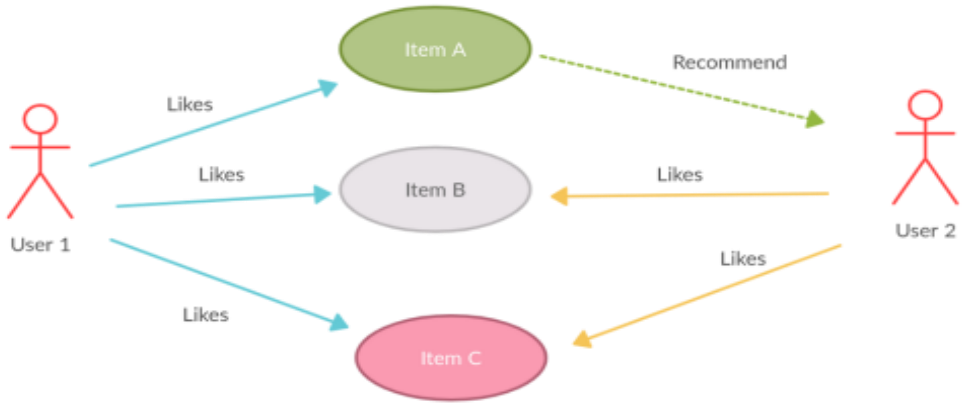
It is a highly sophisticated recommendation system. It operates under the basis of comparable consumers or comparable products. Netflix, Amazon, YouTube, and other platforms frequently

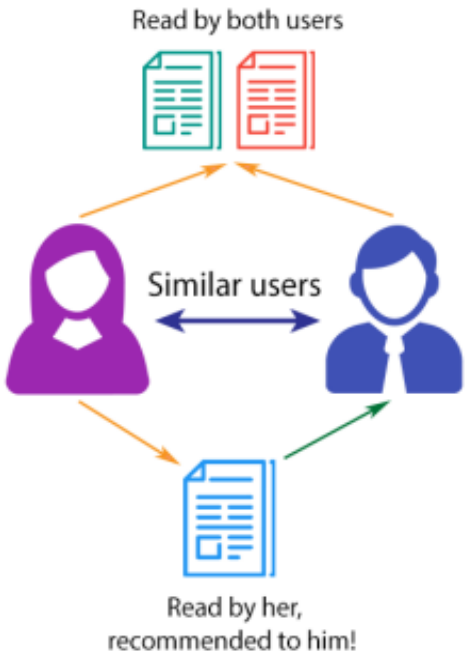
employ this model of recommendation systems. Given that we are capable of providing adequate volume of data about products and consumers, the system will operate with sufficient effectiveness (Singhal et al., 2020).

Identifying matching customers and products may be done using a variety of techniques. In collaborative filtering, there are primarily two methods that are applied:

1. User Based Nearest Neighbor Collaborative Filtering

The customers who share similar preferences in goods purchasing are identified to be similar customers. Relying on buying behavior, customer similarity is calculated. Since they have made similar purchases of items, Users 1 and 2 are comparable users.





2. Item Based Nearest Neighbor Collaborative Filtering

It offers recommendations for products similar to those the customer has previously purchased before. For the purpose of forecasting a personalized recommendation, similarity across various products is calculated solely on the products rather than the consumers.

5.3.5. Singular Value Decomposition and Matrix Factorization

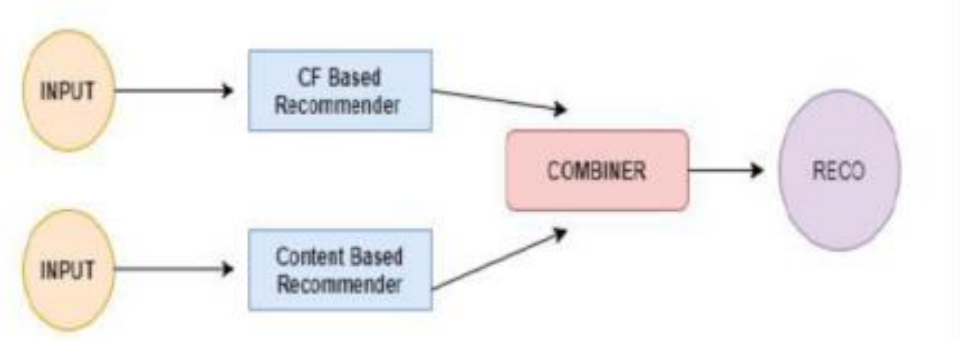
This is an approach that is based on matrix factorization for the purposes of decreasing the dimensionality of the database from N to K, where K is less than N, hence decreasing the number of its attributes. Just the matrix factorization generating recommendations is performed, and it is applied on the user-item rating matrix. Items and users are represented as vectors whereas the dot product of them corresponds to the predicted scoring.

Avoiding overfitting is a crucial component of every machine learning model, and generalization is employed to achieve this. It eliminates the possibility of an overfitted model. The customer and product vectors are multiplied by the penalty factor, which is then square sum of the magnitudes. The SVD method reduces the discrepancy across the real and anticipated values.

