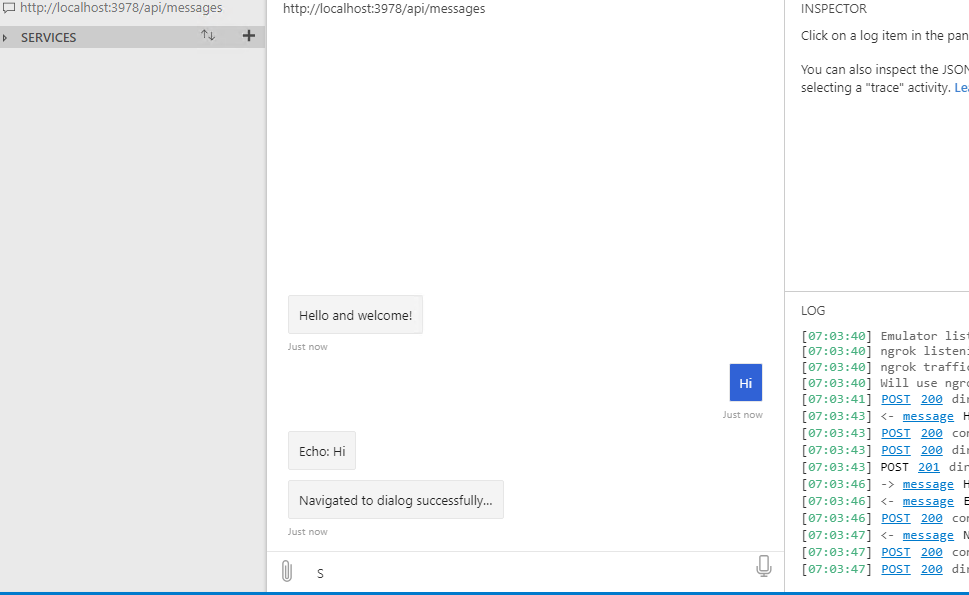
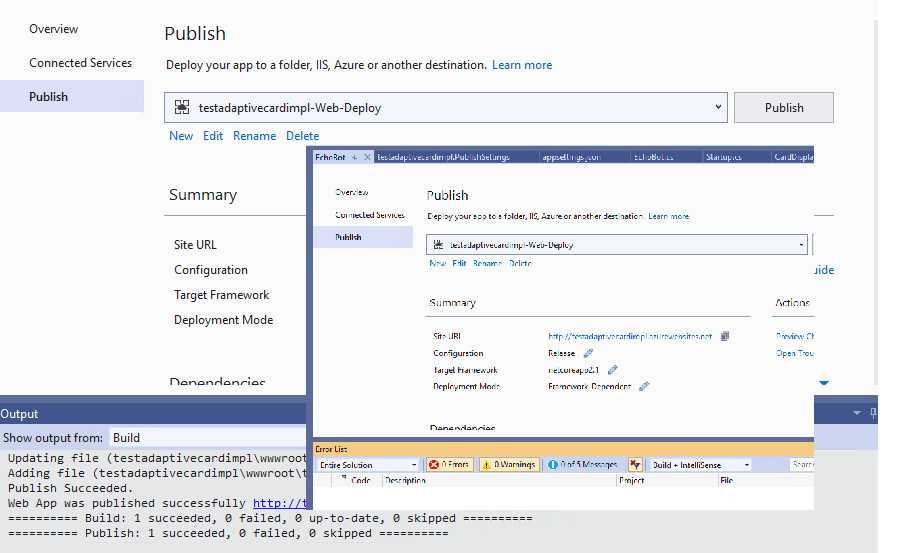
As suggested created a sample BOT which just echoes back the input on top of it just added a simple dialog navigation without anything else which just displays a message.

