|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Weeks** | **Topics** | **Links** | **YouTube** | **Remarks** |
| 0 | 5 Door Challenge |  |  |  |
| 1 | Appointment Scheduling Optimization |  |  |  |
| 2 | Patient Triage System |  |  |  |
| 3 | Hospital Bed Allocation |  |  |  |
| 4 | Health Condition Prediction Based on Symptoms |  |  |  |
| 5 | Health Data Clustering |  |  |  |
| 6 | Symptom Progression Modeling |  |  |  |
| 7 | Emergency Response Optimization |  |  |  |
| 8 | Drug Dosage Calculation |  |  |  |
| 9 | Patient Data Privacy (Anonymization) |  |  |  |
| 10 | Disease Outbreak Prediction |  |  |  |
| MIDTERM | | | | |
| 12 | Fitness Tracker Data Analysis |  |  |  |
| 13 | Medical Record Search Engine |  |  |  |
| 14 | Predicting Hospital Readmission |  |  |  |
| 15 | Chronic Disease Risk Assessment |  |  |  |
| 16 | Medical Inventory Management |  |  |  |
| 17 | Medical Image Processing for Diagnosis |  |  |  |
| 18 | Health Insurance Claims Fraud Detection |  |  |  |
| 19 | Personalized Treatment Plan Generator |  |  |  |
| 20 | Checking |  |  |  |
| FINALS | | | | |

**5 Door Challenge**

The 5 Door Challenge is a simulation problem where a set of 5 doors is toggled multiple times based on a set of passes. Initially, all the doors are closed. In each pass, specific doors are toggled (opened or closed) based on the pass number. After completing all the passes, the doors that have been toggled an odd number of times will be open, while the doors toggled an even number of times will remain closed.

In this challenge:

* **Pass 1** toggles every door.
* **Pass 2** toggles every second door.
* **Pass 3** toggles every third door, and so on.
* The challenge concludes after 5 passes, showing which doors remain open.

**Example Output**

After 5 passes on 5 doors, the states of the doors will change as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Door Number | Pass 1 | Pass 2 | Pass 3 | Pass 4 | Pass 5 |
| Door 1 | Open | Open | Open | Open | Open |
| Door 2 | Open | Closed | Closed | Closed | Closed |
| Door 3 | Open | Open | Closed | Closed | Open |
| Door 4 | Open | Open | Open | Closed | Closed |
| Door 5 | Open | Open | Open | Open | Closed |

Explanation:

* **Pass 1**: All doors are toggled to open.
* **Pass 2**: Every second door is toggled (doors 2 and 4 are toggled).
* **Pass 3**: Every third door is toggled (door 3 is toggled).
* **Pass 4**: Every fourth door is toggled (door 4 is toggled).
* **Pass 5**: Every fifth door is toggled (door 5 is toggled).

At the end, the doors that have been toggled an odd number of times will remain open, and those toggled an even number of times will remain closed.

3. **Step-by-Step Guide on How to Implement the 5 Door Challenge in ASP.NET Core MVC Using C#**

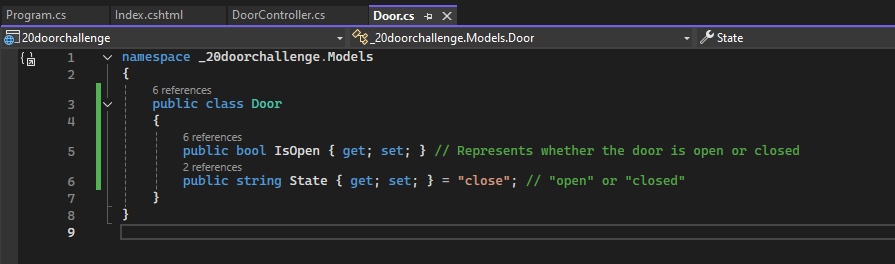
**Step 1: Create a New ASP.NET Core MVC Project**

1. **Open Visual Studio:**
   * If you don't have Visual Studio installed, you can download it from [here](https://visualstudio.microsoft.com/).
2. **Create a New Project:**
   * Select Create a new project.
   * Choose ASP.NET Core Web Application.
   * Click Next.
3. **Configure Your Project:**
   * Name the project (e.g., TwentyDoorChallenge).
   * Choose the location where you want to save the project.
   * Click Create.
4. **Select Template:**
   * Choose the Web Application (Model-View-Controller) template.
   * Make sure .NET Core and ASP.NET Core version is selected correctly.
   * Click Create.