5.3.6. Hybrid Recommendation Systems

Hybrid recommendation systems are ones that incorporates characteristics from several recommendation engines. According to studies, it can be more effective in several instances to combine collaborative filtering and content-based filtering. Such systems could be created using a variety of techniques, including developing content-based and collaborative forecasts separately before integrating them, boosting each technique's capability.

When compared to simple content-based or collaborative-based methods, studies demonstrate that hybrid models outperform them. In recommendation systems, hybrid techniques also outperform issues such as the cold start and sparsity challenges. Among the main representations of a hybrid model is Netflix. Their method makes recommendations by analyzing what viewers have watched, accounting for comparable users, or User Based Collaborative Filtering, and also recommending movies whose content are alike to movies that has received the highest rating by the viewer.



5.4. Challenges of Recommendation Systems

5.4.1. Cold Start

This issue arises whenever a new customer logs in or a new product is added to the database. In such situations, the system becomes unable to provide a rating for the new product, and is also unable to identify users with similar preferences for the new user because of its inability to identify the new user's likings to begin with (Trappey et al., 2013). This kind of issue could be addressed by asking the user to rate random product genres or by inquiring about the consumer's preferences or suggesting the buyer things based on socio-demographic information such as age and gender.

5.4.2. Synonymy

This issue arises when the same product is presented under many identifiers or groupings which have the same meaning. [10] In some situations, the recommendation system is unable to identify if it's the same product or a distinct one. For instance, a User Based Collaborative Filtering recommendation system will identify a "comedy movie" to be different from "comedy film." Synonym use decreases the recommendation system's effectiveness. Since item content is not taken into account, the recommendation system doesn't really take hidden connections across products into account. That's the major cause of why a new item with similar naming or under similar category would not be recommended even if its ratings were there. Latent Semantic Indexing and Singular Value Decomposition can both be utilised to solve the issue[19].

5.4.3. Shilling Attacks

Imagine the scenario when a dishonest individual enters the website and begins to give false ratings in order to boost the popularity of a product or vice versa. This kind of behaviour is known as Shilling Attacks [11]. These attacks have the capability to impair the recommender's integrity and effectiveness. This is particularly prevalent in recommender systems built on Collaborative Filtering. Several attack paradigms, such as bandwagon, random, average, and reverse bandwagon attacks, may be identified using a variety of techniques, such as prediction shift and hit ratio.

5.4.4. Privacy

Giving a recommendation system your personal data could undoubtedly result in useful suggestions, but it may compromise your information security. While privacy concerns crippling recommendation systems, users are reluctant to provide their information.

Therefore, a recommendation system must foster consumer trust. In Collaborative Filtering based solutions, customer information, including rating information, is stored in a centralized repository that is vulnerable to hacking, increasing the risk of data theft. In order to make individualized suggestions without involving other parties, encryption techniques should be put to use.

5.4.5. Gray Sheep

In simple Collaborative Filtering recommendation systems, grey sheep happens when a customer's preferences don't align with any category of users and as a result, they are incapable of making use of recommendations [21]. In order to improve efficiency and reduce recommendation inaccuracy, clustering techniques like K-means could be employed to identify such categories of users[22].

5.4.6. Evaluation and Benchmark Datasets

The recommendation system's competence is revealed by the assessment measures. For recommendation systems, choosing assessment methods is regarded as a challenge. Conventional recommendation systems[11], [16] employ evaluation criteria like MAE, precision, and F measures for assessment. The lack of a benchmark dataset which could be employed to assess the recommendation engine is another difficulty.

These kinds of datasets promote research and recommendation system advancement.

5.5) Dataset and Proposed Model

5.5.1. Introduction

The dataset for the Amazon Customer Reviews could be obtained as a TSV file that will be available in the index. The Amazon Customer Reviews Dataset is a substantial collection of TSV files that contain more than 100 million reviews from customers who have purchased items from the Amazon platform. For Amazon Customer Reviews are a great data source that could be utilized for many Machine Learning (ML) and Natural Language Processing (NLP) applications. Reviews from clients in five countries encompass the years between 1995 and 2015. Although each TSV file has millions of reviews, the majority of them are for products that fall under just a single genre (beverages, clothes, shoes, gardening equipment, etc.). The collection of data provided includes 7 million or more ratings on a 5-star scale for many products provided on the Amazon website from at least 4 million unique consumers. Between August 8th, 1995 to August

31st, 2015, there were more than 86 thousand items in more than 16 thousand product parents in 39 distinct categories. Mobile applications, electronic books, computer games, home decor, cosmetic products are just a few of the product genres.

5.5.2. Dataset Columns

Marketplace: The two-letter identifier of the country in which the review was published.

Customer ID: A randomized code that which could be employed in order to group reviews made by a unique user.

Review ID: A specific Identifier assigned to each review.

Product ID: A distinctive identifier for each product.

Product parent: This is a randomized identifier which may be leveraged to group reviews for each item.

Product title: The item's name.

Product category: A generic merchandise category which could be utilized to cluster reviews under the same merchandise category.

