Bright Expo 2024 Challenges

Bright Expo is a premier global platform designed to ignite creativity and cultivate the talents of aspiring young minds. The 2024 Challenges are crafted to inspire and empower students to demonstrate their abilities across a spectrum of fields—spanning science, technology, engineering, programming, artificial intelligence, and the creative arts. This is more than a competition; it's a dynamic opportunity to delve deep into your potential, bring groundbreaking ideas to life, and engage with a global community driven by the passion to solve real-world challenges through inventive thinking.

Whether you are an inventor with disruptive concepts, a programmer eager to tackle complex coding problems, or an AI enthusiast seeking to make a transformative impact, the Bright Expo Challenges are your stage to shine. Here, you will connect with like-minded peers, push the boundaries of innovation, and be part of an inspiring journey that shapes a brighter and more innovative future.

Embrace the challenge, showcase your ingenuity, and let your ideas illuminate the world!

How to Register

Step 1: Check Your Eligibility

Step 2: Choose Your Challenge Category

Step 3: Visit the Bright Expo Website

Step 4: Upload Your Project Files

Use any cloud storage tool of your choice (e.g., Google Drive, Dropbox, OneDrive) to upload your project files.

Step 5: Share Your Solution

Once your files are ready and uploaded to the cloud storage, generate a shareable link to your project. Send this link, along with your registration details (name, category, and team information if applicable), to our official email: info@brightexpo.org.

Step 6: Receive Confirmation

After submitting your solution, you will receive a confirmation email acknowledging your submission. Be sure to keep this email as it will contain important information about the next steps and event updates.

If you need any assistance during the registration process, please don't hesitate to contact our support team at support@brightexpo.org.

1. Programming Challenge: Decoding World War II Messages

Story:

During World War II, the Nazi army used a simple encryption technique called the "Caesar Cipher" to send secret messages. This method works by shifting each letter in the message by a fixed number of positions in the alphabet. For example, if the letter 'A' is shifted by 3 positions, it becomes 'D'. This encryption technique allows you to decode the encrypted messages by applying specific algorithms.

Challenge:

Create a JavaScript program that can decode messages encrypted using the "Caesar Cipher" technique. Note that the shift value (number of positions each letter is shifted) may vary for each message. The encrypted messages should be converted back to their original, meaningful text.

What is a Shift?

A shift in the Caesar Cipher algorithm refers to the number of positions each letter is moved in the alphabet. For example, if the shift is 3, the letter 'A' becomes 'D', and the letter 'B' becomes 'E'. To decode a message, you need to shift the letters in the opposite direction by the specified number.

Challenge Details:

Input : **Encrypted Messages:** An array of strings containing the encrypted messages. For example:

```
const encodedMessages = [
    "Khoor zruog",
    "Wklv lv d whvw phvvdjh"
  ];
```

Output : **Decoded Messages:** An array of strings containing the decoded, meaningful messages. For example:

```
const decodedMessages = [
    "Hello world",
    "This is a test message"
  ];
```

Technical Requirements:

1. **HTML and CSS:** Design a user interface that includes input fields for encrypted messages, the shift value, and displays the decoded messages.
2. **JavaScript:** Implement the Caesar Cipher algorithm to decode the messages.

2. Programming Challenge: The Treasure Hunter in the Maze of Death

Story:

In a mythical land, you are a treasure hunter who has entered a dark and mysterious maze. The maze is full of complex paths, stone walls, and deadly traps. At every stage of your journey, the paths you can take change randomly, and if you choose the wrong one, you will fall into a dangerous trap.

The maze is designed in such a way that, in order to find the treasure, you must discover a specific path from the starting point to the endpoint. You can use certain symbols and codes to find the correct path. These symbols are generated based on a special algorithm that influences each stage of the path.

Challenge:

You must write a program using **JavaScript** that can discover a path through the maze to reach the treasure. The maze is represented as an NxN grid, containing obstacles (walls) and free paths. Your program should be able to find the best path from the starting point to the endpoint and avoid obstacles and traps.

