Front Page

Mîț Alexia Teodora

Bachelor's Computer Science Program

2024-2025 Academic Year

NHL Stenden University of Applied Sciences

Personal email: alexia2207@yahoo.com

School email: alexia.mit@student.nhlstenden.com

Number of EC's: 0/180

Table of Contents

Introduction

My name is Mîț Alexia Teodora, and I am 19 years old. I moved from Romania to The Netherlands by myself so I can follow the Computer Science program at NHL Stenden University of Applied Sciences. I chose to come here, because it has a different way of teaching than the Universities from my country. In my opinion, the decision that I made will give me more opportunities to pursue my future career.

This portfolio will show in detail my progress and the skills I gained while following the chosen program.

In addition, my GitHub name is Alexia220700.

Week 1

Computer Science – Python

Hello

In a file called hello.py in a folder called sentimental-hello, implement a program that prompts a user for their name, and then prints hello, so-and-so, where so-and-so is their provided name.

This is the code for the assignment:

```
from cs50 import get_string

answer = get_string("What's your name?  ")
print("hello, " + answer)
```

The Harvard score with the title of the program and my name:

me50 / users / Alexia220700 / cs50 / problems / 2024 / x / sentimental / hello

My Submissions | My Courses | Docs | Log Out

#1 submitted 8 days ago, Monday, November 11, 2024 9:49 PM CET
check50 3/3 • style50 1.00 • 0 comments
tar.gz • zip

https://submit.cs50.io/check50/5ccf69fc1ec306d275c1c77437c9ab6972a9d038

Mario – more

In a file called mario.py in a folder called sentimental-mario-more, write a program that recreates a half-pyramid using hashes (#) for blocks.

This is the code for the assignment:

```
from cs50 import get_int

# ask for correct input height
height = get_int("Choose a height between 1-8")

while (height < 1 or height > 8):
    height = get_int("Choose a height between 1-8")

i = 0
j = 0

# i goes through rows
for i in range(height):
    # j goes through collumns
    for j in range(height - i - 1):
        # prevents the newline and instead specify a different string to be appended at the
end of the output
        # used to print multiple items on the same line
        print(" ", end="")

# print hashes for the pyramid
    for j in range(i + 1):
        print("#", end="")

# print the gap between the two parts of the pyramid
    print("  ", end="")

# print the right side hashes
    for j in range(i + 1):
        print("#", end="")

    # move to a new line after every row
    print()
```

The Harvard score with the title of the program and my name:

me50 / users / Alexia220700 / cs50 / problems / 2024 / x / sentimental / mario / more

#1 submitted 6 days ago, Wednesday, November 13, 2024 11:56 AM CET
check50 9/9 • style50 1.00 • 0 comments
tar.gz • zip

https://submit.cs50.io/check50/3004fd3fbd1488b0687bcd66075134a35e73d9fe

Credit

In a filed called credit.py in a folder called sentimental-credit, write a program that prompts the user for a credit card number and then reports (via print) whether it is a valid American Express, MasterCard, or Visa card number.

This is the code for the assignment:

```
from cs50 import get_int

card_number = get_int("Enter a card number: ")
aux = card_number
first2_nr = card_number

# counting the length of card number
length = 0
while (card_number != 0):
    card_number = card_number // 10
    length += 1

# sum of digits
sum = 0

# counts position of number
alternate = 1

while aux > 0:
    digit = aux % 10
    aux = aux // 10

    if alternate % 2 == 0:
        # multiplies every other digit by 2, starting with the number's
        # second-to-last digit
        digit = digit * 2

        if digit > 9:
            digit = digit - 9

    sum = sum + digit
    alternate += 1
```

```
# check if the length and first two digits are correct
# or only the first digit for VISA
if length == 15 and (first2_nr // 10000000000000 == 34 or first2_nr // 10000000000000 == 37)
and sum % 10 == 0:
    print("AMEX")
elif length == 16 and (first2_nr // 100000000000000 >= 51 and first2_nr // 100000000000000 <=
55) and sum % 10 == 0:
    print("MASTERCARD")
elif (length == 13 or length == 16) and (first2_nr // 1000000000000 == 4 or first2_nr //
1000000000000000 == 4) and sum % 10 == 0:
    print("VISA")
else:
    print("INVALID")
```

The Harvard score with the title of the program and my name:

me50 / users / Alexia220700 / cs50 / problems / 2024 / x / sentimental / credit

#2 submitted 5 days ago, Thursday, November 14, 2024 12:11 PM CET
check50 14/14 • style50 1.00 • 0 comments
tar.gz • zip

#1 submitted 5 days ago, Thursday, November 14, 2024 12:09 PM CET
check50 14/14 • style50 0.91 • 0 comments
tar.gz • zip

Readability

Write, in a file called readability.py in a folder called sentimental-readability, a program that first asks the user to type in some text, and then outputs the grade level for the text, according to the Coleman-Liau formula.

This is the code for the assignment:

```
from cs50 import get_string

text = get_string("Input a text to be verified: ")

letters = 0
# the first word is not counted
words = 1
sentences = 0

i = 0
# len is strlen from C, but in Python
for i in range(len(text)):
    # check if the character is a letter
    if text[i].isalpha():
        letters += 1

    # checks for sentences
    elif text[i] == "." or text[i] == "!" or text[i] == "?":
        sentences += 1

    # checks for spaces to count words
    elif text[i].isspace():
        words += 1

# calculate L
# average number of letters per 100 words in the text
L = letters / words * 100

# calculate S
# average number of sentences per 100 words in the text
S = sentences / words * 100
```

```
# rounf the index for the grades
index = round(0.0588 * L - 0.296 * S - 15.8)

# print the grade level
if index < 1:
    print("Before Grade 1")
elif index >= 16:
    print("Grade 16+")
else:
    print("Grade ", + index)
```

The Harvard score with the title of the program and my name:

me50 / users / Alexia220700 / cs50 / problems / 2024 / x / sentimental / readability

#2 submitted 5 days ago, Thursday, November 14, 2024 8:41 PM CET
check50 10/10 • style50 1.00 • 0 comments
tar.gz • zip

#1 submitted 5 days ago, Thursday, November 14, 2024 8:40 PM CET
check50 10/10 • style50 0.93 • 0 comments
tar.gz • zip

https://submit.cs50.io/check50/11c72e8a4aabb6f72e289c4c9c0a4369b6fbe36b

DNA

In a file called dna.py in a folder called DNA, implement a program that identifies to whom a sequence of DNA belongs.

- The program should require as its first command-line argument the name of a CSV file containing the STR counts for a list of individuals and should require as its second command-line argument the name of a text file containing the DNA sequence to identify.
- If your program is executed with the incorrect number of command-line arguments, your program should print an error message of your choice (with print). If the correct number of arguments are provided, you may assume that the first argument is indeed the filename of a valid CSV file and that the second argument is the filename of a valid text file.
- Your program should open the CSV file and read its contents into memory.
- You may assume that the first row of the CSV file will be the column names. The first column will be the word name, and the remaining columns will be the STR sequences themselves.
- Your program should open the DNA sequence and read its contents into memory.
- For each of the STRs (from the first line of the CSV file), your program should compute the longest run of consecutive repeats of the STR in the DNA sequence to identify. Notice that we've defined a helper function for you, longest_match, which will do just that!
- If the STR counts match exactly with any of the individuals in the CSV file, your program should print out the name of the matching individual.
- You may assume that the STR counts will not match more than one individual.
- If the STR counts do not match exactly with any of the individuals in the CSV file, your program should print No match.

This is the code for the assignment:

```
import csv
import sys



def main():
    # Check for command-line usage
    if len(sys.argv) != 3:
        print("Usage: python3 dna.py database.csv sequence.txt")
        sys.exit(1)  # exit if there are not exactly two arguments

    # Read the database file into a variable
    rows = []
    with open(sys.argv[1], newline='') as file:
        # creates a list of dictionaries from the file
        reader = csv.DictReader(file)
        for row in reader:
            rows.append(row)

    # Read DNA sequence file into a variable
    with open(sys.argv[2], 'r') as f:
        dna_sequence = f.read()

    # Extract STR sequences from the first row of the CSV (exclude 'name' column)
    # 1: slicing syntax in Python that returns all elements starting from the second element
    str_sequences = reader.fieldnames[1:]

    # Find longest match of each STR in the DNA sequence
    str_counts = {}
    for str_sequence in str_sequences:
    # iterating over the str_sequences list and using a function (longest_match()) to compute
the longest match for each sequence of repeating DNA bases in the dna_sequence
    # the result is stored in a dictionary called str_counts, where the keys are the sequence
names
        str_counts[str_sequence] = longest_match(dna_sequence, str_sequence)

    # Check database for matching profiles
    match_found = False
    # iterate over a list
    for row in rows:
        # Check if STR counts match the individual in the database
        match = True
```

```
        for str_sequence in str_sequences:
          # comparing two values: the value from the row for a specific key (identified by
str_sequence) and the corresponding value from the str_counts dictionary
            if int(row[str_sequence]) != str_counts[str_sequence]:

                match = False
                break

        if match:
            print(row['name'])
            match_found = True
            break  # exit the loop after finding the first match

    # If no match was found
    if not match_found:
        print("No match")

# designed to find the longest consecutive run of a specific subsequence within a given sequence
def longest_match(sequence, subsequence):
    """Returns length of longest run of subsequence in sequence."""
    longest_run = 0
    subsequence_length = len(subsequence)
    sequence_length = len(sequence)

    # Check each character in sequence for the most consecutive runs of subsequence
    for i in range(sequence_length):
        count = 0
        # Check for a subsequence match in a "substring" (a subset of characters) within
sequence
        while True:
            start = i + count * subsequence_length
            end = start + subsequence_length

            # If there is a match in the substring
            if sequence[start:end] == subsequence:
                count += 1
            else:
                break

        # Update most consecutive matches found
        longest_run = max(longest_run, count)

    return longest_run
```

```python
# This condition checks whether the Python script is being run directly (as the main program) or if it is being imported into another script
if __name__ == "__main__":
    main()
```