Created a sample bot which just echoes back the user input and navigates to a dialog and displays message Navigated successfully:

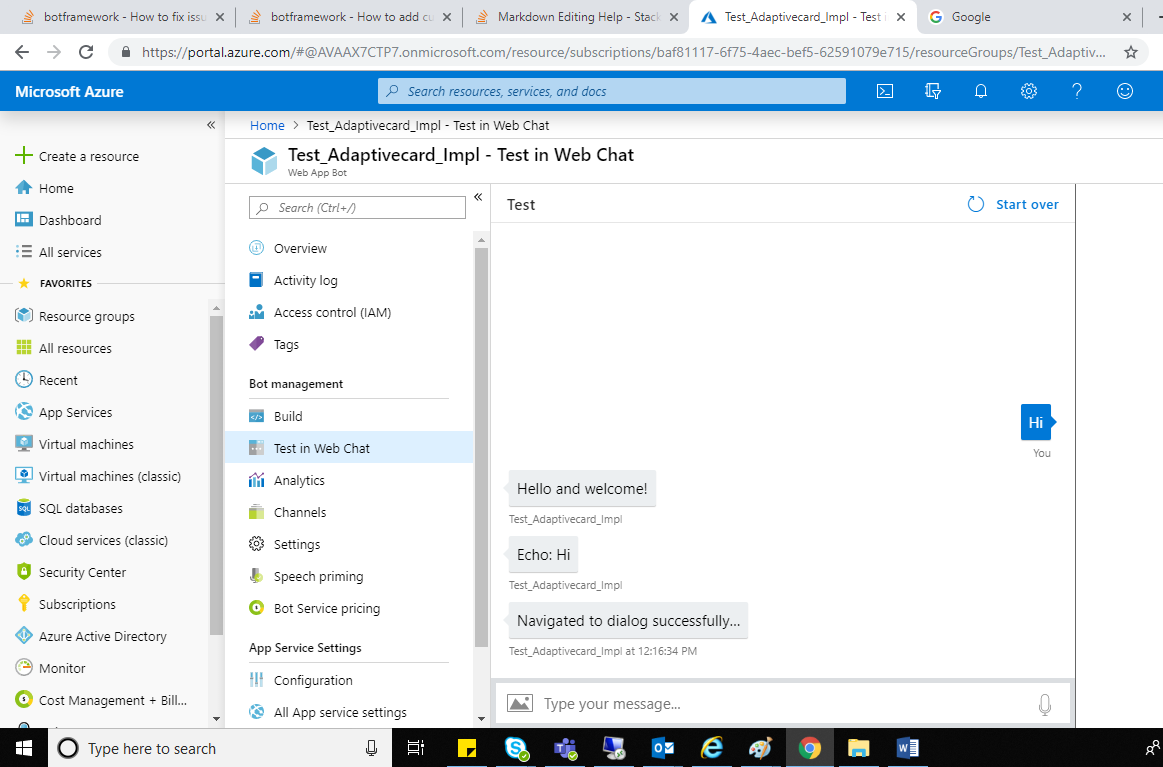
**Tested in Emulator Successfully:**



**Bot published through Visual Studio 2019 successfully without any errors or warnings:**



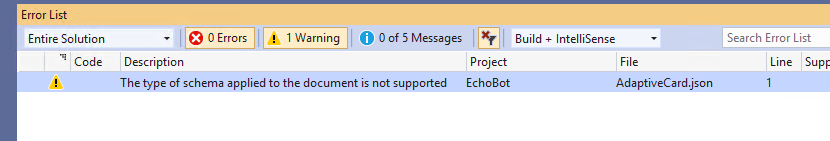
**Tested in azure test in webchat successfully and same result or emulator is displayed:**



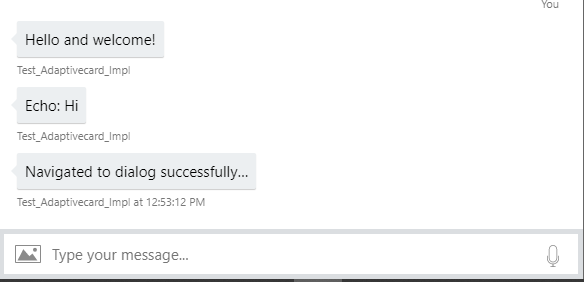
**Adding Adaptive card file and publishing again but not adding any functionality just creating a adaptive card JSON file adding it to solution and publishing it which is successful with just 1 warning on the json file added for adaptive card.**