Star rating: The review's rating, from 1 to 5 scale where 5 is the highest and 1 is the lowest.

Helpful votes: Count of positive ratings that a review has earned.

Downvotes: A review's total count of negative ratings.

Total votes: Cumulative votes obtained by the review, expressed as an integer.

Verified purchase: This column correlates to whether a review relates to a legitimate buy that had occurred.

Review headline: Refers to the review's heading.

Review body: The content of the review.

Review date: It indicates the time the review was published.

Purchased count: The number of buys the item has witnessed.

5.5.3. Data Wrangling

Data wrangling, also known as data cleaning, data remediation, or data munging, pertains to a number of procedures which are performed to convert large amounts of data into forms that are easier to work with. According to the data you're using and the objective being pursued, the actual approaches employed vary from one project to another.

Data wrangling includes, for instance:

1. Constructing a unified collection of data by combining several data sources.
2. Locating missing data and replacing them with non-null values such as zero, the mean value or a value identical to that of the row above or beneath the missing value, or even removing the records that have missing values altogether.
3. Finding significant anomalies in data and either using analysis methods for justifying the anomalies, or eliminating such data points in order to enhance the accuracy of the model.
4. The creation of new features.
5. Reducing the number of features in the dataset known as dimensionality reduction.

For the purposes of the creation of a reliable recommendation system, and since the majority of the merchandise products in the dataset have extremely few transactions. We desired to choose goods with a sufficient number of sales while keeping the dataset as complete as feasible. We were capable of keeping 85 percent of the dataset with goods which had at least 100 sales

And although around 75,000 of distinct goods were wasted, the majority of these were found to be individual buys of similar items with multiple product IDs, which is the synonymy challenge that was previously discussed. If this challenge hadn't been handled by removing the veracious data rows, it could introduce a lot of unnecessary noise throughout our analysis and modelling stages.

Data records which contained missing values were imputed to enhance the accuracy of recommendations. In addition, all duplicates were also imputed and the dataset was arranged according to user ID for simpler searching and retrieval.

The final dataset contained 5,833,870 reviews made by 3,653,661 unique users for 11,502 distinctive merchandise goods under 29 categories.

5.5.4. Recommendation System's Proposed Model

Amazon is a Seattle-based global technology corporation with an emphasis on digital commerce, cloud services, digital streaming and artificial intelligence. The digital commerce portal

"Amazon.com" is among its most well-known offerings. Amazon's merchandise recommendation system is one approach Amazon takes in order to retain its customers by giving them a customized shopping experience. The item recommender system makes suggestions for products relying on a customer's prior buys and the expenditures of other customers who made analogous buys. Considering the aforementioned behaviors, the system suggests products which the customer might want. Although content-based filtering permits customers to examine products in the same territory of the products they bought and loved, collaborative filtering provides customers with the ability to examine products from different categories, that they didn't know they even wanted to begin with. Collaborative filtering-based recommender systems enable customers to pinpoint relevant merchandise items that happen to be comparable to these previously purchased by the user, yet gives the shopper the flexibility to leap to new product groups depending on resemblance with other customers and goods, in contrast to content-based filtering, that really solely suggests comparable goods. For all the aforementioned reasons, Collaborative Filtering was chosen as the approach for the implementation of the recommendation system.

Three alternative applications for collaborative-based filtering are examined:

1. According to how well-liked a certain item is by a particular customer; the first technique proposes items to such customer. In essence, it locates other customers who have given comparable ratings to initial customer and then presents the best picks from those customers.
2. The second technique makes predictions about the goods a customer would presumably prefer relying on their prior rating history in addition to previous ratings of other users. Unlike the first technique, this approach does not propose recommendations solely based on the review of a single item.

3. By identifying the closest neighbors to a given item, the third technique recommends products that resemble that item. For customers that prefer to buy things in bundles, this strategy works well.

The above techniques' performance will be assessed by:

1. Examining the items that the recommendation engines generate.
2. Recognizing the impacts of matrix factorization on collaborative filtering-based recommendation systems.

Assessing recall and precision of each method.