Challenge Details:

Input : A matrix NxN that represents the maze. Each cell in the matrix can have one of the following values:

- `0` : Represents an open path
- `1` : Represents a wall
- `T` : Represents a trap (If you step on this, you lose)
- `S` : The start point
- `E` : The endpoint (the treasure)

```
const maze = [
    ['S', '0', '1', 'T'],
    ['1', '0', '1', '0'],
    ['1', '0', '0', '1'],
    ['T', '0', 'E', '1']
  ];
```

Output : For this maze, the output would be:

```
[
  [0, 0],  // Start from [0, 0] (S)
  [0, 1],  // Move to [0, 1]
  [1, 1],  // Move to [1, 1]
  [2, 1],  // Move to [2, 1]
  [2, 2],  // Move to [2, 2]
  [3, 1],  // Move to [3, 1]
  [3, 2]   // Reach [3, 2] (E)
]
```

If no path is found (for example, if traps or walls block the way), the program output will be:
"No Path Found"

3. Programming Challenge: Warehouse Management with JavaScript

Story: In an era where the world is increasingly becoming digital, you, as a proficient JavaScript developer, are tasked with helping a warehouse manager solve the issue of organizing items in their warehouse. Your mission is to write a program that assists this warehouse manager in placing their goods efficiently in a 2D grid representing the warehouse.

Challenge: You need to design a 2D warehouse grid. The width of the warehouse is represented by the elements in an array, and the height is represented by the number of arrays. For example, if the warehouse has a width of 4 and a height of 2, it would look like this:

```
[ 0 , 0 , 0 , 0 ]
[ 0 , 0 , 0 , 0 ]
```

In this setup:

- The number 0 represents an empty space.
- The number 1 represents the first item.
- The number 2 represents the second item, and so on.

Your task is to write a program where the warehouse manager specifies the number of items and their dimensions (width and height). The program will place these items in the available spaces in the warehouse.

Input:

- Width of the warehouse
- Height of the warehouse
- Number of items
- Dimensions of each item (width and height)

Example: Let's say we have a warehouse with a width of 3 and a height of 4, and two items to place:

- The first item has a width of 1 and a height of 2.
- The second item has a width of 2 and a height of 2.

The placement of these items in the warehouse will look like this:

```
[ 1 , 0 , 0 ]
[ 1 , 0 , 0 ]
[ 2 , 2 , 0 ]
[ 2 , 2 , 0 ]
```

Technical Requirements:

For this challenge, you need to design an attractive and user-friendly interface (UI) that includes input fields for the warehouse's width and height, the number of items, the dimensions of each item, and a graphical display of the 2D warehouse with the placed items.

4. Programming Challenge: Legal Chess Piece Movements

Story: Imagine you are designing a chess game. Before you can build the full game, you need to understand how each piece is allowed to move. In this challenge, you'll focus on two of the most iconic chess pieces: the **Knight** and the **Bishop**. Your task is to determine the valid moves for these pieces on a standard 8x8 chessboard.

Challenge: You are given an 8x8 chessboard represented as follows:

```
[ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ]
[ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ]
[ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ]
[ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ]
[ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ]
[ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ]
[ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ]
[ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ]
```

- **Knight (1):** Wherever the knight is placed, its valid moves should be marked with the number 2.
- **Bishop (3):** Wherever the bishop is placed, its valid moves should be marked with the number 4.

For example, if the knight is placed in the middle of the board, it moves as follows:

```
[ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ]
[ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ]
[ 0 , 0 , 2 , 0 , 2 , 0 , 0 , 0 ]
[ 0 , 2 , 0 , 0 , 0 , 2 , 0 , 0 ]
[ 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 ]
[ 0 , 2 , 0 , 0 , 0 , 2 , 0 , 0 ]
[ 0 , 0 , 2 , 0 , 2 , 0 , 0 , 0 ]
[ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ]
```

If the bishop is placed somewhere, its valid diagonal moves are as follows:

```
[ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ]
[ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 4 ]
[ 4 , 0 , 0 , 0 , 0 , 0 , 4 , 0 ]
[ 0 , 4 , 0 , 0 , 0 , 4 , 0 , 0 ]
[ 0 , 0 , 4 , 0 , 4 , 0 , 0 , 0 ]
[ 0 , 0 , 0 , 3 , 0 , 0 , 0 , 0 ]
[ 0 , 0 , 4 , 0 , 4 , 0 , 0 , 0 ]
[ 0 , 4 , 0 , 0 , 0 , 4 , 0 , 0 ]
```

5. Programming Challenge: Snake Game Design

Story: Imagine you are tasked with creating a classic Snake game using JavaScript. In this game, the goal is to control the snake using keyboard arrow keys, avoid the boundaries of the game area, and eat the food items that appear within the environment. The more food the snake eats, the longer and more challenging it becomes to control.