The Harvard score with the title of the program and my name:

me50 / users / Alexia220700 / cs50 / problems / 2024 / x / dna

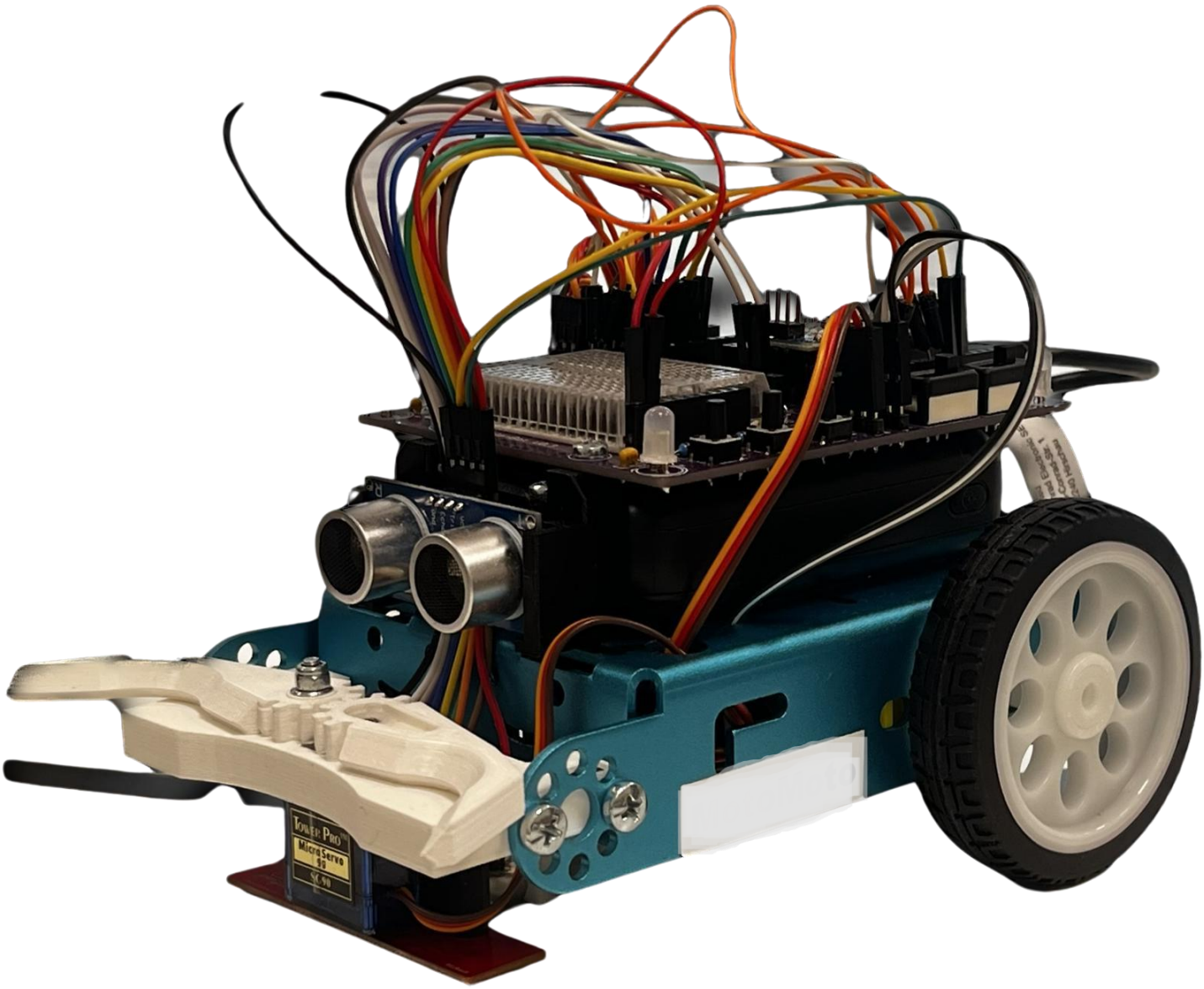
My Submissions | My Courses | Docs | Log Out

#2 submitted a few seconds ago, Wednesday, January 15, 2025 10:06 PM CET
check50 21/21 • style50 1.00 • 0 comments
tar.gz • zip

#1 submitted 8 hours ago, Wednesday, January 15, 2025 1:51 PM CET
check50 21/21 • style50 1.00 • 0 comments
tar.gz • zip

https://submit.cs50.io/check50/be3a07c016613a9c1b74dabf42b80a8835b4341b

Embedded Systems



Basic Movements

For this assignment, the robots had to be programmed so they could move forwards, move backwards, turn left, turn right, rotate left, rotate right, stop.

This is the code for the assignment:

```
#include "Arduino.h"

// Motor speed definitions
#define motorLFFullSpeed 220
#define motorRFFullSpeed 220
#define motorLBFullSpeed 253
#define motorRBFullSpeed 253
#define motorLHalfSpeed 120
#define motorRHalfSpeed 120
#define motorStop 0

// Motor PWM speed pins (must be PWM-capable pins)
#define motorLB 11 // motor left backwards
#define motorRF 9  // motor right forwards
#define motorLF 10 // motor left forwards
#define motorRB 3   // motor right backwards



void setup()
{
  // Set motor direction pins as outputs
  pinMode(motorLF, OUTPUT);
  pinMode(motorRF, OUTPUT);
  pinMode(motorLB, OUTPUT);
  pinMode(motorRB, OUTPUT);
}

void loop()
{

  moveForwards();
  delay(1000);
  stopMotors();
```

```
  delay(500);
  moveBackwards();
  delay(1000);
  stopMotors();
  delay(500);

  rotRigth();
  delay(500);
  stopMotors();
  delay(500);
  rotLeft();
  delay(500);
  stopMotors();
  delay(500);



  turnLeft();
  delay(1500);
  stopMotors();
  delay(500);
  turnRigth();
  delay(1500);
  stopMotors();
  delay(500);

  stopMotors();
}

void moveForwards()
{
  analogWrite(motorLF, motorLFFullSpeed);
  analogWrite(motorRF, motorRFFullSpeed);
  analogWrite(motorLB, motorStop);
  analogWrite(motorRB, motorStop);
}

void moveBackwards()
{
  analogWrite(motorLB, motorLBFullSpeed);
  analogWrite(motorRB, motorRBFullSpeed);
  analogWrite(motorLF, motorStop);
  analogWrite(motorRF, motorStop);
}
```

```
void stopMotors()
{
  analogWrite(motorRB, motorStop);
  analogWrite(motorLB, motorStop);
  analogWrite(motorLF, motorStop);
  analogWrite(motorRF, motorStop);
}

void rotRigth()
{
  analogWrite(motorRB, motorLBFullSpeed);
  analogWrite(motorLB, motorStop);
  analogWrite(motorLF, motorLBFullSpeed);
  analogWrite(motorRF, motorStop);
}

void rotLeft()
{
  analogWrite(motorRF, motorLBFullSpeed);
  analogWrite(motorLF, motorStop);
  analogWrite(motorLB, motorLBFullSpeed);
  analogWrite(motorRB, motorStop);
}

void turnRigth()
{
  analogWrite(motorRB, motorStop);
  analogWrite(motorLB, motorStop);
  analogWrite(motorLF, motorLFFullSpeed);
  analogWrite(motorRF, motorRHalfSpeed);
}

void turnLeft()
{
  analogWrite(motorRF, motorRFFullSpeed);
  analogWrite(motorLF, motorLHalfSpeed);
  analogWrite(motorLB, motorStop);
  analogWrite(motorRB, motorStop);
}
```

Logbook from date to date

Monday	I made Hello, Mario More assignments.
Tuesday	I made Credit.
Wednesday	I worked on Readability.
Thursday	I finished Readability and started DNA.
Friday	I kept trying to make DNA assignment.
Saturday	I couldn’t manage to make the assignment.
Sunday	I finished DNA.

Self-reflection

In my opinion, the first week of Period 2 was easy. I enjoyed working with Python for my Computer Science assignments. We only had to translate our projects from C programming language into Python.

For Embedded Systems we had to program a RelayBot. Me and Fabiana decided to work on the first one from our group, the Line-Follower. It was a bit of a struggle with finding the correct pins for the robot, but we asked for help from our teammates. Besides that, we managed to work on our code and show the robot so we could get our papers stamped. The robot had to move backwards, forwards, turn left/right, rotate left/right, stop.

Week 2

Computer Science

Songs

Write SQL queries to answer questions about a database of the 100 most-streamed songs on Spotify in 2018.

This is the code for the assignment:

4. sql:

Task:

In 4.sql, write a SQL query that lists the names of any songs that have danceability, energy, and valence greater than 0.75.

Your query should output a table with a single column for the name of each song.

```
-- retrieve the name of the songs from the songs table --
SELECT songs.name
FROM songs
WHERE danceability > 0.75 AND energy > 0.75 AND valence > 0.75
```

7. sql:

In 7.sql, write a SQL query that returns the average energy of songs that are by Drake.

Your query should output a table with a single column and a single row containing the average energy.

You should not make any assumptions about what Drake's artist_id is

```
SELECT AVG(songs.energy)
FROM songs
JOIN artists ON artists.id = songs.artist_id
WHERE artists.name = "Drake"
```

8. sql:

In 8.sql, write a SQL query that lists the names of the songs that feature other artists.

Songs that feature other artists will include "feat." in the name of the song.

Your query should output a table with a single column for the name of each song.

```
SELECT songs.name
FROM songs
WHERE songs.name LIKE "%feat.%"
```

The Harvard score with the title of the program and my name:

me50 / users / Alexia220700 / cs50 / problems / 2024 / x / songs

My Submissions | My Courses | Docs | Log Out

#1 submitted 2 hours ago, Friday, November 22, 2024 9:44 AM CET
check50 11/11 • 0 comments
tar.gz • zip

https://submit.cs50.io/check50/364d14b16a1fd50dcc22c13339f8239129b9a3f3

Movies

Provided to you is a file called movies.db, an SQLite database that stores data from IMDb about movies, the people who directed and starred in them, and their ratings. Write SQL queries to answer questions about this database of movies.

This is the code for the assignment:

8. sql:

In 8.sql, write a SQL query to list the names of all people who starred in Toy Story.

Your query should output a table with a single column for the name of each person.

You may assume that there is only one movie in the database with the title Toy Story.

```
SELECT people.name
FROM people
JOIN stars ON stars.person_id = people.id
JOIN movies ON movies.id = stars.movie_id
WHERE movies.title = "Toy Story";
```

9. sql:

In 9.sql, write a SQL query to list the names of all people who starred in a movie released in 2004, ordered by birth year.

Your query should output a table with a single column for the name of each person.

People with the same birth year may be listed in any order.

No need to worry about people who have no birth year listed, so long as those who do have a birth year are listed in order.

If a person appeared in more than one movie in 2004, they should only appear in your results once.

```
SELECT movies.title
FROM movies
JOIN stars ON stars.movie_id = movies.id
JOIN people ON people.id = stars.person_id
JOIN ratings ON movies.id = ratings.movie_id
WHERE people.name = "Chadwick Boseman"
LIMIT 5;
```

13. sql:

In 13.sql, write a SQL query to list the names of all people who starred in a movie in which Kevin Bacon also starred.

Your query should output a table with a single column for the name of each person.

There may be multiple people named Kevin Bacon in the database. Be sure to only select Kevin Bacon, born in 1958.