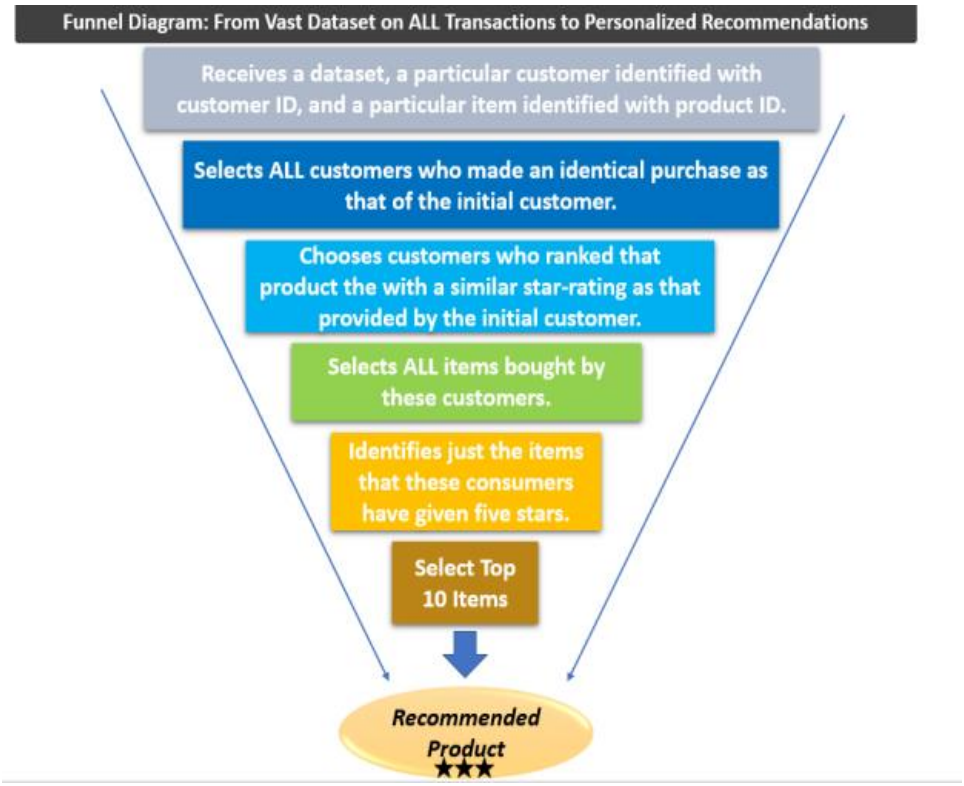
5.5.4.1. Simple Collaborative Based Filtering

5.5.4.1.1. Base Model

This straightforward collaborative-based filtering recommender engine serves the objective of illuminating the underlying functioning of the core of a recommendation system, which will be rendered our base model. The goal of implementation is to find a way to suggest products to a consumer that they might not have even previously considered buying, providing a sophisticated method of exploring goods without being limited to a group of items that are comparable.

All goods which are suggested to a consumer by this recommender system are determined by the rating they assigned a particular item. It pinpoints users who have given the product a similar rating and displays the most prominent items those consumers have bought, basically promoting goods loved by other users.

The next figure shows a funnel diagram containing the steps the Collaborative Filtering based Recommendation System takes in order to provide personalized recommendations for each unique user.



Products proposed to customers will enable them to view items that were purchased and loved by other customers having the same preferences as them. Suggestions provided are both customer

and merchandise item centric. The goods suggested aren't always comparable. The technique employed was designed to not only provide recommendations, but also to explain the rationale behind reaching that recommendation.

The criteria for the datasets which are passed to the recommendation system vary, and according to such variations, significant impact on time taken for generate recommendations and the effectiveness of such recommendations were monitored. Recommendations were found to be generated more quickly as the criterions went up. This can be explained by taking a look at the funnel diagram shown above, whereas if the filtering criteria encapsulates a vast number of users and products, this will result in a bigger number of comparable users, and accordingly, will take more time to process such users, their past transactions as well as the star ratings for each of the aforementioned transactions.

Criterions:

1. "ALL" criterion includes all customers starting with one transaction and going upward. This filter allows all customers registered on the website who had made AT LEAST one buy.
2. Customers with two transactions or more pass the "Multiple" criterion, encapsulating customers who made more than one buy.
3. The "Several" criterion enclose customers who made three transactions and above, basically all customers who made more than two buys.
4. Everyone who has made four buys or more is a “Many” customer.

The number of records containing information in each dataset, distinct consumers, and distinct items are what cause the variations in criteria.

5.5.4.1.1. Base Model Evaluation

5.5.4.1.1.1. Speed

This assessment measures how long the recommender engine requires to traverse every criterion. By raising the criterion values, the generated recommendations are produced more quickly due to fewer distinctive consumers and overall merchandise items to compare.

Variations in suggested products from one criterion level to another will be noticed as a result.

1. Applying the “All” Filter

```
%%time

# Criterion: 1 or More Transactions Per Customer
collab(all, 10285, 'B00AREIAI8')
```

```
----------------------------------------
19124 other customers purchased this product.
Similar customers purchased 29287 other products.

We recommend these products from those similar customers:
0   Minecraft
1   Flappy Wings (not Flappy Bird)
2   Candy Crush Saga
3   Crossy Road
4   My Horse
5   Subway Surfers
6   Minion Rush: Despicable Me Official Game
7   Guess The Emoji
8   Temple Run 2
9   Temple Run


CPU times: user 1.94 s, sys: 161 ms, total: 2.1 s
Wall time: 1.58 s
```

2. Applying the “Multiple” Filter

```
%%time

# Criterion: 2 or More Transactions Per Customer
collab(multiple, 10285, 'B00AREIAI8')
```

```
----------------------------------------
6734 other customers purchased this product.
Similar customers purchased 16897 other products.

We recommend these products from those similar customers:
0   Minecraft
1   Flappy Wings (not Flappy Bird)
2   My Horse
3   Subway Surfers
4   Crossy Road
5   Minion Rush: Despicable Me Official Game
6   Candy Crush Saga
7   Guess The Emoji
8   Temple Run 2
9   Temple Run


CPU times: user 1.12 s, sys: 88.5 ms, total: 1.21 s
Wall time: 913 ms
```