Challenge: Design and implement a visually appealing and user-friendly Snake game where players can control the snake's movement with the arrow keys on the keyboard. The game should include the following features:

1. **Movement Control:** Allow the player to control the snake's direction using the arrow keys (up, down, left, right).
2. **Boundary Collision:** Ensure the snake cannot cross the boundaries of the game area.
3. **Food Consumption:** Implement food items that randomly appear within the game area. The snake should grow longer each time it consumes food.
4. **Increasing Difficulty:** As the snake grows longer, make it progressively harder to control.

Technical Requirements:

1. **HTML and CSS:**
    - Create an engaging and user-friendly user interface (UI) for the game. The design should include a clear game area where the snake moves, food items, and a score display.
2. **JavaScript:**
    - Implement the game logic for snake movement, boundary collision detection, food generation, and score tracking. Ensure the game dynamics are smooth and responsive to user input.

Sample Input:

```
const gameArea = document.getElementById('gameArea');
const gameSize = 400;
const snakeSize = 20;
let snake = [{ x: 100, y: 100 }];
let direction = 'RIGHT';
let food = { x: 200, y: 200 };
let score = 0;
```

Output:

When you run this code, you'll see a 400x400 game area with a snake and food items. The snake moves according to the arrow keys, grows when it eats food, and the game ends if it hits the boundaries or itself.

6. Online Business Card with QR Code

Story:

You are a web services designer tasked with creating a digital business card for your clients. Each business card must include the individual’s name, contact details, address, and a unique QR Code. Upon scanning the QR Code, the business card with the person's information should appear.

Challenge:

Using PHP and SQL for the backend, create a program that generates a digital business card for each individual. The business card should contain a QR Code that, when scanned, displays the individual's information (such as name, phone number, and address). The frontend should be built with HTML and CSS to provide a clean and professional display of the business card.

Input:

Details for the individual, including:

- Name
- Phone number
- Address

Output:

A digital business card with a unique QR Code for each individual, which when scanned displays the person's information.

Sample Input Code (PHP):

```php
<?php
// Connect to the database (SQL)
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "business_cards";
```

Backend Details:

- **Backend**: PHP and SQL for storing user data and generating QR codes.
- **Frontend**: HTML and CSS for designing and displaying the business card interface.

1. Water Allocation Optimization in a Post-Apocalyptic Future

Story:

It's the year 3050, and the Earth has become unbearably hot due to unpredictable climate change. Water resources are incredibly limited, and cities heavily rely on their available water. However, these resources are running dangerously low and require intelligent water management. As an AI developer, you are tasked with optimizing the allocation of water resources based on population, agriculture, industry, and other needs. However, the conditions are highly unstable, and the data is uncertain. Your goal is to ensure that no city suffers from a water shortage while still meeting diverse demands.

Challenge:

As an AI developer, your task is to build an optimization model using Python that can allocate water resources for different uses in a city. Your model should take the following information as input and determine how much water should be allocated for drinking, agriculture, industry, hygiene, and sanitation:

1. City population and its demographic distribution (age, gender, and special needs).
2. The number of agricultural lands and the types of crops grown (crops that require high water consumption and those that require less).
3. Weather forecast data and the probability of changes in water availability (unexpected changes in temperature, rainfall, and drought).
4. The amount of water that needs to be allocated for daily use (drinking, hygiene, and sanitation).

Additional Complexity:

Evaporation rate: Due to extremely high temperatures, the evaporation rate in certain regions might increase, reducing the available water resources.