Kevin Bacon himself should not be included in the resulting list.

```sql
-- stars, movies, people
SELECT DISTINCT people.name
FROM people
JOIN stars ON stars.person_id = people.id
JOIN movies ON movies.id = stars.movie_id
WHERE movies.id IN
(
    SELECT movie_id
    FROM stars
    JOIN people ON stars.person_id = people.id
    WHERE people.name = "Kevin Bacon" AND people.birth = 1958
) AND people.name != "Kevin Bacon";
```

The Harvard score with the title of the program and my name:

me50 / users / Alexia220700 / cs50 / problems / 2024 / x / movies

My Submissions | My Courses | Docs | Log Out

#1 submitted 21 hours ago, Thursday, November 21, 2024 3:00 PM CET

check50 14/14 • 0 comments

tar.gz • zip

https://submit.cs50.io/check50/3470851df95b88966d83c86eb2b04bb46f78eda0

Fiftyville

The CS50 Duck has been stolen! Authorities believe that the thief stole the duck and then, shortly afterwards, took a flight out of town with the help of an accomplice. Your goal is to identify:

- Who the thief is,
- What city the thief escaped to, and
- Who the thief's accomplice is who helped them escape

All you know is that the theft took place on July 28, 2023, and that it took place on Humphrey Street.

This is the code for the assignment:

```
-- find the names of the witnesses and the description of the crime
SELECT interviews.name AS witness_name, crime_scene_reports.description
FROM interviews
JOIN crime_scene_reports ON crime_scene_reports.id = interviews.id
WHERE interviews.year = 2023
AND interviews.month = 7
AND interviews.day = 28
AND transcript LIKE '%bakery%';
-- witnesses: Ruth, Eugene, Raymond

-- to see the timetable of the theft
-- hint: theft took place at 10:15am
SELECT crime_scene_reports.description
FROM crime_scene_reports
WHERE year = 2023
AND month = 7
AND day = 28;

-- using this to see the transcripts of the witnesses
-- Ruth: gives the hint about the parking and license plate
-- Eugene gives the hint about the ATM
-- Raymond gives the hint about the phone call, that took < 1 minute
SELECT name AS witness_name, transcript
FROM interviews
```

```sql
JOIN crime_scene_reports ON crime_scene_reports.id = interviews.id
WHERE interviews.year = 2023
AND interviews.month = 7
AND interviews.day = 28
AND transcript LIKE '%bakery%';

-- find the names of the people that parked there
-- in plus, see the hour and minute of entrance and exit
-- check if it matches the timetable for theft
SELECT people.name AS thief_name, bakery_security_logs.license_plate,
bakery_security_logs.hour, bakery_security_logs.minute, bakery_security_logs.activity
FROM people
JOIN bakery_security_logs ON bakery_security_logs.license_plate = people.license_plate
-- to narrow down the list
WHERE bakery_security_logs.year = 2023
AND bakery_security_logs.month = 7
AND bakery_security_logs.day = 28;

-- check if a person from the other list withdrew money from that certain ATM
-- narrow down the list by using the location of the atm
SELECT DISTINCT people.name AS possible_thief_name, atm_transactions.transaction_type,
atm_transactions.atm_location
FROM atm_transactions
JOIN bank_accounts ON atm_transactions.account_number = atm_transactions.account_number
JOIN people ON people.id = bank_accounts.person_id
WHERE atm_transactions.year = 2023
AND atm_transactions.month = 7
AND atm_transactions.day = 28
AND atm_transactions.atm_location = 'Leggett Street';
-- Taylor, Diana, Bruce

-- check the phone call that took under one minute
SELECT people.name, phone_calls.duration, phone_calls.receiver
FROM phone_calls
JOIN people ON people.phone_number = phone_calls.caller
WHERE phone_calls.year = 2023
AND phone_calls.month = 7
AND phone_calls.day = 28;
-- Bruce, Taylor, Diana
-- Bruce's phone call took 45 seconds

-- join people table and passengers table through passport_number to find matching names
-- with the other lists
SELECT people.name, flights.origin_airport_id, flights.destination_airport_id
```

```sql
FROM people
JOIN passengers ON passengers.passport_number = people.passport_number
-- join passengers table with flights table
JOIN flights ON flights.id = passengers.flight_id
WHERE flights.year = 2023
AND flights.month = 7
AND flights.day = 29;
-- thief is Bruce

-- find out where the thief escaped to
SELECT airports.city
FROM airports
JOIN flights ON flights.destination_airport_id = airports.id
WHERE flights.year = 2023
AND flights.month = 7
AND flights.day = 29
AND flights.destination_airport_id = 4;
-- New York City

-- find the name of the accomplice
-- compare Bruce's phone call that took 45 seconds
-- with the phone calls of the receivers
SELECT people.name, phone_calls.duration
FROM phone_calls
JOIN people ON people.phone_number = phone_calls.receiver
WHERE phone_calls.year = 2023
AND phone_calls.month = 7
AND phone_calls.day = 28;
-- accomplice is Robin
```

The Harvard score with the title of the program and my name:

me50 / users / Alexia220700 / cs50 / problems / 2024 / x / fiftyville

My Submissions | My Courses | Docs | Log Out

#1 submitted 8 hours ago, Tuesday, November 26, 2024 2:29 PM CET

check50 3/3 • 0 comments

tar.gz • zip

https://submit.cs50.io/check50/c0f58ff621933b322a6dfe52c53c3c1b397df03

The is a part of my notes:

```
FIFTYVILLE

Witnesses: Ruth, Eugene, Raymond
Theft took place on 28 July
ATM Location: Legget Street
10:15 AM Bakery Street

License plate:
Names of people with that license plate: George, Michael , Kelsey,  Carolyn, , Martha,

Matching license plate w ATM transaction on 28 July 2023:
Robin, Ethan, Debra,  Andrew, Ralph, Jeremy, Alice,  Brandon, Peter, Bruce, Wayne, Sophi

Atm Transactions: Samuel, Janice, Bruce, Cheryl, Wayne, Alan, Jennifer, Harlod,Christian
Logan, Amanda, Heather, Jordan, Frances, Karen, Diana, Cynthia, Linda, Rachel, Douglas,
Laura, Brooke, Jack, Judy, Raymond, Ernest, Jerry, Denise, Nathan etc

Matching names from the 2 lists:
Robin, Ethan, Debra, Andrew, Ralph, Jeremy, Alice, Brandon, Eugene!, Peter, Bruce, Wayne
Taylor, John

Phone calls around 1 min:
*should be less than a minute
Sofia 51 s
Kelsey 36 s
Bruce 45 s
Kathryn 60 s
Jason 69 s
Kelsey 50 s
Taylor 43 s
Diana 49 s
Harold 67 s
Peter 61 s
John 88 s
Kenny 55 s
Bruce 75 s

possible thieves from the 3 lists combined: Bruce,  Taylor, Diana

flight ticket on 29: Diana 6 flight_id, Taylor 4, Bruce 4
```

Embedded Systems

Object avoidance

```
#include "Arduino.h"

 // Motor speed definitions

#define motorLFFullSpeed 255

#define motorRFFullSpeed 255

#define motorLBFullSpeed 255

#define motorRBFullSpeed 255

#define motorLHalfSpeed 150

#define motorRHalfSpeed 150

#define motorStop 0



// Motor PWM speed pins (must be PWM-capable pins)

#define motorLB 11 // motor left backwards

#define motorRF 9  // motor right forwards

#define motorLF 10 // motor left forwards

#define motorRB 3   // motor right backwards

#define TRIG_PIN 12

#define ECHO_PIN 13


void setup()

{
```

```
  // Set motor direction pins as outputs
  pinMode(motorLF, OUTPUT);
  pinMode(motorRF, OUTPUT);
  pinMode(motorLB, OUTPUT);
  pinMode(motorRB, OUTPUT);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  Serial.begin(9600); // For debugging
}
void loop()
{
   // Measure distance
  long duration, distance;
  digitalWrite(TRIG_PIN,  LOW);
  delayMicroseconds(2);          // Ensure trigger is off
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);         // Send a 10µs pulse
  digitalWrite(TRIG_PIN, LOW);
  duration = pulseIn(ECHO_PIN, HIGH);

  // Calculate distance in cm
  distance = duration * 0.034 / 2;
```

```
  // Debugging output
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");


  //  Check distance
  if (distance > 0 && distance <= 20) {
    stopMotors();   // Stop motors if object is close
    delay(200);
    turnRigth();
    delay(1000);
    moveForwards();
    delay(200);
    turnLeft();
    delay(5000);
    stopMotors();
    delay(200);
    turnRigth();
delay(1000);
  } else {
    moveForwards(); // Keep moving forwards otherwise
    Serial.println("No object detected. Moving forwards.");
```

```
  }
  delay(100); // Small delay to stabilize readings
}

void moveForwards()
{
  analogWrite(motorLF, motorLFFullSpeed);
  analogWrite(motorRF, motorRFFullSpeed);
  analogWrite(motorLB, motorStop);
  analogWrite(motorRB, motorStop);
}

void moveBackwards()
{
  analogWrite(motorLB, motorLBFullSpeed);
  analogWrite(motorRB, motorRBFullSpeed);
  analogWrite(motorLF, motorStop);
  analogWrite(motorRF, motorStop);
}

void stopMotors()
{
  analogWrite(motorRB, motorStop);
```

```
  analogWrite(motorLB, motorStop);
  analogWrite(motorLF, motorStop);
  analogWrite(motorRF, motorStop);
}

void rotRigth()
{
  analogWrite(motorRB, motorLBFullSpeed);
  analogWrite(motorLB, motorStop);
  analogWrite(motorLF, motorLBFullSpeed);
  analogWrite(motorRF, motorStop);
}

void rotLeft()
{
  analogWrite(motorRF, motorLBFullSpeed);
  analogWrite(motorLF, motorStop);
  analogWrite(motorLB, motorLBFullSpeed);
  analogWrite(motorRB, motorStop);
}

void turnRigth()
{
```

```
  analogWrite(motorRB, motorStop);

  analogWrite(motorLB, motorStop);

  analogWrite(motorLF, motorLFFullSpeed);

  analogWrite(motorRF, motorRHalfSpeed);

}


void turnLeft()

{

  analogWrite(motorRF, motorRFFullSpeed);

  analogWrite(motorLF, motorLHalfSpeed);

  analogWrite(motorLB, motorStop);

  analogWrite(motorRB, motorStop);

}
```

Logbook from date to date

Monday	On Monday I started working on Movies assignment.
Tuesday	I started working on Songs assignment.
Wednesday	I finished Songs assignment.
Thursday	I asked for help for Movies assignment and finished it.
Friday	I started working on Fifty Ville assignment.
Saturday	I kept working on Fifty Ville.
Sunday	I finished the last assignment.

Self-reflection

This week, for the Computer Science class we had to complete three assignments in SQL. Firstly, I started to work on the Movies assignment, so it was pretty hard for me, but Peter, my groupmate helped me and explained to me how it works. After that, SQL became easier.

I completed Songs fast, then I worked on Fiftyville, which was my favorite. We had to solve a mystery by finding clues in different tables. Besides working

with SQL, this assignment also taught me how to use different clues to find out something.

For Embedded Systems we had to make our robot avoid an object. This part was really easy, so we finished it in one day.

Week 3

Computer science

Trivia

Design a webpage using HTML, CSS, and JavaScript to let users answer trivia questions.