3. Applying the “Several” Filter

```
%%time

# Criterion: 3 or More Transactions Per Customer
collab(several, 10285, 'B00AREIAI8')
```

```
----------------------------------------
3015 other customers purchased this product.
Similar customers purchased 10415 other products.

We recommend these products from those similar customers:
0   My Horse
1   Flappy Wings (not Flappy Bird)
2   Crossy Road
3   Minion Rush: Despicable Me Official Game
4   Minecraft
5   Subway Surfers
6   Candy Crush Saga
7   Temple Run 2
8   MY LITTLE PONY - Friendship is Magic
9   Temple Run


CPU times: user 725 ms, sys: 58.8 ms, total: 784 ms
Wall time: 558 ms
```

4. Applying the “Many” Filter

```
%%time

# Criterion: 4 or More Transactions Per Customer
collab(many, 10285, 'B00AREIAI8')
```

[10]

```
----------------------------------------
1493 other customers purchased this product.
Similar customers purchased 6646 other products.

We recommend these products from those similar customers:
0   My Horse
1   Flappy Wings (not Flappy Bird)
2   Minion Rush: Despicable Me Official Game
3   Minecraft
4   Crossy Road
5   Candy Crush Saga
6   Temple Run 2
7   Temple Run
8   Subway Surfers
9   MY LITTLE PONY - Friendship is Magic
```

5.5.4.1.2.2 Evaluating Recommendations Made on Products with Vast Number of Sales

The previous assessment used merchandise items with an average number of sales. This assessment examines how well the recommender engine operates given an item that has had a substantial number of sales.

```
# Randomly Filtering a Customer ID Which Has Purchased a Certain Product Having a Large Amount of Past Purchases
All[(All.purchased_counts > 25000) & (All.purchased_counts < 45000)][
    ['customer_id', 'product_id', 'product_title', 'product_category',
     'star_rating', 'review_headline', 'purchased_counts']
].sort_values(['purchased_counts', 'customer_id']).tail(1)
```

	customer_id	product_id	product_title	product_category	star_rating	review_headline	purchased_counts
5812802	53095685	B00E8KLWB4	The Secret Society® - Hidden Mystery	Mobile Apps	5.0	Fun and well done hidden object game.	40945

```
# Evaluate Recommendations Made For the Same Customer Across Different Criterion Levels
collab(All, 53095685, 'B00E8KLWB4')
collab(multiple, 53095685, 'B00E8KLWB4')
collab(several, 53095685, 'B00E8KLWB4')
collab(many, 53095685, 'B00E8KLWB4')
```

1. Applying the “All” Filter

```
14914 other customers purchased this product.
Similar customers purchased 25435 other products.

We recommend these products from those similar customers:
0   Candy Crush Saga
1   Township
2   Dark Arcana: the Carnival
3   The Secret Society® - Hidden Mystery
4   Minion Rush: Despicable Me Official Game
5   Can You Escape
6   Letters From Nowhere®: A Hidden Object Mystery
7   Mystery Manor
8   Magic Jigsaw Puzzles
9   Alice in the Mirrors of Albion
```

2. Applying the “Multiple” Filter

```
----------------------------------------
6025 other customers purchased this product.
Similar customers purchased 16546 other products.

We recommend these products from those similar customers:
0   Township
1   Dark Arcana: the Carnival
2   The Secret Society® - Hidden Mystery
3   Minion Rush: Despicable Me Official Game
4   Can You Escape
5   Candy Crush Saga
6   Letters From Nowhere®: A Hidden Object Mystery
7   Magic Jigsaw Puzzles
8   Mystery Manor
9   Alice in the Mirrors of Albion
```

3. Applying the “Several” Filter

```
----------------------------------------
2951 other customers purchased this product.
Similar customers purchased 11337 other products.

We recommend these products from those similar customers:
0   Township
1   The Secret Society® - Hidden Mystery
2   Minion Rush: Despicable Me Official Game
3   Letters From Nowhere®: A Hidden Object Mystery
4   Candy Crush Saga
5   Magic Jigsaw Puzzles
6   Can You Escape
7   Mystery Manor
8   Midnight Castle – A Free Hidden Object Mystery Game for Fire! Find objects and solve puzzles!
9   Alice in the Mirrors of Albion
```

4. Applying the “Many” Filter

```
---------------------------------------
1685 other customers purchased this product.
Similar customers purchased 8315 other products.

We recommend these products from those similar customers:
0   Minion Rush: Despicable Me Official Game
1   Frozen Free Fall
2   Letters From Nowhere®: A Hidden Object Mystery
3   The Secret Society® - Hidden Mystery
4   Candy Crush Saga
5   Magic Jigsaw Puzzles
6   Can You Escape
7   Midnight Castle - A Free Hidden Object Mystery Game for Fire! Find objects and solve puzzles!
8   Mystery Manor
9   Alice in the Mirrors of Albion
```

5.5.4.1.2.3. Evaluating Recommendations Made on Products with Small Number of Sales

This final assessment investigates the performance of the recommender engine when a product has had relatively few sales. Due to the small number of buys, proposed recommendations vary less across different criterion levels.