*1 warning: The type of schema applied to the document is not supported*

*I think this can be ignored as this is given by the visualizer itself*



**Tested in azure test in webchat successfully and same result as 1st test is displayed i.e. Test passed:**



Now adding code to the Dialog class the display of adaptive card which only displays the card does nothing and then again publishing it and then view the o/p in Test in Webchat and Emulator:

**Added the following code in dialog:**

AddStep(async (stepContext, cancellationToken) =>

{ const string cards = @"./AdaptiveCard.json";

await stepContext.Context.SendActivityAsync(MessageFactory.Text($"Navigated to dialog successfully..."));

var cardAttachment = CreateAdaptiveCardAttachment(cards);

var reply = stepContext.Context.Activity.CreateReply();

reply.Attachments = new List<Attachment>() { cardAttachment };

await stepContext.Context.SendActivityAsync(reply, cancellationToken);

return await stepContext.EndDialogAsync();

});

public static Attachment CreateAdaptiveCardAttachment(string filePath)

{

var adaptiveCardJson = File.ReadAllText(filePath);

var adaptiveCardAttachment = new Attachment()

{

ContentType = "application/vnd.microsoft.card.adaptive",

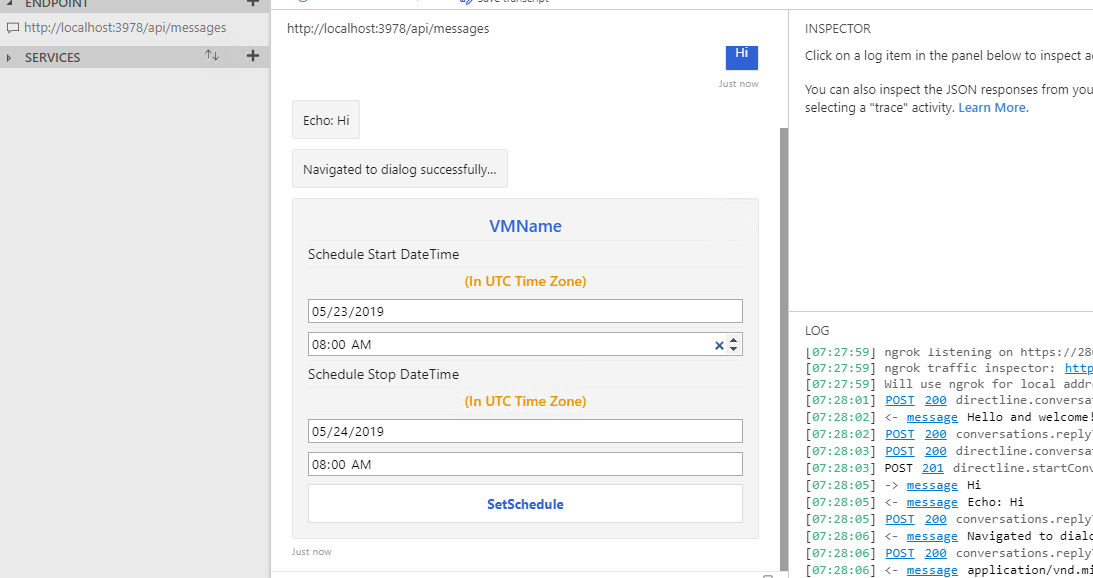
Content = JsonConvert.DeserializeObject(adaptiveCardJson),

};

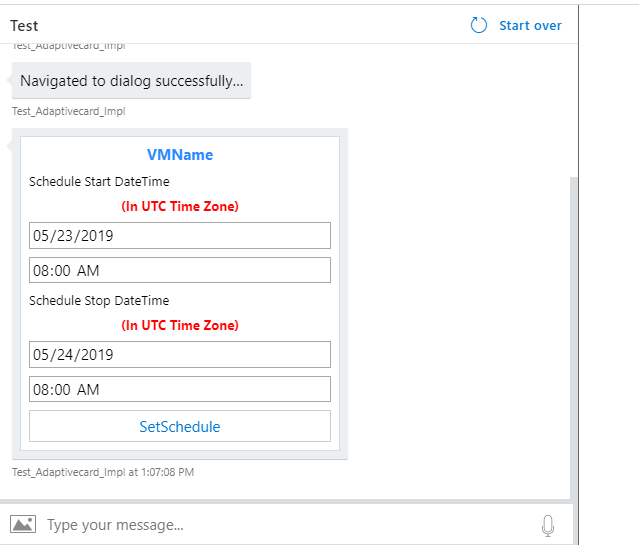
return adaptiveCardAttachment;