<u>This is the code for the assignment:</u>

HTML and JavaScript:

```
<!DOCTYPE html>

<html lang="en">
    <head>
        <link
href="https://fonts.googleapis.com/css2?family=Montserrat:wght@500&display=swap"
rel="stylesheet">
        <link href="styles.css" rel="stylesheet">
        <title>Trivia!</title>
        <script>
            // TODO: Add code to check answers to questions
        </script>
    </head>
    <body>
        <div class="header">
            <h1>Trivia!</h1>
        </div>

        <div class="container">
            <div class="section">
                <h2>Part 1: Multiple Choice </h2>

                <hr>
                <!-- question -->
```

```
                <h3>Which one of the following castles from Romania was named Dracula's
Castle?</h3>
                <button data-correct="false">Peleș Castle</button>
                <button data-correct = "true">Bran Castle</button>
                <button data-correct="false">Corvin Castle</button>
                <button data-correct="false">Banffy Castle</button>
            </div>

            <div class="section">
                <h2>Part 2: Free Response</h2>
                <hr>
                <!-- second question -->
                <h3>What is the most famous food in Romania?</h3>
                <!-- the responses to my question -->
                 <input type="text" id="response">
                 <button id="submit">Submit</button>
            </div>
        </div>

<script>
function displayText(button, message)
{
    var newText = document.createElement("p");
    newText.textContent = message;
    button.parentNode.insertBefore(newText, button.nextSibling);
}

document.querySelectorAll('button').forEach(button =>
{
    button.addEventListener('click', function()
    {
        if (this.getAttribute('data-correct') == 'true')
        {
            this.style.backgroundColor = '#008000'; // Green for correct
            displayText(this, "Correct!");
        }
        else
        {
            this.style.backgroundColor = '#FF0000'; // Red for incorrect
            displayText(this, "Incorrect");
        }
    });
});
```

```
<!-- check if answer is correct or incorrect for the second question-->
document.getElementById('submit').addEventListener('click', function()
{
    var response = document.getElementById('response').value;
    var feedback = document.getElementById('feedback');

    if (response.toLowerCase() === 'sarmale')
    { // Assuming 'sarmale' is the correct answer
        document.getElementById('response').style.backgroundColor = '#008000'; // Green
        feedback.textContent = 'Correct!';
    }
    else
    {
        document.getElementById('response').style.backgroundColor = '#FF0000'; // Red
        feedback.textContent = 'Incorrect';
    }
});
</script>

    </body>
</html>
```

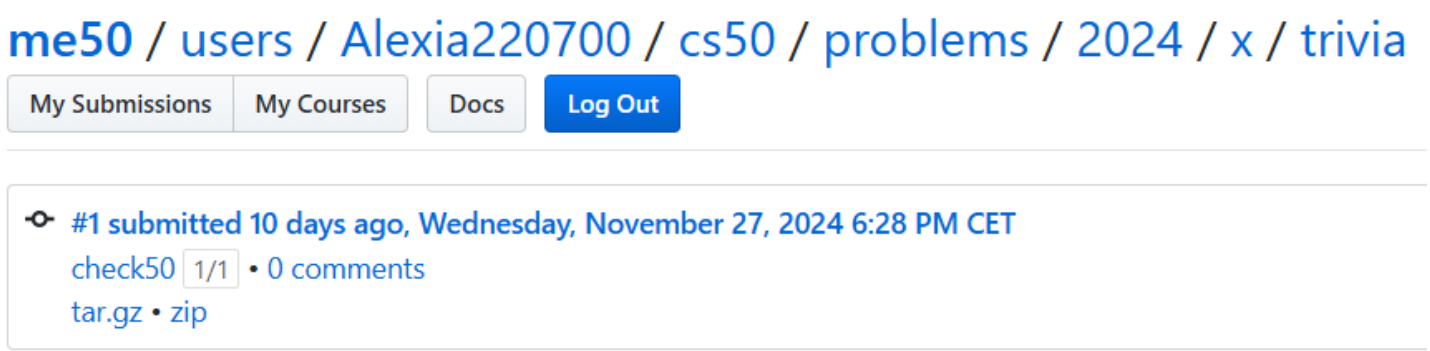
CSS:

```
body
{
    background-color: #fff;
    color: #212529;
    font-size: 1rem;
    font-weight: 400;
    line-height: 1.5;
    margin: 0;
    text-align: left;
}

.container
{
    margin-left: auto;
    margin-right: auto;
    padding-left: 15px;
    padding-right: 15px;
}
```

```
.header
{
    background-color: #477bff;
    color: #fff;
    margin-bottom: 2rem;
    padding: 2rem 1rem;
    text-align: center;
}

.section
{
    padding: 0.5rem 2rem 1rem 2rem;
}

.section:hover
{
    background-color: #f5f5f5;
    transition: color 2s ease-in-out, background-color 0.15s ease-in-out;
}

h1 {
    font-family: 'Montserrat', sans-serif;
    font-size: 48px;
}

button, input[type="submit"]
{
    background-color: #d9edff;
    border: 1px solid transparent;
    border-radius: 0.25rem;
    font-size: 0.95rem;
    font-weight: 400;
    line-height: 1.5;
    padding: 0.375rem 0.75rem;
    text-align: center;
    transition: color 0.15s ease-in-out, background-color 0.15s ease-in-out, border-color
0.15s ease-in-out, box-shadow 0.15s ease-in-out;
    vertical-align: middle;
}

button, input[type="submit"]
{
    background-color: #d9edff;
    border: 1px solid transparent;
```

```
    border-radius: 0.25rem;
    font-size: 0.95rem;
    font-weight: 400;
    line-height: 1.5;
    padding: 0.375rem 0.75rem;
    text-align: center;
    transition: color 0.15s ease-in-out, background-color 0.15s ease-in-out, border-color
0.15s ease-in-out, box-shadow 0.15s ease-in-out;
    vertical-align: middle;
}

input[type="text"]
{
    line-height: 1.8;
    width: 25%;
}

input[type="text"]:hover
{
    background-color: #f5f5f5;
    transition: color 2s ease-in-out, background-color 0.15s ease-in-out;
}
```

The Harvard score with the title of the program and my name:



https://submit.cs50.io/check50/be68377d83aa9fe41473580a14f476a17fcd6e10

Homepage

Build a simple homepage using HTML, CSS, and JavaScript, that introduces yourself, your favorite hobby or extracurricular, or anything else of interest to you.

- HTML, or HyperText Markup Language, which is used to describe the content of websites;
- CSS, Cascading Style Sheets, which is used to describe the aesthetics of websites; and
- JavaScript, which is used to make websites interactive and dynamic.

These are the HTML codes for the assignment:

This part shows the code for the Bootstrap:

```
<link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
integrity="sha384-TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2"
crossorigin="anonymous">
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
crossorigin="anonymous"></script>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-ho+j7jyWK8fNQe+A12Hb8AhRq26LrZ/JpcUGGOn+Y7RsweNrtN/tE3MoK7ZeZDyx"
crossorigin="anonymous"></script>
```

Link the CSS file to the HTML file:

```
<head>
    <link href="contactform.css" rel="stylesheet">
</head>
```

Make the page fit different types of screens:

```
<head>
```

```
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
```

Create the menu bar:

```
<!--create menu bar-->
        <!--<nav> tag in HTML is used to define a block of navigation links-->
        <nav>
            <!--make an unsorted list using <ul>-->
            <ul>
              <!-- <li> tag is used to define a list item within either an unordered list (<ul>) or an ordered list
(<ol>).-->
              <li><a href="index.html">Home</a></li>
              <li><a href="learnaboutme.html">About my website</a></li>
              <li><a href="myprojects.html">Horror genre</a></li>
              <li><a href="contactform.html">Contact</a></li>
              <li><a href="ratemysite.html">Reviews</a></li>
              <li><a href="useditems.html">Used items</a></li>
            </ul>
          </nav>
```

Create the footer:

```
<footer>© December 2024</footer>
```

Add fade to pictures when scrolling:

```
<script>
    <!-- adds an event listener to the document that listens for the "scroll" event-->
    document.addEventListener("scroll", function()
{
    <!-- selects all elements with the class fade-in and stores them in the elements variable-->
    const elements = document.querySelectorAll('.fade-in');
    <!-- loops through each element in the elements NodeList-->
    elements.forEach(element =>
        {
        <!-- gets the position of the current element relative to the viewport-->
        const rect = element.getBoundingClientRect();
        <!-- checks if the element is within the viewport-->
        if (rect.top < window.innerHeight && rect.bottom > 0)
        {
```

```
            <!-- if the element is visible, this line adds the visible class to the element-->
            element.classList.add('visible');
        }
    });
});
</script>
```

Ask for name input:

```
<h2>· Input your name:</h2>
    <!--make the field for user to input the information-->
    <input type="text" id="name" name="name">
```

Ask for email input:

```
<h3>· Input your email address ✉:</h3>
    <input type="text" id="email" name="email">
```

Ask for a message input:

```
<h4>· Ask me a question or send a message here:</h4>
    <input type="text" id="message" name="message">
```

Make an interactive button:

```
<!--make an interactive button-->
    <!--it alerts if the form is correct and submitted or not-->
    <form id="contactForm">
        <button type="button" onclick="validateForm()">Click here!</button>
    </form>
    <script>
        function validateForm()
         {
            <!--checks if all the fields are completed-->
            const name = document.getElementById('name').value;
            const email = document.getElementById('email').value;
            const message = document.getElementById('message').value;

            if (name && email && message)
             {
```

```
                alert("Form submitted successfully!");
            }
            else
            {
                alert("Please fill out all fields.");
            }
        }

    </script>
```

Add hover to pictures:

```
<div>
<img class="hover-image image1" src="images/The Shining_ Halloween edition - King, Stephen_
9781473695498.jpg">
<img class="hover-image image2" src="images/Carrie.jpg">
<img class="hover-image image3" src="images/Read It online free by Stephen King.jpg">
<img class="hover-image image4" src="images/Dracula's Guest.jpg">
<img class="hover-image image5" src="images/The Complete Tales and Poems of Edgar Allan
Poe.jpg">
<img class="hover-image image6" src="images/_Thrillers don't come any better than
this__….jpg">
</div>
```

Style menu bar (CSS):

```
/*create menu bar background*/
nav ul
{
    display:flex;
    /* Disperse items */
    justify-content: space-between;
    list-style-type: none;
    margin: 0;
    padding: 0;
    overflow: hidden;
    background-color: #0f0005;
}

/*align links to left and horizontally*/
nav ul li
{
```

```
    float: left;
}



/*made them look like buttons*/
nav ul li a {
    display: block;
    background-color: #0f0005;
    color: white;
    text-align: center;
    margin: 0;
    padding: 16px;
    text-decoration: none;
}

nav ul li a:hover
{
    background-color: #343434;
}
```

Style footer:

```
footer
{
    background-color: #0f0005;
    color: white;
    text-align: center;
    padding: 20px;
    position: static;
    bottom: 0;
    width: 100%;
    margin-top: 3px;
}
```

Style input field:

```
input
{
    font-size: 17px;
    height: 60px;
    /*using percentage so the input field changes when on full or small screen type*/
```

```
    width: 90%;
    margin-left: 40px;
}
```

Style button:

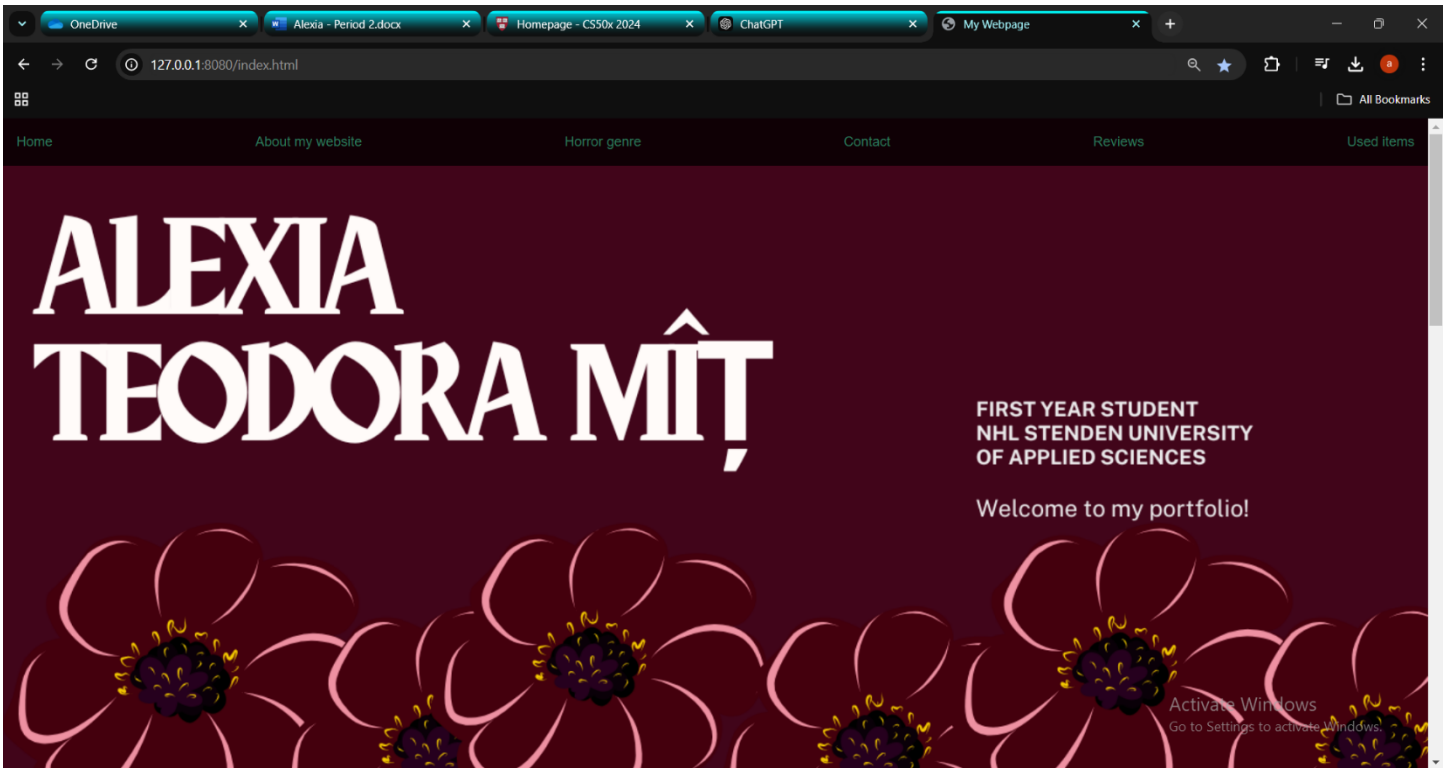
```
/*style button*/
body button
{
    background-color: #ee3b47;
    border: none;
    border-radius: 20px;
    height: 60px;
    width: 89px;
    /*displaying the button in the middle of the page*/
    text-align: center;
    color: white;
    font-size: 15px;
    margin-bottom: 60px;
}

/*place button*/
.container1
{
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
}
```

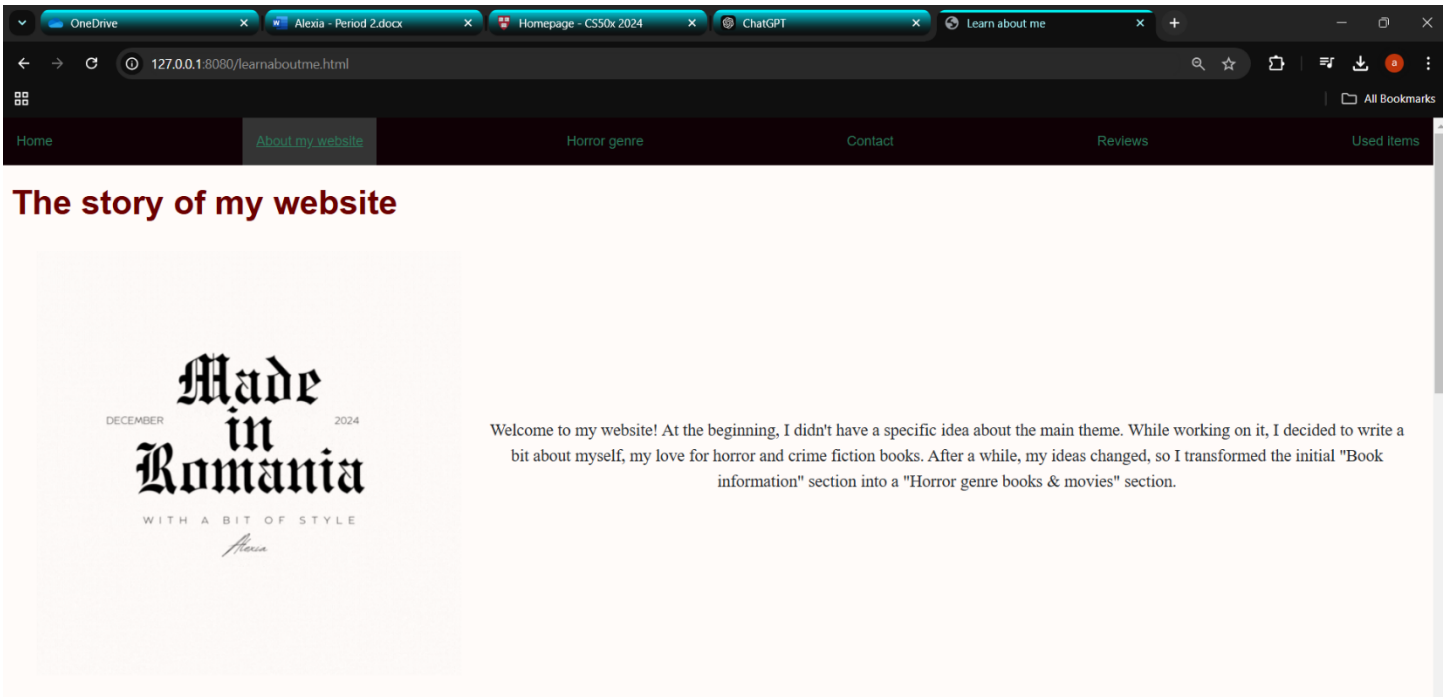
Style links and add hover:

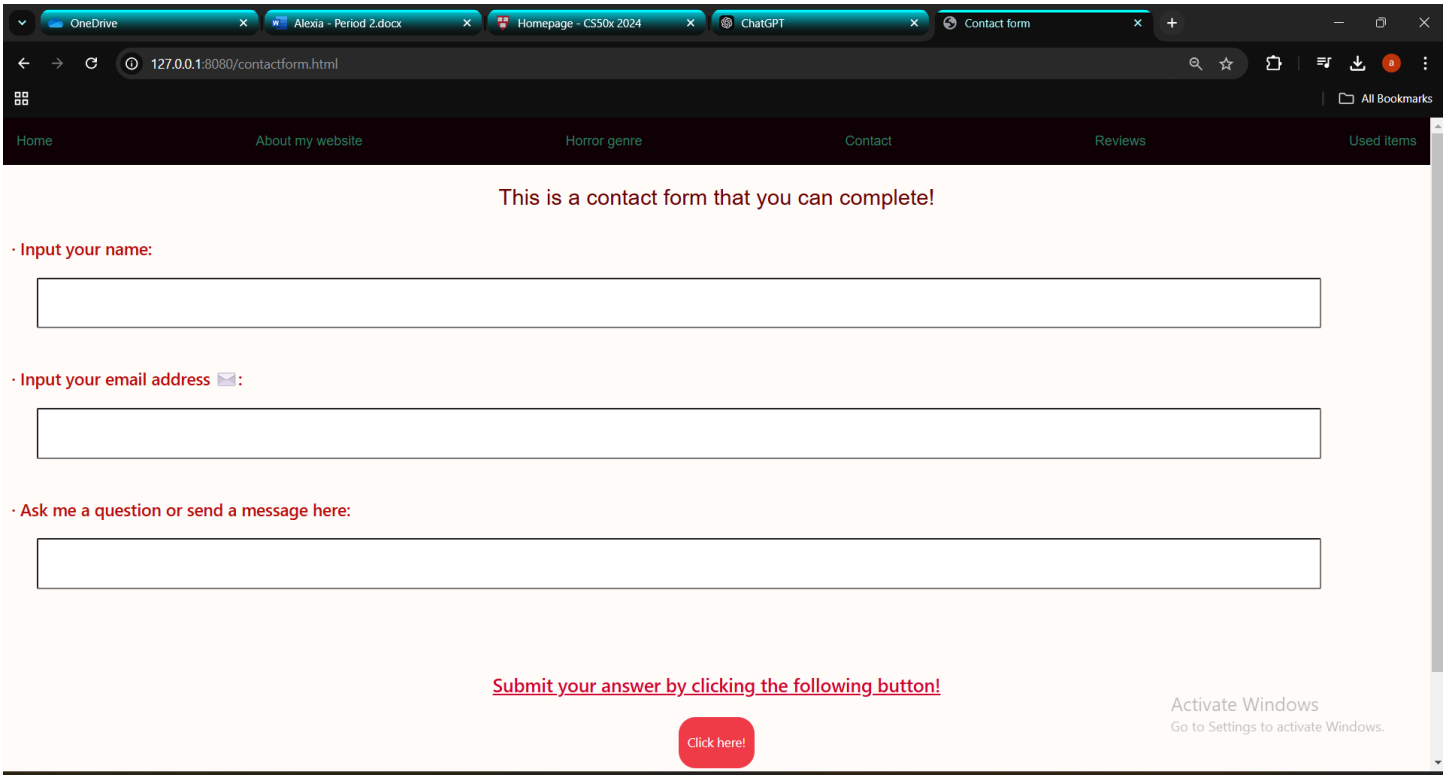
```
/*for the links*/
a
{
    /*force the color change*/
    color: #2e7e5f !important;
    font-size: 16px;
    /*inline-block used to fit every type of text width for bgcolor*/
    display: inline-block;
```

```
    margin: 0;
    font-family: Arial;
    background-color: #fce5e5;
    /* the text will continue on the same line until a line break (<br>) or the end of the text is reached */
    white-space: nowrap;
    text-decoration: underline;
}

a:hover
{
    color: #fce5e5;
    background-color: #C41A0B;
    transition: transform 0.1s;
}
```

Here are some pictures of my Homepage:






OneDrive
Alexia - Period 2.docx
Homepage - CS50x 2024
ChatGPT
Contact form
127.0.0.1:8080/contactform.html
All Bookmarks
Home
About my website
Horror genre
Contact
Reviews
Used items
This is a contact form that you can complete!
· Input your name:
· Input your email address :
· Ask me a question or send a message here:
Submit your answer by clicking the following button!
Click here!
Activate Windows
Go to Settings to activate Windows.