The observed results are why, during the data cleaning phase, we had any product with less than 100 sales imputed.

```
# Find specific customer and product
All[All.purchased_counts == 1000][
    ['customer_id', 'product_id', 'product_title', 'product_category',
     'star_rating', 'review_headline', 'purchased_counts']
].head(1)
```

	customer_id	product_id	product_title	product_category	star_rating	review_headline	purchased_counts
22981	502696	B0001NBMBC	Vol. 3: The Subliminal Verses	Music	5.0	Awsome	1000

```
# Test specific customer and product among different thresholds
collab(All, 502696, 'B0001NBMBC')
collab(multiple, 502696, 'B0001NBMBC')
collab(several, 502696, 'B0001NBMBC')
collab(many, 502696, 'B0001NBMBC')
```

1. Applying the “All” Filter

```
--------------------------------------
497 other customers purchased this product.
Similar customers purchased 2266 other products.

We recommend these products from those similar customers:
0   Aenima
1   Ashes Of The Wake
2   Slipknot - Disasterpieces
3   Toxicity
4   All Hope Is Gone
5   Vol. 3: The Subliminal Verses
6   Iowa
7   Slipknot (EX)
8   Master of Puppets
9   Mezmerize
```

2. Applying the “Multiple” Filter

```
--------------------------------------
221 other customers purchased this product.
Similar customers purchased 1990 other products.

We recommend these products from those similar customers:
0   Aenima
1   Ashes Of The Wake
2   All Hope Is Gone
3   Vol. 3: The Subliminal Verses
4   Toxicity
5   Iowa
6   Slipknot (EX)
7   Slipknot - Disasterpieces
8   Mezmerize
9   Master of Puppets
```

3. Applying the “Several” Filter

```
---------------------------------------
150 other customers purchased this product.
Similar customers purchased 1868 other products.

We recommend these products from those similar customers:
0   Aenima
1   Ashes Of The Wake
2   All Hope Is Gone
3   Vol. 3: The Subliminal Verses
4   Vulgar Display Of Power
5   Iowa
6   Slipknot (EX)
7   Mezmerize
8   Toxicity
9   Master of Puppets
```

4. Applying the “Many” Filter

```
---------------------------------------
118 other customers purchased this product.
Similar customers purchased 1790 other products.

We recommend these products from those similar customers:
0   Aenima
1   Ashes Of The Wake
2   All Hope Is Gone
3   Vol. 3: The Subliminal Verses
4   Iowa
5   Slipknot (EX)
6   Reign In Blood
7   Mezmerize
8   Toxicity
9   Master of Puppets
```

5.5.4.2. Recommendation System Employing “Surprise”

Surprise is a Python scikit module that allows for developing as well as evaluating recommendation engines based on non-implicit ratings data.

Surprise allows the following objectives to be performed:

- Allow developers complete command over their Machine Learning model investigations. To that aim, Surprise has placed a heavy focus on thorough documentation highlighting each element of every algorithm used to build a recommender system.

- Reduce the agony of dealing with Datasets. Developers are also able to utilize either ready-made datasets (Movie lens and Jester) or other datasets, like the Amazon dataset employed in this project.
- Several prediction algorithms available include baseline algorithms, neighborhood approaches, matrix factorization based (SVD, PMF, SVD++, NMF), among others. Different similarity measurements (cosine, MSD, Pearson, etc.) are also included.
- Make it simple to introduce novel algorithmic concepts.
- Offer facilities enabling assessing, analyzing, and comparing the effectiveness of various algorithms. Cross-validation methods are fairly simple to implement utilizing sophisticated CV iterators (influenced by scikit-learn's comprehensive tools) and an extensive sweep across a group of parameters.

5.5.4.2.1. Selection of Best Performing Algorithm

5.5.4.2.1.1. Selection of Benchmark Model

The algorithms previously mentioned will be contrasted across each other for the purposes of determining which algorithm provides the lowest RMSE value, employing a randomized sample of 50000 review. These models will be used to train and test our random sample, then Cross Validation was performed utilizing RMSE as a measure for evaluating different algorithms on the sample.

The output can be seen in the table below, and it can be seen that Singular Value Decomposition (SVD) provides the least RMSE score and accordingly, SVD will be chosen as the benchmark algorithm. A benchmark model is a basic Machine Learning (ML) model with default parameter settings that developers use to compare the performance of their models to.