**Step 2: Create the Door Model**

1. **Add a Door Model:**
   * In the Models folder, right-click and select Add -> Class.
   * Name the class Door.cs.
   * Add the following code to represent the state of each door:

|  |
| --- |
| 1. namespace \_20doorchallenge.Models 2. { 3. public class Door 4. { 5. public bool IsOpen { get; set; } // Represents whether the door is open or closed 6. public string State { get; set; } = "close"; // "open" or "closed" 7. } 8. } |



**Step 3: Create the Controller**

1. **Add a Door Controller:**
   * Right-click on the Controllers folder and select Add -> Controller.
   * Choose MVC Controller - Empty and click Add.
   * Name the controller DoorController.
2. **Add Logic for the 20 Door Challenge:**
   * In the DoorController.cs, add the following code:

|  |
| --- |
| 1. using Microsoft.AspNetCore.Mvc; 2. using \_20doorchallenge.Models; 3. namespace \_20doorchallenge.Controllers 4. { 5. public class DoorController : Controller 6. { 7. private static List<Door> doors = new List<Door>(); 8. private static int passCount = 0; 9. public DoorController() 10. { 11. // Initialize the doors (only needed once when the app starts) 12. if (doors.Count == 0) 13. { 14. for (int i = 1; i <= 5; i++) // Initialize only 5 doors 15. { 16. doors.Add(new Door { IsOpen = false, State = "Close" }); 17. } 18. } 19. } 20. [HttpGet] 21. public IActionResult Index() 22. { 23. return View(doors); // Return the doors state to the view 24. } 25. [HttpPost] 26. public IActionResult Toggle() 27. { 28. if (passCount < 5) // Check if we have passed less than 5 times 29. { 30. passCount++; 31. // Run the appropriate pass based on the count 32. for (int i = 1; i <= doors.Count; i++) 33. { 34. if (i % passCount == 0) // Toggle every 'passCount' door 35. { 36. doors[i - 1].IsOpen = !doors[i - 1].IsOpen; // Toggle the door state 37. doors[i - 1].State = doors[i - 1].IsOpen ? "Open" : "Close"; 38. } 39. } 40. } 41. ViewData["PassCount"] = passCount; // Display the current pass count 42. return View("Index", doors); 43. } 44. } 45. } |

**Step 4: Create the View**

1. **Add a View for Displaying the Doors:**
   * Right-click on the Views folder, then the Door folder inside Views (if it doesn't exist, create it).
   * Select Add -> View.
   * Name the view Index.cshtml.
2. **Create the HTML Layout to Display the Doors:**
   * In Index.cshtml, add the following code to display the doors with their states (open or closed):

|  |
| --- |
| 1. @model List<\_20doorchallenge.Models.Door> 2. <!DOCTYPE html> 3. <html lang="en"> 4. <head> 5. <meta charset="UTF-8"> 6. <meta name="viewport" content="width=device-width, initial-scale=1.0"> 7. <title>5 Door Challenge</title> 8. <style> 9. .door { 10. display: inline-block; 11. width: 50px; 12. height: 100px; 13. margin: 5px; 14. background-color: gray; 15. text-align: center; 16. vertical-align: top; 17. font-size: 18px; 18. } 19. .door.open { 20. background-color: green; 21. color: white; 22. } 23. </style> 24. </head> 25. <body> 26. <h1>5 Door Challenge</h1> 27. <!-- Display Pass Count --> 28. <p>Current Pass: @ViewData["PassCount"]</p> 29. <!-- Display the doors' states --> 30. <div> 31. @for (int i = 0; i < Model.Count; i++) 32. { 33. <div class="door @(Model[i].IsOpen ? "open" : "")"> 34. @if (Model[i].IsOpen) 35. { 36. @:Open 37. } 38. else 39. { 40. @:Closed 41. } 42. </div> 43. } 44. </div> 45. <!-- Button to trigger the toggle logic --> 46. <**form** method="post" **asp-action**="Toggle"> 47. <!-- Disable the button after 5 passes --> 48. <button type="submit" @(ViewData["PassCount"] != null && (int)ViewData["PassCount"] >= 5 ? "disabled" : "")> 49. Next Pass 50. </button> 51. </**form**> 52. </body> 53. </html> |

**Step 5: Set the Route in Program.cs (Optional for Older Versions)**

1. Open Program.cs if you're using ASP.NET Core 3.x or earlier.
2. Add or check that the following lines are in the Configure method

|  |
| --- |
| 1. app.MapControllerRoute( 2. name: "default", 3. pattern: "{controller=Door}/{action=Index}/{id?}"); |

**Step 6: Run the Application**

1. **Build the Project:**
   * Click on Build -> Build Solution to ensure there are no errors.
2. **Run the Application:**
   * Press Ctrl + F5 (or click the "Play" button in Visual Studio) to run the application.
   * This will open the app in your browser.

**Step 7: Test the Application**

1. **Test the 20 Door Challenge:**
   * When you open the page, you should see 20 doors displayed.
   * The doors will be toggled based on the logic you implemented: after 20 passes, some doors will be open (green) and others will remain closed (gray).
   * The toggling will be based on the rules: the first pass toggles all doors, the second toggles every second door, and so on.