Embedded Systems

Stay on track

```
// Motor Pins

#define MOTOR_LEFT_FWD 10

#define MOTOR_LEFT_BWD 11

#define MOTOR_RIGHT_FWD 9

#define MOTOR_RIGHT_BWD 3


// Sensor Pins

const unsigned char TRACK_SENSORS[] = {A7, A6, A5, A4, A3, A2, A1, A0};  // 8 sensors

int sensorReadings[8] = {0, 0, 0, 0, 0, 0, 0, 0};


// Threshold for detecting black

const unsigned int thresholdBlack =


800;  // Adjust based on sensor calibration


// Motor Speeds

#define SPEED_LEFT 200

#define SPEED_RIGHT 220

#define TURN_SPEED_LEFT 55
```

```
#define TURN_SPEED_RIGHT 60

// Variable to remember the last direction the robot lost the line
bool lastDirectionWasLeft = false;

void setup() {
  // Set motor pins as outputs
  pinMode(MOTOR_LEFT_FWD, OUTPUT);
  pinMode(MOTOR_LEFT_BWD, OUTPUT);
  pinMode(MOTOR_RIGHT_FWD, OUTPUT);
  pinMode(MOTOR_RIGHT_BWD, OUTPUT);

  // Start serial communication for debugging
  Serial.begin(9600);
}

void loop() {
  // Read sensor values
  for (int i = 0; i < 8; i++) {
    sensorReadings[i] = analogRead(TRACK_SENSORS[i]);  // Read analog values from the sensors
  }

  // Check the position of the line
```

```
int leftCount = 0;
int rightCount = 0;

for (int i = 0; i < 4; i++) {  // Check left side sensors (0-3)
  if (sensorReadings[i] > thresholdBlack) {
    leftCount++;
  }
}

for (int i = 4; i < 8; i++) {  // Check right side sensors (4-7)
  if (sensorReadings[i] > thresholdBlack) {
    rightCount++;
  }
}

// If both left and right sides detect the line, move forward
if (leftCount > 0 && rightCount > 0) {
  moveForward();
}
// If left side detects the line more, turn left
else if (leftCount > 0) {
  turnLeft();
  lastDirectionWasLeft = true;  // Remember the last direction
```

```
  }
  // If right side detects the line more, turn right
  else if (rightCount > 0) {
    turnRight();
    lastDirectionWasLeft = false;  // Remember the last direction
  }
  // If no sensors detect the line, continue turning in the last known direction
  else {
    if (lastDirectionWasLeft) {
      turnLeft();  // Continue turning left
    } else {
      turnRight();  // Continue turning right
    }
  }

  delay (50); // Short delay before the next sensor read
}

// Function to move both motors forward
void moveForward() {
  analogWrite(MOTOR_LEFT_FWD, SPEED_LEFT);
  analogWrite(MOTOR_RIGHT_FWD, SPEED_RIGHT);
```

```
}

// Function to turn left by adjusting motor speeds
void turnLeft() {
  // Stop left motor
 analogWrite(MOTOR_RIGHT_FWD, TURN_SPEED_RIGHT);  // Speed up right motor

}

// Function to turn right by adjusting motor speeds
void turnRight() {
 analogWrite(MOTOR_LEFT_FWD, TURN_SPEED_LEFT);  // Speed up left motor

}
```

Logbook from day to day

Monday	I made Trivia.
Tuesday	I started working on the Homepage.
Wednesday	I worked on the base of the website, made most of the HTML code.
Thursday	I continued to work on the HTML and started designing with CSS language.
Friday	I kept working on the design. Besides, I helped my team to make the Technical Design for the RelayBot.
Saturday	I also used Canva for making some posters and the logo for my website.
Sunday	I finished the website.

Self-reflection

This week was my favorite one, because I could use my creativity while programming. For Computer Science, we had to create a small Trivia Quiz, which was easy. Besides, we had to make a Homepage about a subject we preferred. I created it using HTML for the base, CSS for styling and JavaScript for making it interactive. It had pages about me, one of my hobbies, reviews for the website and a page where people could send questions or messages. I added hover to some of the pictures, an interactive button, a menu bar and a footer etc.

I also worked with my teammates on the technical design for the Relaybot.

For Embedded Systems we had to make the Relaybot stay on track. This was much harder than the other assignments, so it took us longer to make it work as it should. In the end, we asked for help from one of our teammates and another colleague that was more experienced.

Week 4

Computer Science

Birthdays

Complete the implementation of a web application to let users store and keep track of birthdays.

- When the / route is requested via GET, your web application should display, in a table, all of the people in your database along with their birthdays.
- First, in app.py, add logic in your GET request handling to query the birthdays.db database for all birthdays. Pass all of that data to your index.html template.
- Then, in index.html, add logic to render each birthday as a row in the table. Each row should have two columns: one column for the person's name and another column for the person's birthday.
- When the / route is requested via POST, your web application should add a new birthday to your database and then re-render the index page.
- First, in index.html, add an HTML form. The form should let users type in a name, a birthday month, and a birthday day. Be sure the form submits to / (its "action") with a method of post.
- Then, in app.py, add logic in your POST request handling to INSERT a new row into the birthdays table based on the data supplied by the user.

<u>This is the Python code for implementing Flask:</u>

```
@app.route("/", methods=["GET", "POST"])
def index():
    if request.method == "POST":

        # Add the user's entry into the database
        # collect form data
        name = request.form.get('name')
        month = request.form.get('month')
```

```
        day = request.form.get('day')

        # insert form data into database
        db.execute("INSERT INTO birthdays (name, month, day) VALUES (?, ?, ?)", name, month,
day)

        # render the template and pass the retrieved data to it
        return redirect("/")

    else:
        # retrieve all entries from the database
        rows = db.execute("SELECT * FROM birthdays")

        # Display the entries in the database on index.html
        return render_template("index.html", birthdays=rows)
```

This is the HTML code for creating the form with inputs:

```
<h2>Add a Birthday</h2>
                <!-- Create a form for users to submit a name, a month, and a day -->
                <!--make the field for user to input the information-->
                <form action="/" method="post">
                <!--placeholder used for showing words in the input fields-->
                <input type="text" id="name" name="name" placeholder="Name">
                <input type="number" id="month" name="month" placeholder="Month">
                <input type="number" id="day" name="day" placeholder="Day">
                <button type="submit">Add Birthday</button>
                </form>
```

This is the HTML code for looping through the database and display the table with columns and rows:

```
<div class="section">

                <h2>All Birthdays</h2>
                <table>
                    <thead>
                        <tr>
                            <th>Name</th>
                            <th>Birthday</th>
```

```
                        </tr>
                    </thead>
                    <tbody>
                                <!-- Loop through the database entries to display them in this
table -->
                                {% for birthday in birthdays %}
                                <tr>
                                    <!-- Print the name in the first cell -->
                                    <td>{{ birthday.name }}</td>
                                    <!-- Print the birthday (month/day) in the second cell --
>
                                    <td>{{ birthday.month }}/{{ birthday.day }}</td>
                                </tr>
                                {% endfor %}
                            </tbody>
                        </table>
                    </tbody>
                </table>
            </div>
        </div>
```

This is the CSS code for styling the form with inputs:

```
input
{
    text-align: left;
    margin-left: 5px;
}
```

The Harvard score with the title of the program and my name:

me50 / users / Alexia220700 / cs50 / problems / 2024 / x / birthdays

My Submissions | My Courses | Docs | Log Out

#1 submitted 4 minutes ago, Monday, December 9, 2024 9:44 PM CET

check50 1/1 • style50 0.97 • 0 comments

tar.gz • zip

https://submit.cs50.io/check50/0a1d3af96879fbcd2ac15a6c352192023a2f73a3

Finance

Implement a website via which users can “buy” and “sell” stocks.

Complete the implementation of register in such a way that it allows a user to register for an account via a form.

Complete the implementation of quote in such a way that it allows a user to look up a stock’s current price.

Complete the implementation of buy in such a way that it enables a user to buy stocks.

Complete the implementation of index in such a way that it displays an HTML table summarizing, for the user currently logged in, which stocks the user owns, the numbers of shares owned, the current price of each stock, and the total value of each holding (i.e., shares times price). Also display the user’s current cash balance along with a grand total (i.e., stocks’ total value plus cash).

Complete the implementation of sell in such a way that it enables a user to sell shares of a stock (that he or she owns).

Complete the implementation of history in such a way that it displays an HTML table summarizing all of a user’s transactions ever, listing row by row each and every buy and every sell.