Input Example:

```
{
    "city_population": 1000000,  # Total population of the city
    "agricultural_land": 5000,  # Number of hectares of agricultural land
    "crop_types": [
        {"name": "wheat", "water_needed_per_hectare": 5000},
        {"name": "corn", "water_needed_per_hectare": 8000}
    ],
    "industrial_water_demand": 10000,  # Water demand for industry (in cubic meters)
    "available_water": 1000000,  # Total available water (in cubic meters)
    "evaporation_rate": 0.05,  # Evaporation rate (percentage)
    "weather_forecast": [
        {"day": 1, "rainfall": 10, "temperature": 45},
        {"day": 2, "rainfall": 0, "temperature": 50},
        {"day": 3, "rainfall": 5, "temperature": 47}
    ]
}
```

Expected Output Example:

```
{
    "drinking_water": 200000,  # Water allocated for drinking (in cubic meters)
    "agriculture_water": 500000,  # Water allocated for agriculture (in cubic meters)
    "industrial_water": 10000,  # Water allocated for industry (in cubic meters)
    "hygiene_water": 50000  # Water allocated for hygiene and sanitation (in cubic meters)
}
```

2. Intelligent Traffic Control System Using Real-Time Image Analysis

Story:

It's the year 2045, and traffic congestion in major cities has reached an all-time high. Streets are packed with vehicles, and traditional traffic management systems can no longer handle the volume and complexity of modern urban traffic. City governments are seeking innovative solutions for intelligent traffic management. participants must use street images either taken manually or sourced from online resources (such as Google Street View or other real-world street photos).

Challenge:

Develop an **AI model** that uses **image processing** and **deep learning** techniques to analyze street images and manage traffic light control. The model should process the images to estimate traffic volume and movement, and then generate an optimized schedule for traffic lights.

Challenge Steps:

1. Participants must collect street images or use images from online sources that show various traffic conditions.
2. The AI model should analyze these images to identify vehicle counts, traffic density, and vehicle speeds.
3. Based on the image analysis, the model should optimize traffic light timings, determining which light should be green or red and for how long.

Inputs:

- **Street Images:** Participants should provide various images of streets and intersections showing vehicles and traffic conditions.

Outputs:

- Duration for each green and red light for different roads.
- Prioritization of roads with heavier traffic or longer waiting times.

Sample input :

```
{
    "images": ["image1.jpg", "image2.jpg"],
    "weather": {"rain": 0, "temperature": 25}
}
```

Sample Output:

```
{
    "traffic_light_schedule": {
        "intersection_1": {
            "north_road": {"green_time": 30, "red_time": 60},
            "south_road": {"green_time": 50, "red_time": 40}
        },
        "intersection_2": {
            "east_road": {"green_time": 20, "red_time": 80},
            "west_road": {"green_time": 45, "red_time": 55}
        }
    }
}
```

3. Background Removal and Replacement with Paris Eiffel TowerStory:

Story:

Imagine you want to create a photo where it appears as if you are standing in front of the Eiffel Tower in Paris, but you don't have a picture there yet. You are tasked with designing a system that takes a user's uploaded image, removes its background, and replaces it with a scenic image of Paris, featuring the Eiffel Tower.

Challenge:

Create a Python-based system that allows a user to upload an image of themselves. The system should then automatically remove the background from the uploaded image and replace it with a preset image of Paris, specifically one featuring the Eiffel Tower.

Input:

- An image of the user (uploaded).
- A background image of Paris with the Eiffel Tower.

Output:

- A new image where the user is placed in front of the Eiffel Tower in Paris.

Sample Input Code (Python):

```python
import cv2
import numpy as np
from rembg import remove
from PIL import Image

# Load user-uploaded image
user_image_path = "user_image.png"
user_image = Image.open(user_image_path)
```

4. **Predicting Student Dropout Risk Using Educational Data**

Story:

A school principal is concerned that some students may be at risk of dropping out due to various factors such as academic challenges, personal issues, or disciplinary problems. To address this concern, the principal has hired you as an AI developer. Your task is to develop a Python program that uses various data points, including class attendance, grades, disciplinary records, and other relevant information, to predict the likelihood of each student dropping out.

Challenge:

Develop an AI model that can predict the probability of students dropping out based on data such as class attendance, academic performance, disciplinary status, extracurricular activities, and other relevant information. The model should analyze these factors and provide a percentage indicating the likelihood of each student dropping out.

Inputs:

- **Class Attendance Data:** Information on attendance or absences.
- **Academic Grades:** Grades for exams and assignments.
- **Disciplinary Records:** Records of disciplinary actions.
- **Extracurricular Activities:** Participation in activities or clubs.
- **Personal Information:** Age, gender, economic status, etc.