**Additional Notes:**

* You can customize the door styling and layout to suit your preferences (e.g., using images for doors).
* If you want to add more interactivity (e.g., buttons to reset or toggle doors individually), you can add additional functionality in the controller and view.
* Consider adding unit tests to verify that the door logic is working as expected.

By following these steps, you'll create a simple yet effective implementation of the 20 Door Challenge using C# and ASP.NET Core MVC.

 “**Appointment Scheduling Optimization"**

* **Concept:** Scheduling, Time Complexity
* **Description:** Given a list of patients and available time slots for doctors, design an algorithm to schedule appointments optimally (minimize waiting time for patients, while maximizing doctor availability).

 “**Patient Triage System"**

* **Concept:** Priority Queues, Decision Trees
* **Description:** Develop a system that ranks patients based on their symptoms (e.g., severity, type) and prioritizes their visits. This could use a priority queue or decision tree to evaluate severity.

 “**Hospital Bed Allocation"**

* **Concept:** Dynamic Programming, Greedy Algorithms
* **Description:** A hospital has a limited number of beds and a variable number of patients arriving each day. Write an algorithm to allocate beds based on patient severity and hospital capacity, maximizing the number of patients treated.

 **"Health Condition Prediction Based on Symptoms"**

* **Concept:** Decision Trees, Machine Learning
* **Description:** Given a list of symptoms, predict the most likely health condition (e.g., flu, cold, COVID-19) using a decision tree or rule-based system.

 **"Health Data Clustering"**

* **Concept:** Clustering Algorithms (K-Means, Hierarchical Clustering)
* **Description:** Given a dataset of patients' health records (e.g., age, symptoms, medical history), cluster patients into groups for personalized treatment.

 **"Symptom Progression Modeling"**

* **Concept:** Regression Analysis, Predictive Modeling
* **Description:** Given data on how a patient's symptoms progress over time, predict when the patient's condition might worsen or improve. Use regression algorithms to model symptom progression.

 **"Emergency Response Optimization"**

* **Concept:** Graph Algorithms (Dijkstra's, A\* Search)
* **Description:** A hospital needs to respond to multiple emergency calls from different locations. Implement an algorithm that finds the fastest route for ambulances to take to reach patients in time.

 **"Drug Dosage Calculation"**

* **Concept:** Algorithmic Computation
* **Description:** Given a patient's age, weight, and medical condition, design an algorithm to calculate the correct drug dosage. Factor in the weight-based calculation and maximum allowed dosage.

 **"Patient Data Privacy (Anonymization)"**

* **Concept:** Data Anonymization, Cryptography
* **Description:** Design an algorithm to anonymize patient data to ensure privacy. This could include removing personally identifiable information and implementing encryption for secure data handling.

 **"Disease Outbreak Prediction"**

* **Concept:** Time Series Analysis, Machine Learning
* **Description:** Given historical data on disease outbreaks, create an algorithm to predict the likelihood of future outbreaks in specific regions.

 **"Fitness Tracker Data Analysis"**

* **Concept:** Data Analysis, Pattern Recognition
* **Description:** Analyze fitness tracker data to identify abnormal health patterns, like irregular heart rates or sleep disturbances, and provide personalized recommendations.

 **"Medical Record Search Engine"**

* **Concept:** Search Algorithms, Hashing
* **Description:** Create an efficient search engine for a healthcare system that allows users (doctors, patients) to search through medical records by various parameters (symptoms, diagnosis, medication).

 **"Predicting Hospital Readmission"**

* **Concept:** Logistic Regression, Predictive Analytics
* **Description:** Using historical data, predict whether a patient is likely to be readmitted to the hospital within 30 days after discharge based on factors like their previous condition, age, and treatment.

 **"Chronic Disease Risk Assessment"**

* **Concept:** Classification Algorithms (Logistic Regression, SVM)
* **Description:** Given a patient’s medical history and lifestyle factors (e.g., diet, exercise), create an algorithm to assess the risk of developing chronic diseases such as diabetes or heart disease.

 **"Medical Inventory Management"**

* **Concept:** Dynamic Programming, Supply Chain Optimization
* **Description:** Optimize the inventory of medical supplies in a hospital or clinic by predicting the required amount of medication or equipment based on patient demand and historical data.

 **"Medical Image Processing for Diagnosis"**

* **Concept:** Image Recognition, Neural Networks
* **Description:** Develop an algorithm that can analyze medical images (e.g., X-rays, MRIs) to assist doctors in diagnosing conditions such as pneumonia or fractures.

 **"Health Insurance Claims Fraud Detection"**

* **Concept:** Anomaly Detection, Machine Learning
* **Description:** Design an algorithm to detect fraudulent health insurance claims based on patterns in claim data (e.g., frequency of claims, type of services, and location).

 **"Personalized Treatment Plan Generator"**

* **Concept:** Machine Learning, Personalized Health Algorithms
* **Description:** Given a patient’s health data (e.g., age, condition, previous treatments), generate a personalized treatment plan that optimizes outcomes based on historical success rates for similar patients.