app.py:

```
import os

from cs50 import SQL
```

```python
from flask import Flask, flash, redirect, render_template, request, session
from flask_session import Session
from werkzeug.security import check_password_hash, generate_password_hash

from helpers import apology, login_required, lookup, usd

# Configure application
app = Flask(__name__)

# Custom filter
app.jinja_env.filters["usd"] = usd

# Configure session to use filesystem (instead of signed cookies)
app.config["SESSION_PERMANENT"] = False
app.config["SESSION_TYPE"] = "filesystem"
Session(app)

# Configure CS50 Library to use SQLite database
db = SQL("sqlite:///finance.db")


@app.after_request
def after_request(response):
    """Ensure responses aren't cached"""
    response.headers["Cache-Control"] = "no-cache, no-store, must-revalidate"
    response.headers["Expires"] = 0
    response.headers["Pragma"] = "no-cache"
    return response


@app.route("/")
@login_required
def index():
    """Show portfolio of stocks"""
    user_id = session["user_id"]

    # group by and sum to add all the newly added data together
    stocks = db.execute(
        "SELECT symbol, name, price, SUM(shares) as totalShares FROM transactions WHERE
user_id = ? GROUP BY symbol", user_id)

    # removes stocks with zero shares
    stocks = [stock for stock in stocks if stock["totalShares"] > 0]
```

```
    # [0]: Assumes the query returns a list of rows (even if only one). [0] accesses the
first row.
    # ["cash"]: Extracts the value of the cash column from that row.
    cash = db.execute("SELECT cash FROM users WHERE id = ?", user_id)[0]["cash"]

    total = cash
    for stock in stocks:
        total += stock["price"] * stock["totalShares"]

    return render_template("index.html", stocks=stocks, cash=cash, total=total, usd=usd)


@app.route("/buy", methods=["GET", "POST"])
@login_required
def buy():
    """Buy shares of stock"""
    if request.method == "POST":
        symbol = request.form.get("symbol").upper()
        item = lookup(symbol)

        if not symbol:
            return apology("Symbol missing!")
        elif not item:
            return apology("Invalid Symbol!")

        try:
            shares = int(request.form.get("shares"))
        except:
            return apology("Shares must be an integer!")

        if shares <= 0:
            return apology("Shares must be positive")

        user_id = session["user_id"]
        cash = db.execute("SELECT cash FROM users WHERE id = ?", user_id)[0]["cash"]

        item_name = item["name"]
        item_price = item["price"]
        total_price = item_price * shares

        if cash < total_price:
            return apology("Insufficient cash")
```

```
        else:
            db.execute("UPDATE users SET cash = ? WHERE id = ?", cash - total_price, user_id)
            db.execute("INSERT INTO transactions (user_id, name, shares, price, type, symbol)
VALUES(?, ?, ?, ?, ?, ?)",
                       user_id, item_name, shares, item_price, 'buy', symbol)
            return redirect("/")
    else:
        user_id = session["user_id"]
        cash = db.execute("SELECT cash FROM users WHERE id = ?", user_id)[0]["cash"]
        return render_template("buy.html", cash=cash)



@app.route("/history")
@login_required
def history():
    """Show history of transactions"""
    user_id = session["user_id"]
    transactions = db.execute(
        "SELECT type, symbol, price, shares, time FROM transactions WHERE user_id = ?",
user_id)

    return render_template("history.html", transactions=transactions, usd=usd)



@app.route("/login", methods=["GET", "POST"])
def login():
    """Log user in"""

    # Forget any user_id
    session.clear()

    # User reached route via POST (as by submitting a form via POST)
    if request.method == "POST":
        # Ensure username was submitted
        if not request.form.get("username"):
            return apology("must provide username", 403)

        # Ensure password was submitted
        elif not request.form.get("password"):
            return apology("must provide password", 403)

        # Query database for username
        rows = db.execute(
```

```
            "SELECT * FROM users WHERE username = ?", request.form.get("username")
        )

        # Ensure username exists and password is correct
        if len(rows) != 1 or not check_password_hash(
            rows[0]["hash"], request.form.get("password")
        ):
            return apology("invalid username and/or password", 403)

        # Remember which user has logged in
        session["user_id"] = rows[0]["id"]

        # Redirect user to home page
        return redirect("/")

    # User reached route via GET (as by clicking a link or via redirect)
    else:
        return render_template("login.html")


@app.route("/logout")
def logout():
    """Log user out"""

    # Forget any user_id
    session.clear()

    # Redirect user to login form
    return redirect("/")


@app.route("/quote", methods=["GET", "POST"])
@login_required
def quote():
    """Get stock quote."""
    if request.method == "POST":
        symbol = request.form.get("symbol")

        if not symbol:
            return apology("Symbol missing")

        item = lookup(symbol)
```

```
        if not item:
            return apology("Invalid Symbol")

        return render_template("quoted.html", item=item, usd=usd)
    else:
        return render_template("quote.html")


@app.route("/register", methods=["GET", "POST"])
def register():
    """Register user"""
    if request.method == "GET":
        return render_template("register.html")
    else:
        username = request.form.get("username")
        password = request.form.get("password")
        confirmation = request.form.get("confirmation")

        if not username:
            return apology("Username missing")

        if not password:
            return apology("Password missing")

        if not confirmation:
            return apology("Must Give Confirmation")

        if password != confirmation:
            return apology("Passwords Do Not Match")

        hash = generate_password_hash(password)

        try:
            db.execute("INSERT INTO users (username, hash) VALUES (?, ?)", username, hash)
        except:
            return apology("Username already exists")

        return redirect("/")


@app.route("/sell", methods=["GET", "POST"])
```

```
@login_required
def sell():
    """Sell shares of stock"""
    user_id = session["user_id"]

    if request.method == "POST":
        symbol = request.form.get("symbol")
        shares = int(request.form.get("shares"))

        if shares <= 0:
            return apology("Shares must be a positive!")

        item_price = lookup(symbol)["price"]
        item_name = lookup(symbol)["name"]
        price = shares * item_price

        rows = db.execute(
            "SELECT SUM(shares) AS total_shares FROM transactions WHERE user_id = ? AND
symbol = ? GROUP BY symbol", user_id, symbol)
        if not rows or rows[0]["total_shares"] is None or rows[0]["total_shares"] < shares:
            return apology("Not enough shares!")
        shares_owned = rows[0]["total_shares"]

        if shares_owned < shares:
            return apology("Not enough shares!")

        current_cash = db.execute("SELECT cash FROM users WHERE id = ?", user_id)[0]["cash"]
        db.execute("UPDATE users SET cash = ? WHERE id = ?", current_cash + price, user_id)
        db.execute("INSERT INTO transactions (user_id, name, shares, price, type, symbol)
VALUES (?, ?, ?, ?, ?, ?)",
                   user_id, item_name, -shares, item_price, "sell", symbol)

        return redirect("/")
    else:
        symbols = db.execute(
            "SELECT symbol FROM transactions WHERE user_id = ? GROUP BY symbol", user_id)
        return render_template("sell.html", symbols=symbols)


@app.route("/money", methods=["GET", "POST"])
@login_required
def money():
    """add money on homepage"""
```

```
    user_id = session["user_id"]

    if request.method == "POST":
        try:
            extra_cash = int(request.form.get("money"))
        except ValueError:
            return apology("Invalid input")

        if extra_cash <= 0:
            return apology("Enter a positive number")

        current_cash = db.execute("SELECT cash FROM users WHERE id = ?", user_id)[0]["cash"]
        db.execute("UPDATE users SET cash = ? WHERE id = ?", current_cash + extra_cash,
user_id)
        return redirect("/")
    else:
        return redirect("/")
```

register.html:

```
{% extends "layout.html" %}

{% block title %}
    Register
{% endblock %}

{% block main %}
    <form action="/register" method="post">
        <div class="mb-3">
            <input autocomplete="off" autofocus class="form-control mx-auto w-auto"
name="username" placeholder="Username" type="text">
        </div>

        <div class="mb-3">
            <input class="form-control mx-auto w-auto" name="password" placeholder="Password"
type="password">
        </div>

        <div class="mb-3">
            <input class="form-control mx-auto w-auto" name="confirmation"
placeholder="Confirm Password" type="password">
```

```
        </div>

        <button class="btn btn-primary" type="submit">Register</button>
    </form>
{% endblock %}
```

quote.html:

```
{% extends "layout.html" %}

{% block title %}
    Quote
{% endblock %}

{% block main %}
    <form action="/quote" method="post">
        <div class="mb-3">
            <input autocomplete="off" autofocus class="form-control mx-auto w-auto"
name="symbol" placeholder="Symbol" type="text">
        </div>

        <button class="btn btn-primary" type="submit">Quote</button>
    </form>
{% endblock %}
```

buy.html:

```
{% extends "layout.html" %}

{% block title %}
    BUY
{% endblock %}

{% block main %}
    <form action="/buy" method="post">
        <div class="mb-3">
            <input autocomplete="off" autofocus class="form-control mx-auto w-auto"
name="symbol" placeholder="Symbol" type="text">
        </div>
        <div class="mb-3">
```

```
            <input class="form-control mx-auto w-auto" name="shares" placeholder="Shares"
type="text">
        </div>
        <button class="btn btn-primary" type="submit">BUY</button>
    </form>
    <p>Cash: {{ cash }}</p>
{% endblock %}
```

index.html:

```
{% extends "layout.html" %}

{% block title %}
    HomePage
{% endblock %}

{% block main %}
<table class="table table-striped">
    <thead>
        <tr>
            <th>Symbol</th>
            <th>Name</th>
            <th>Shares</th>
            <th>Price</th>
            <th>TOTAL</th>
        </tr>
    </thead>
    <tbody>
        {% for stock in stocks %}
        <tr>
            <td>{{ stock["symbol"] }}</td>
            <td>{{ stock["name"] }}</td>
            <td>{{ stock["totalShares"] }}</td>
            <td>{{ usd(stock["price"]) }}</td>
            <td>{{ usd(stock["totalShares"] * stock["price"]) }}</td>
        </tr>
        {% endfor %}
        <tr>
            <td>CASH</td>
            <td colspan = "3"></td>
            <td>{{ usd(cash) }}</td>
        </tr>
```

```
    </tbody>

    <tfoot>
        <td colspan = "4"></td>
        <td><strong>{{ usd(total) }}</strong></td>
    </tfoot>
</table>

<form action = "/money" method = "post">
    <div class="mb-3">
        <input class="form-control mx-auto w-auto" name="money" placeholder="Extra Cash"
type="number">
    </div>
    <button class="btn btn-primary" type="submit">Cash in</button>
</form>
{% endblock %}
```

sell.html:

```
{% extends "layout.html" %}

{% block title %}
    SELL
{% endblock %}

{% block main %}
    <form action="/sell" method="post">
        <div class="mb-3">
            <select name = "symbol">
                {% for symbol in symbols %}
                <option>{{ symbol["symbol"] }}</option>
                {% endfor %}
            </select>
        </div>
        <div class="mb-3">
            <input class="form-control mx-auto w-auto" name="shares" placeholder="Shares"
type="number">
        </div>
        <button class="btn btn-primary" type="submit">Sell</button>
    </form>
{% endblock %}
```

history.html:

```
{% extends "layout.html" %}

{% block title %}
    History
{% endblock %}

{% block main %}
<table class="table table-striped">
    <thead>
        <tr>
            <th>Symbol</th>
            <th>type</th>
            <th>Shares</th>
            <th>Price</th>
            <th>Time</th>
        </tr>
    </thead>
    <tbody>
        {% for transaction in transactions %}
        <tr>
            <td>{{ transaction["symbol"].upper() }}</td>
            <td>{{ transaction["type"] }}</td>
            <td>{{ transaction["shares"] }}</td>
            <td>{{ usd(transaction["price"]) }}</td>
            <td>{{ transaction["time"] }}</td>
        </tr>
        {% endfor %}
    </tbody>
</table>
{% endblock %}
```

Logbook from day to day

Monday	I started working on Birthdays.
Tuesday	I continued to work on Birthdays.
Wednesday	I started making Finance.
Thursday	I kept trying to make Finance.
Friday	I got different errors for Finance.
Saturday	I couldn’t fix it.
Sunday	I struggled with the assignment, so I gave up and made it later.

Week 5

Computer Science

SQL Murder Mystery

A crime has taken place, and the detective needs your help. The detective gave you the crime scene report, but you somehow lost it. You vaguely remember that the crime was a murder that occurred sometime on **Jan.15, 2018 and** that it took place in **SQL City.** Start by retrieving the corresponding crime scene report from the police department's database.

