**Session 1 :**

This session is about building a pizza store app using Blazor. The app will allow users to order pizzas, customize them, and track order deliveries. The initial solution for the pizza store app is available in a repository, which includes four projects: BlazingPizza.Client, BlazingPizza.Server, BlazingPizza.Shared, and BlazingPizza.ComponentsLibrary. The BlazingPizza.Server project should be set as the startup project. The home page component is implemented as a single component that currently displays a simple home page. The next step is to display the list of available pizza specials. A @code block with a list field is added to the Index.razor page to keep track of available specials. An HttpClient is injected into the Index component using the @inject directive to retrieve the list of pizza specials. The OnInitializedAsync method is overridden to handle the deserialization of the response JSON using the GetFromJsonAsync<T>() method. A markup is then added to the Index component to list the pizza specials. The layout component for the pizza store app is defined in Shared/MainLayout.razor, which inherits from LayoutComponentBase. The MainLayout component is then updated to define a top bar with a branding logo and a nav link for the home page.

**Session 2 :**

In this Blazor app tutorial, we are adding functionality to enable users to customize their pizza orders. We begin by adding an event handler to listen for clicks on pizza specials and show a pizza customization dialog. We also add some additional fields to the @code block to track the pizza being customized and whether the pizza customization dialog is visible.

Next, we create a new component called ConfigurePizzaDialog.razor that lets users specify the size and toppings for their pizza. We add a slider to the dialog to allow users to specify the pizza size and use data binding to update the pizza size stored in the configuringPizza object when the slider is moved.

Finally, we add buttons to the dialog to let users cancel or add the pizza to their order. We also update the Pages/Index.razor file to show the ConfigurePizzaDialog when a pizza special is selected, and add functionality to close the dialog when the user clicks the cancel button.

**Session 3 :**

**Session 4 :**

The AppState pattern is a pattern in Blazor applications that involves adding an object to the dependency injection (DI) container to coordinate state between related components. This object, typically called the AppState, can outlive the components and hold on to state even when the UI changes. The pattern leads to greater separation between presentation (components) and business logic.

The AppState pattern solves the problem of state management in Blazor applications by moving shared state outside of components and into the AppState. Components call methods on the AppState to trigger a state change. EventCallback takes care of dispatching change notifications to the appropriate components. This approach prevents state from being lost when a user navigates between pages in the application.

**Session 5 :**

The instructions are for adding a checkout page to a pizza ordering app in Blazor. The steps involve creating a new page component for the checkout, capturing the delivery address using a reusable AddressEditor component, and adding server-side and client-side validation to ensure a valid address is entered before an order can be submitted. The article provides code snippets and instructions on how to perform each step.

**Session 6 :**

**Session 7 :**

In this session, the tutorial covers using JavaScript interop to add a real-time map to the order status page for tracking pizza delivery. The Map component is pre-built and uses dependency injection to get an IJSRuntime instance that can be used to make JavaScript calls to browser APIs or existing JavaScript libraries. The Map component renders a div with a unique ID and then calls the deliveryMap.showOrUpdate function to display the map with specified markers passed to the Map component.

The tutorial then adds a confirm prompt using JavaScript interop to verify if the user really wants to remove the pizza from the order when the Remove button is clicked. A Confirm extension method is added off of IJSRuntime using the built-in JavaScript confirm function. The Index component is injected with the IJSRuntime service to make JavaScript interop calls, and an async RemovePizza method is added that calls the Confirm method to prompt the user before removing the pizza from the order.

**Session 8 :**

**Session 9 :**

This passage explains how to add Progressive Web App (PWA) features to a Blazor web application. PWA features include installing the app into the OS taskbar or home screen, working offline, and receiving push notifications. To enable these features, a service worker is required, which is a small JavaScript file that provides event handlers that the browser can invoke outside of the application's context. The article explains how to add a service worker and make the app installable by adding a manifest.json file and a link to it in the index.html file. The article also explains how to send push notifications by getting a subscription from the user's browser and using a web API to send the notification.