Sample Inputs:

```
input_data = {
    'attendance_absent_days': 2,  # Number of days absent
    'average_grade': 75,          # Average grade
    'discipline_incidents': 1,    # Number of disciplinary incidents
    'extracurricular_participation': 1,  # 1: Active, 0: Not Active
    'age': 18,                    # Age of the student
    'economic_status': 1          # 1: Middle, 2: High
}
```

Sample Outputs:

```
#Sample Output:
print(f"Predicted Dropout Risk: {predicted_risk}%")
```

5. Real-Time Gym Exercise Instruction Using Motion Modeling:

Imagine you are developing a system to help users with their gym workouts by providing real-time feedback on their exercise movements. The goal is to create a system that can detect and analyze the user's body movements during exercises using a webcam. The system should visually represent the user's body using lines and keypoints and offer feedback to correct or improve their exercise form.

Challenge:

Develop a Python program that uses a webcam to capture the user's movements and apply motion modeling techniques to display the body's key points and lines in real-time. The system should automatically detect and analyze the user's exercise movements, identify weaknesses or mistakes, and provide corrective feedback.

Inputs:

- **Webcam Feed:** Real-time video input from the user's webcam.
- **Exercise Data:** Specific movements and poses that need to be analyzed.

Output:

- **Visual Representation:** A real-time visual display of the user's body with lines and keypoints indicating their posture and movement.
- **Feedback:** Automated suggestions or corrections for improving exercise form based on the detected movements.

Sample Inputs:

```
import cv2
import mediapipe as mp

# Initialize MediaPipe Pose
mp_pose = mp.solutions.pose
pose = mp_pose.Pose()
mp_drawing = mp.solutions.drawing_utils
```

Explanation:

- **MediaPipe Pose:** Used for detecting and tracking human poses.
- **Webcam Feed:** Captures real-time video of the user.
- **Visual Representation:** Draws keypoints and connections on the user's body to show their posture.
- **Feedback:** Can be extended to include analysis and suggestions for improving exercise form based on the keypoints.

6. Create Music Based on a Face

Story:

Imagine if every person's face could generate a unique piece of music based on their emotions. Different facial expressions, like happiness, sadness, surprise, or anger, could inspire various musical notes. Now, your task is to design a system that analyzes a person's facial expression and generates music that corresponds to their emotional state.

Challenge:

Design an AI system that, after receiving an image of a person's face, analyzes the facial expression to determine the dominant emotion. Based on the detected emotion, the system will generate or play a piece of music that matches the emotion (e.g., cheerful for happy, somber for sad). The system should be built using Python and facial emotion recognition technology.

Input:

- An image of the user's face (uploaded by the user).

Output:

- Play a piece of music corresponding to the detected emotion (e.g., happy, sad, or calm music).

Sample Input Code (Python):

```
import cv2
from deepface import DeepFace
import random
```

Explanation:

1. **Facial Emotion Detection**: Using the DeepFace library, the system can analyze the uploaded face image and detect the dominant emotion, such as happiness, sadness, or anger.
2. **Music Generation**: Based on the detected emotion, the system selects and plays a corresponding music track, like a cheerful song for happy emotions or a calm melody for neutral emotions.

7. Create a Type-to-Speech (TTS) Robot

Story:

You are tasked with creating a text-to-speech (TTS) robot that can take any input text and speak it aloud in your own voice. The challenge is to implement this without using any external APIs. The more languages your TTS robot can support, the higher your score will be.

Challenge:

Write a Python program that converts a given text into speech using your own recorded voice. The system should analyze the text and synthesize speech based on pre-recorded audio snippets of your voice, ensuring that the output sounds natural. You are not allowed to use any external APIs or libraries for text-to-speech conversion—everything must be built from scratch or use open-source techniques that you develop independently.

Input:

- A text string provided by the user.

Output:

- The text spoken aloud using the user's own recorded voice.

Sample Input Code (Python):

```
import os
import wave
from pydub import AudioSegment
```

Requirements:

- **Pre-recorded Phonemes**: You will need to record your voice for each phoneme (or sound) in the language you are targeting.
- **Multiple Languages**: The more phoneme libraries you record in different languages, the more comprehensive your TTS robot will be.