<u>This is the code for the assignment:</u>

```
/*find the witnesses: Morty Schapiro, Torie Thalman*/
SELECT *
FROM person
JOIN interview ON interview.person_id = person.id
WHERE address_street_name = 'Northwestern Dr'
ORDER BY address_number DESC LIMIT 2;

/*show the transcripts*/
SELECT person.name, interview.transcript
FROM person
JOIN interview
ON person.id = interview.person_id
WHERE person.id = 14887 OR person.id = 16371;

/*get description of crime*/
SELECT crime_scene_report.description
FROM crime_scene_report
WHERE crime_scene_report.city = "SQL City";

/*find the people w the most income*/
SELECT person.name, income.annual_income
FROM income
JOIN person
ON income.ssn = person.ssn
WHERE annual_income > 450000;
```

```sql
/*find the name of the person w that license plate number*/
SELECT person.name, drivers_license.plate_number
FROM person
JOIN drivers_license ON person.license_id = drivers_license.id
WHERE drivers_license.plate_number LIKE "%H42W%";
/*Jeremy Bowers, Maxine Whitely, Tushar Chandra*/

/*find the name of the person w that gym membership*/
SELECT DISTINCT get_fit_now_member.name, get_fit_now_member.membership_status,
get_fit_now_check_in.check_in_date
FROM get_fit_now_member
JOIN get_fit_now_check_in ON get_fit_now_check_in.membership_id = get_fit_now_member.id
WHERE get_fit_now_check_in.check_in_date = "20180109";
```

The is the check for the name of the criminal:

Check your solution

Did you find the killer?

```sql
INSERT INTO solution VALUES (1, 'Jeremy Bowers');

        SELECT value FROM solution;
```

RUN ⇩ RESET

value
Congrats, you found the murderer! But wait, there's more... If you think you're up for a challenge, try querying the interview transcript of the murderer to find the real villain behind this crime. If you feel especially confident in your SQL skills, try to complete this final step with no more than 2 queries. Use this same INSERT statement with your new suspect to check your answer.

The is a part of my notes:

```
mistery
murder that occurred sometime on Jan.15, 2018
it took place in SQL City

crime scene description from report:
Security footage shows that there were 2 witnesses. The first witness lives at the last house
on "Northwestern Dr". The second witness, named Annabel, lives somewhere on "Franklin Ave".

witnesses: Morty Schapiro, Annabel Miller

Morty: heared a gunshot and saw a man running, "Get Fit Now Gym" bag
The membership number on the bag started with "48Z" => gold member
car with a plate that included "H42W"

Annabel Miller:  working out last week on January the 9th, saw him there

//check membership, license plate of car, most income
the names of the persons w gold membership and check in on 9 January:
Jeremy Bowers, Sarita Bartosh, Burton Grippe, Carmen Dimick, Joe Germuska, Annabel Miller

the names of the persons w that license plate number:
Jeremy Bowers, Maxine Whitely, Tushar Chandra


Jeremy Bowers is on both lists and is also a man, as in the transcript => he is the criminal
```

Week 6

Computer Science

Teamsite

For this assignment, we had to create a website for our Relaybots. I decided to work on this since I enjoyed making the Homepage for myself. I created different pages on the website, the first one showing the Battle bots, a small introduction for each etc.

We also needed to add a Dashboard, but I got help from Lennon, my teammate.

This is a part of the code for the assignment

Creating the Menu bar:

```
<nav>
          <ul>
            <li><a href="home-page.html">Home</a></li>
            <li><a href="specification.html">Specification</a></li>
            <li><a href="contact.html">Contact</a></li>
            <li><a href="reviews.html">Reviews</a></li>
            <li><a href="ourteam.html">Our team</a></li>
          </ul>
        </nav>
```

Adding an image:

```
<div class="poster">
            <img src="images/Untitled design (5).gif">
          </div>
```

Aligning the pictures and specifications for each robot as a column:

```
<h2>Meet our Relaybots:</h2>
```

```
        <div class="columnForRobots">

          <div class="robot">
              <img src="images/WhatsApp_Image_2024-12-06_at_09.32.14-removebg-preview.png">
              <h3>King Julien</h3>
              <p>Designed to follow a line and evade obstacles with agility.</p>
          </div>

          <div class="robot">
              <img src="images/Untitled design.png">
              <h4>MotoMoto</h4>
              <p>Dedicated to solving a line maze with precision.</p>
          </div>

          <div class="robot">
              <img src="images/WhatsApp_Image_2024-12-05_at_20.58.42-removebg-preview.png">
              <h5>Mort</h5>
              <p>Programmed to finish a physical maze with efficiency.</p>
          </div>

      </div>
```

Creating a footer:

```
<footer>© December 2024</footer>
```

Editing the columns by using CSS:

```
.columnForRobots
{
    display: grid;
    /*creates 3 columns, 1 fraction of the available space*/
    grid-template-columns: repeat(3, 1fr);
    /*creates 2 rows, 1 fraction of the available space*/
    grid-template-rows: repeat(3, 1fr);
    gap: 10px;
    /*changes the items position*/
    justify-items: center;
    margin-top: 20px;
    margin-bottom:0px;
}
```

Logbook from day to day

Monday	I started working on the Teamsite.
Tuesday	I made the Home page, the menu bar and footer, using HTML.
Wednesday	I made the other pages we needed in VSCode.
Thursday	I started looking for matching colors for our website.
Friday	I started designing the website using CSS.
Saturday	I tried to understand how the Dashboard should look like.
Sunday	I asked for help from my teammates to create the Dashboard and they fixed it.

Self-reflection

Week 6 was a bit of a challenge, because of the Data Storage assignment for the Relay Bot, but luckily two of our teammates, Lennon and Peter, did it last year, so that gave us an advantage.

For Computer Science we had to create a Teamsite to present our robots. I offered to create that since I liked making my own page before. I asked my teammates if they agree with the theme of the website, the pictures I can use, the colors. When it was ready in VSCode, I uploaded everything to GitHub.

Professional Skills

Exercise in constructive feedback

Exercise in constructive feedback
(Hoekstra et. Al., 2018)

Situation: A group member puts in less work in the assignment than agreed upon.	
Step 1: Describe objectively observable behavior Feedback is always about behavior and not about the person. With the help of feedback, the recipient can change something about his behavior. Feedback first describes concrete observable behavior: what do you see ..., do you hear...? Example: When you entered, you left the door open.	I noticed that you did not contribute equally to the assignment, as some parts were missing from your section when we reviewed the work together.
Step 2: Describe the effect Feedback is also about the effect that something has on you. It describes observable behavior and its effect. Feedback is meant to be learned from. Therefore, be specific and concrete. Indicate your feelings, say what it does to you. An I message helps with this. Example: When you entered, you left the door open. I found that annoying, because I got cold.	I noticed that some parts were missing from your section when we reviewed the work together. I found this frustrating because it added extra pressure on the rest of the team to complete the missing sections.
Step 3: Describe desired behavior, give a tip or suggestion Someone learns from feedback when he also gets information about a potentially successful (ler) continuation in the learning process. It helps to describe which behavior is very successful and what must therefore be preserved or to describe what is desired in the future. Example: When you entered, you left the door open. I found that annoying, because I got cold. Next time you can help me by closing the door behind you / It's nice to close the door behind you next time.	Next time, it would be helpful if you could communicate any difficulties you are facing earlier so we can support you or redistribute tasks if necessary.
Name: Alexia Mit	

Debriefing

During Period 2, our team, Los Perdidos, programmed three different Relaybots using **Arduino**, with all the codes uploaded to **GitHub** on three separate branches. For all 3 robots the requirements are that they can pick up an object, complete the course and drop the object off at the end after it finishes its own course, use the different colors for the LED so the customer could undertand in which direction our Relaybot it's going. The budget for the robot was 50 euros, which covered the hardware parts we needed to make them.

King Julian (Robot 1): I worked with Fabiana on programming. We focused on coding the line-following algorithm, object handling, and obstacle avoidance. Our main goal was to ensure the robot could function autonomously and efficiently.

Moto Moto (Robot 2): Matin, Diogo, and Goncalo worked on the second Relaybot. Their task was to program the robot to follow a line and navigate through a maze. They tackled both the sensor integration and pathfinding algorithms.

Mort (Robot 3): Peter and Lennon were responsible for the third Relaybot. Their focus was on programming Mort to use sensors to detect the right path through the maze and solving the maze autonomously.

Website & Dashboard: In addition to working on the robots, I also created the team website, with help from Lennon who worked on the Dashboard component. The website served as a place to showcase robots, document our progress, and track results.

Connectivity: Initially, Lennon was also responsible for the Connectivity aspect. However, two days before the Relaybot race, Lennon decided to bail on us, only creating the connectivity part for his robot.

Throughout the project, everyone contributed to building and maintaining our **group portfolio**. This included documenting each robot's development, detailing our coding process, and ensuring our progress was recorded effectively.

We had to make the robots work for our customers until 15th January, when we had the race between the four teams.

Course Contents

- Job Aid: Tips for Being a Successful Collaborator
- Working on Shared Goals through Teaming
  4m 46s
- Navigating Different Work Styles of Collaborators
  5m 13s
- Knowledge Check: Becoming a Successful Collaborator
- Retake Test

Course Contents

Becoming a Successful Collaborator

Great job! You passed this test.

Minimum score needed: 70%

You scored

86%

Replay celebration

NHL STENDEN
university of applied sciences

P2. Embedded Systems

Name	Student nr.	Group	Year
MIT ALEXIA TEODORA	5587832		

The student is responsible for safe keeping of this card

Assignments

Wk	Subject	Date	Signature
2.1	Basic moves *(Functions)*	18 NOV 2024	M. Bara
2.2	Object avoidance	18 NOV 2024	Bar
2.3	Stay on track *(Arrays)*	15 JAN 2025	Mar. Bar
2.4	Basic game logic	15 JAN 2025	M. Bar
2.5	Connectivity	15 JAN 2025	
2.6	Data storage *(SQL)*	15 JAN 2025	
2.7	Dashboard	15 JAN 2025	
2.8	Circuit Challenge + Portfolio	15 JAN 2025	M. Bar
T1	**Assessment**		

Comments:

his card is proof that the above assignments are properly completed and should be:

- signed off by the appropriate teacher(s) during the guided ateliers

P2. Computer Science

Name	Student nr.	Group	Year
MIT ALEXIA TEODORA	5587832		

The student is responsible for safe keeping of this card

Assignments

Wk	Subject	Date	Signature
2.1	Hello, Mario More, Credit	27 NOV 2024	
	Readability	27 NOV 2024	
	DNA	27 NOV 2024	
2.2	Songs Movies	27 NOV 2024	
	Fifty Ville	27 NOV 2024	
2.3	Trivia	9 DEC 2024	
	Homepage	9 DEC 2024	
	Hand-in Technical Design	20 DEC 2024	
2.4	Birthdays	15 JAN 2025	
	Finance	15 JAN 2025	
2.6	Teamsite	15 JAN 2025	
2.8	Circuit Challenge + Portfolio	15 JAN 2025	
T2	**Assessment**		

Comments:

This card is proof that the above assignments are properly completed and should be:

- signed off by the appropriate teacher(s) during the guided ateliers
- a digital attachment of your portfolio and
- physically handed over at the beginning of the assessment