}

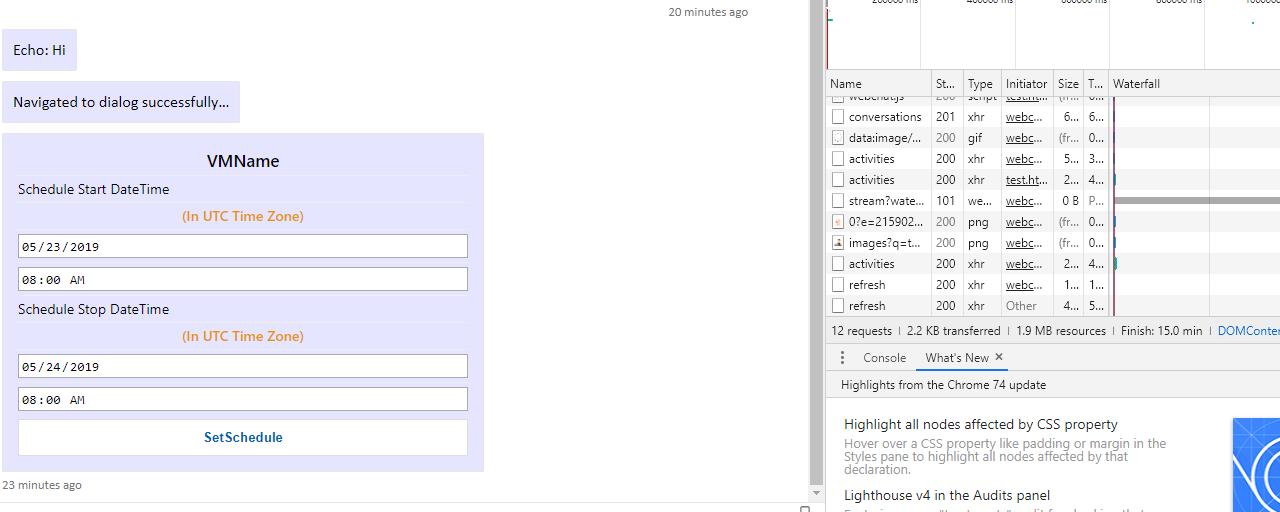
**Card displayed successfully in the Emulator:**



Displayed properly in the TestINwebchatalso:



The adaptive card displayed in the HTML page also without any errors:



Enabling the code to extract the inputs and process it in the dialog class next step of water fall code given above by adding following code:

Code in Dialog Extension class:

if (turnContext.Activity.Type == ActivityTypes.Message)

{

// Ensure that message is a postBack (like a submission from Adaptive Cards)

if (dialogContext.Context.Activity.GetType().GetProperty("ChannelData") != null)

{

var channelData = JObject.Parse(dialogContext.Context.Activity.ChannelData.ToString());

if (channelData.ContainsKey("postBack"))

{

var postbackActivity = dialogContext.Context.Activity;

// Convert the user's Adaptive Card input into the input of a Text Prompt

// Must be sent as a string

postbackActivity.Text = postbackActivity.Value.ToString();

await dialogContext.Context.SendActivityAsync(postbackActivity);

}

}

}

And then another step inside dialog class to process the inputs provided in the adaptive card:

AddStep(async (stepContext, cancellationToken) =>

{

var res = stepContext.Result.ToString();

dynamic jobject = JsonConvert.DeserializeObject(res);

string NewStartDateTime = jobject.Startdate + " " + jobject.Starttime;

string NewStopDateTime = jobject.Stopdate + " " + jobject.Stoptime;

CultureInfo provider = CultureInfo.InvariantCulture;

DateTime dt = DateTime.Parse(NewStartDateTime);

NewStartDateTime = dt.ToString("yyyy-MM-dd HH:mm");

DateTime StartDateTime = DateTime.ParseExact(NewStartDateTime, "yyyy-MM-dd HH:mm", provider);

dt = DateTime.Parse(NewStopDateTime);