Algorithm	test_rmse	fit_time	test_time
SVD	1.113691	1.899741	0.128023
KNNBaseline	1.114587	57.526784	0.207765
BaselineOnly	1.115938	0.176921	0.093586
SVDpp	1.116863	3.660672	0.100597
KNNBasic	1.155447	56.116376	0.166689
SlopeOne	1.160438	1.202418	0.129246
KNNWithMeans	1.160772	60.187209	0.326530
CoClustering	1.161342	4.473748	0.090783
NMF	1.165692	4.658261	0.096936
NormalPredictor	1.469713	0.048221	0.153851

5.5.4.2.1.2. Cross Validation and Hyperparameter Tuning

A data resampling technique called cross-validation is used to evaluate the generalizability of prediction models and avoid overfitting. Cross-validation is a member of the school of Monte Carlo techniques, much like bootstrapping. Theoretically speaking, a predictive algorithm is a method for classifying a situation relying on a collection of facts. The creation of that model is referred to as supervised learning in machine learning. The reliability of the generated model is a key consideration in supervised learning. The main issue in this case is overfitting. It is relatively simple to create a model which is precisely tailored to the current dataset and yet is subsequently incapable of generalizing adequately to new, unobserved data. How then should we evaluate a model's capacity for generalization? In a perfect world, we could test the model's performance and accuracy employing new unseen data which came from an identical population to that employed when training the model. However, in reality, additional independent validation investigations are frequently not practical. Additionally, prior to spending time and various resources to attempt external validation, it's recommended to assess the predictive capability of the model first.

That is generally accomplished using techniques for data resampling, like cross-validation.

Accordingly, cross-validation was performed on the benchmark model using its default parameter values.

A randomized sample of 10,000 reviews was chosen, not all goods are included in the sample though. Hyperparameter tuning is then later performed on the cross-validated benchmark model using the random sample, to see if we can improve the RMSE score for the entire dataset.

In order to do this, the dataset was divided into train and test groups before loading and fitting the model to the data.

The optimal hyperparameters for the sample were identified using a GridSearch and will be later applied to the entire dataset.

The result to that process can be seen in the following figure:

```
{'rmse': 1.1193523328771364}
{'rmse': {'n_factors': 75, 'n_epochs': 100, 'lr_all': 0.002, 'reg_all': 0.01}}
CPU times: user 5min 32s, sys: 3.76 s, total: 5min 36s
Wall time: 5min 47s
```

The next step comprises of splitting the entire dataset into training and testing groups to compare the performance of our customized SVD model to the benchmark model.

```
RMSE: 1.0593
1.059264022518196
CPU times: user 4min 25s, sys: 4.85 s, total: 4min 30s
Wall time: 4min 43s
```

It can be seen that the tuned model is performing better than the benchmark model.

5.5.4.2.1.3. Creation of Recommender Engine

Having validated that our tuned model outperforms the benchmark SVD model, buyer star reviews for merchandise items will be forecasted; such that we enhance the recommendation system's capability for making customized recommendations.

The tuned model will be employed to provide the best 10 recommendations per customer.

The following steps were taken to build the recommender:

- The tuned model, as previously shown, will generate N recommendations for each customer.

- A dictionary will be created having its key as the unique customer ID and the key value would encompass a tuple of each product ID and the predicted star rating for that particular product, and that particular customer.
- The products, for each customer, will be arranged descendingly according to the predicted rating.
- The model will finally propose the products with highest predicted rating.

The output for randomly chosen 5 customers with 5 proposed recommendations each with the predicted product rating, shown in a dataframe for easier visibility:

	0	1	2	3	4
16344847	(The Hunger Games (Hunger Games Trilogy, Book ...	(Downton Abbey Season 3, 4.900347609925889)	(John Adams Season 1, 4.845003612820365)	(Me Before You: A Novel, 4.81854353869497)	(Band of Brothers Season 1, 4.7887700179517205)
30678701	(Downton Abbey Season 3, 4.802539376502293)	(The Pacific Season 1, 4.72619072474924)	(The Hunger Games (Hunger Games Trilogy, Book ...	(Lone Survivor: The Eyewitness Account of Oper...	(Downton Abbey Season 5, 4.687908580319665)
16465135	(Downton Abbey Season 3, 4.922477036291557)	(Band of Brothers Season 1, 4.872192006810055)	(Crossy Road, 4.835400370890903)	(The Hunger Games (Hunger Games Trilogy, Book ...	(The Room (Kindle Tablet Edition), 4.806700331...
45438862	(Wonder, 4.992777341756024)	(Hopeless, 4.98681001560101)	(The Pacific Season 1, 4.966017198459856)	(Downton Abbey Season 3, 4.965416747581851)	(Crossy Road, 4.958280391767791)
35081900	(Downton Abbey Season 3, 5)	(Crossy Road, 4.957932083457072)	(Me Before You: A Novel, 4.9098126796462065)	(Downton Abbey Season 5, 4.893141600858727)	(Five Nights at Freddy's, 4.869514455666185)