NewStopDateTime = dt.ToString("yyyy-MM-dd HH:mm");

DateTime StopDateTime = DateTime.ParseExact(NewStopDateTime, "yyyy-MM-dd HH:mm", provider);

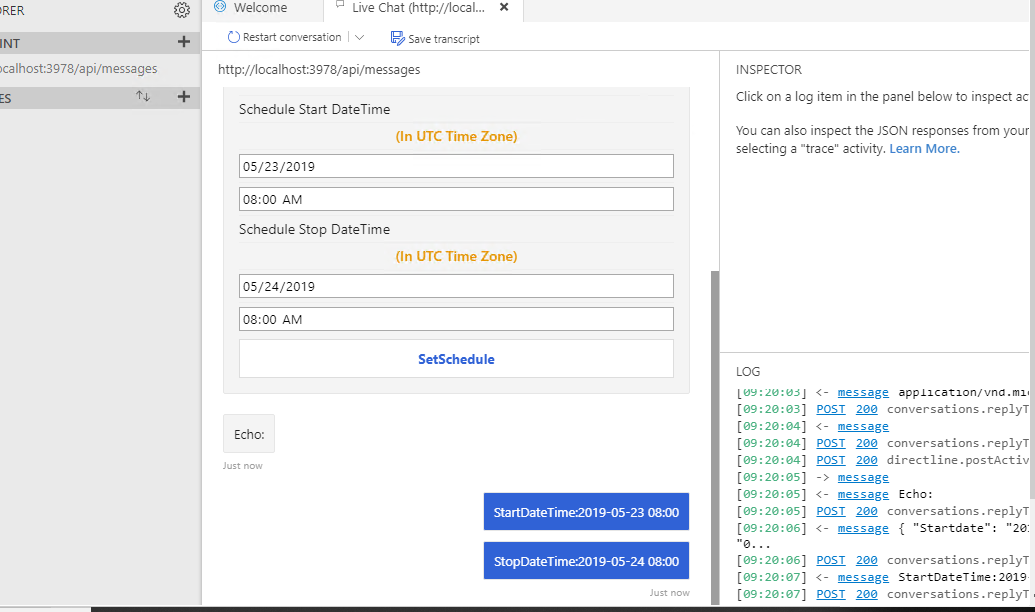
await stepContext.Context.SendActivityAsync($"StartDateTime:{NewStartDateTime}", cancellationToken: cancellationToken);

await stepContext.Context.SendActivityAsync($"StopDateTime:{NewStopDateTime}", cancellationToken: cancellationToken);

return await stepContext.EndDialogAsync();

});

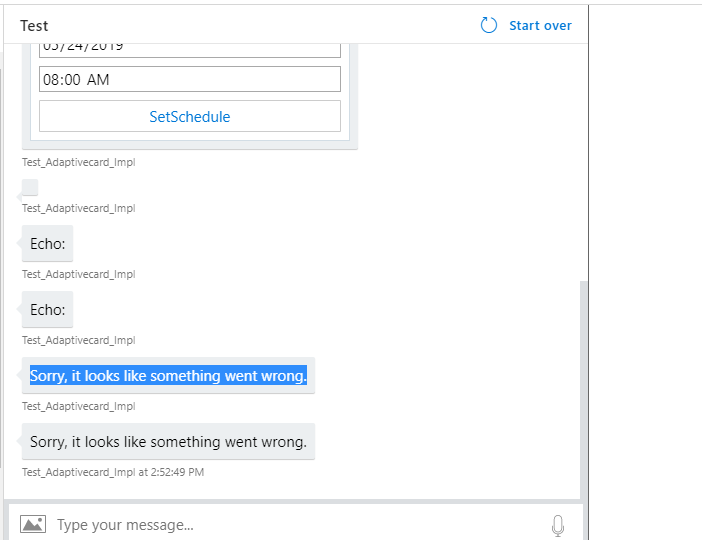
Emulator working fine, printed out what I have given input:



**Publishing it to azure testing in webchat bot:**

No errors in publishing as usual.

**Error in Test in webchat:**



**And also error in HTML web channel chat bot, this error is same as explained in the original/edited post:**