Chapter 6. References

1) Wilkins, J. (2021, August 23). *Front End Developer – What is Front End Development, Explained in Plain English*. freeCodeCamp.Org. https://www.freecodecamp.org/news/front-end-developer-what-is-front-end-development-explained-in-plain-english/
2) *What is Frontend? What is Backend?* (2019, November 1). Frontend GmbH. https://www.frontend-gmbh.de/en/blog/what-is-frontend-what-is-backend/
3) C. (2022, May 26). *UI vs. UX Design: What's the Difference?* Coursera. https://www.coursera.org/articles/ui-vs-ux-design
4) Sarosa, A. (2022, April 29). *What Is HTML? Hypertext Markup Language Basics Explained*. Hostinger Tutorials. https://www.hostinger.com/tutorials/what-is-html#What_Is_HTML
5) *Introduction to HTML*. (2021). W3Schools. https://www.w3schools.com/html/html_intro.asp
6) *What is CSS? - Learn web development | MDN*. (2022, May 30). Mdn Web Docs. https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/What_is_CSS
7) *CSS Introduction*. (2019). W3Schools. https://www.w3schools.com/css/css_intro.asp

8) *JavaScript | MDN*. (2022, June 16). Mdn Web Docs. https://developer.mozilla.org/en-US/docs/Web/JavaScript
9) *First-class Function - MDN Web Docs Glossary: Definitions of Web-related terms | MDN*. (2021, October 8). Mdn Web Docs. https://developer.mozilla.org/en-US/docs/Glossary/First-class_Function
10) *What is JavaScript? - Learn web development | MDN*. (2022, March 23). Mdn Web Docs. https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript
11) Pandit, N. (2021, February 10). *What And Why React.js*. C#corner. https://www.c-sharpcorner.com/article/what-and-why-reactjs/
12) Lutkevich, B. (2020, December 8). *embedded system*. IoT Agenda. https://www.techtarget.com/iotagenda/definition/embedded-system
13) Contributor, T. (2019, August 14). *processor (CPU)*. WhatIs.Com. https://www.techtarget.com/whatis/definition/processor
14) Mazidi, M. A., Naimi, S., & Naimi, S. (2017). *The AVR microcontroller and Embedded systems: Using Assembly and C* (First Edition). MicroDigitalEd.
15) ATMEL, ATMEGA16, 2466, 2009.
16) ESPRESSIF, ESP32 Series, V3.4, 2020
17) *Load Cell Amplifier HX711 Breakout Hookup Guide - learn.sparkfun.com*. (2022). SARAH AL-MUTLAQ. Retrieved June 30, 2022, from https://learn.sparkfun.com/tutorials/load-cell-amplifier-hx711-breakout-hookup-guide?_ga=2.231763804.2108468963.1656547845-610604690.1649381703
18) *Getting Started with Load Cells - learn.sparkfun.com*. (2022). SARAH AL-MUTLAQ. https://learn.sparkfun.com/tutorials/getting-started-with-load-cells?_ga=2.55665259.1189879391.1651968187-610604690.1649381703
19) Grow, GM65, Bar Code Reader Module.
20) AVIA Semiconductors, HX711, 2022
21) Christie, T. (n.d.). *Home - Django REST framework*. Django Rest Fram Work. https://www.django-rest-framework.org/
22) *fw_error_www*. (n.d.). Database. https://www.oracle.com/database/what-is-database/#:%7E:text=A%20database%20is%20an%20organized,database%20management%20system%20(DBMS).
23) *The web framework for perfectionists with deadlines | Django*. (n.d.). Django. https://www.djangoproject.com/
24) *How to build a public facing API using AWS*. (n.d.). Amazon Web Services, Inc. https://aws.amazon.com/what-is/api/#:%7E:text=API%20stands%20for%20Application%20Programming,other%20using%20requests%20and%20responses.
25) J. Son and S. B. Kim, "Academic paper recommender system using multilevel simultaneous citation networks," *Decis. Support Syst.*, vol. 105, pp. 24–33, 2018, doi: https://doi.org/10.1016/j.dss.2017.10.011.

26) A. Gunawardana and G. Shani, “Evaluating Recommender Systems,” in *Recommender Systems Handbook*, F. Ricci, L. Rokach, and B. Shapira, Eds. Boston, MA: Springer US, 2015, pp. 265–308.
27) L. Lü, M. Medo, C. H. Yeung, Y.-C. Zhang, Z.-K. Zhang, and T. Zhou, “Recommender systems,” *Phys. Rep.*, vol. 519, no. 1, pp. 1–49, 2012, doi: https://doi.org/10.1016/j.physrep.2012.02.006.
28) Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan,“Large-Scale Parallel Collaborative Filtering for the Netflix Prize,” in *Algorithmic Aspects in Information and Management*, 2008, pp. 337–348.
29) D. Anand and R. Scholar, “A Survey And Learning Techniques On Access Controls In Cloud Computing,” *Int. J. Res. Advent Technol. Spec. Issue*.
30) K. Singhal and A. Anand, “Monetisation Of Youtube Content Using Data Mining Techniques,” *Int. J. Sci. Technol. Res.*, vol. 9, p. 2, 2020.
31) A. J. C. Trappey, C. V Trappey, C.-Y. Wu, C. Y. Fan, and Y.-L. Lin, “Intelligent patent recommendation system for innovative design collaboration,” *J. Netw. Comput. Appl.*, vol. 36, no. 6, pp. 1441–1450, 2013, doi: https://doi.org/10.1016/j.jnca.2013.02.035.