PREFACE

This took a lot more than 4 hours and I hope that the code does reflect that.
I've created the API, SQL express tables and ORM to interact with the data, script and instructions to create the DB and finally a Unit test project.

API ENDPOINTS

1. **Submit Feedback**

HTTPPOST ../api/Feedback/submit

Sample input:

```
{
    PlayerID : 3,
    GameSessionID:3,
    FeedbackScore :2,
    FeedbackComment: "Another Test"
}
```

Parameters:

a. PlayerID (*Mandatory field*) - specific player submitting feedback.

Look at the seed data section for the id.

In a full blown implementation this would be the logged in user.

b. GameSessionID (*Mandatory field*) - game session for which this feedback submission is being made.

Look at the seed data section for the id.

In a full blown implementation this would be from a filtered selection of all sessions that the logged in user is a part of.

c. FeedbackScore (*Mandatory field*) – 1-5 rating for the session.

A range check for the FeedbackScore is in place

d. FeedbackComment (*Optional field*) – Text accompanying the submission.

There is a 500 char limit on this field.

***NOTE**: If a specific combination of playerId and gamesessionId attempts to submit multiple entries, an error is generated.*

Sample output:

Body Cookies Headers (7) Test Results Status: 201 Created Time: 152 ms

Pretty Raw Preview JSON


```
{
    "feedback": {
        "feedbackID": 1004,
        "playerID": 3,
        "gameSessionID": 3,
        "feedbackScore": 2,
        "feedbackComment": "Another Test",
        "submissionDate": "2019-09-17T12:31:49.6945881-04:00"
    },
    "gamer": {
        "playerID": 3,
        "screenName": "Watcher"
    },
    "gameSession": {
        "gameSessionID": 3,
        "gameID": 1,
        "sessionStartTime": "2019-08-12T11:08:52.468",
        "sessionEndTime": "2019-08-14T13:08:52.468"
    },
    "game": {
        "gameID": 1,
        "gameName": "The Witcher",
        "releaseYear": 2017,
        "publisher": "CD Project"
    }
}
```

ASSUMPTION:

a. The requirements state "*allows players of an online game to submit feedback for their last game session*,". From the data side or API side, I've not put in a restriction on which players can submit feedback for a specific game session, I'm relying on the front end to filter out and only show relevant game sessions to each user.
Thus, the responsibility to only show relevant game sessions is external to this API, however, for data sanity it would also make sense to consider having the API query/validate this information, but this is something that I've not implemented.

2. View All Submitted Feedback (with optional filtering, paging and next/previous page links)

HTTPGET ../api/Feedback/view?FeedbackScore=3&PageNumber=1&PageSize=2

Note: The output is sorted by feedback submission time descending, i.e. the most recent one appears first.

Query-string Parameters:

a. FeedbackScore (*optional*) – to filter out submissions. *If this parameter is missing, all feedback entries are returned.*

b. PageNumber (*optional*) – paging has been implemented and this returns results on the specified page.

c. PageSize (*optional*) – the number of feedback entries to display on a page, *the default is set to 3 and the max is set to 20*.

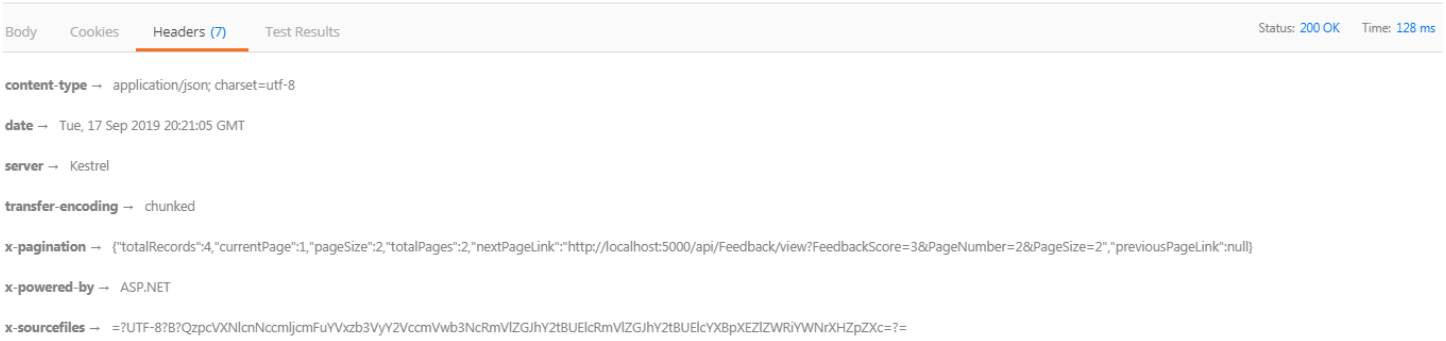
Sample output:

ody Cookies Headers (7) Test Results Status: 200 OK

Pretty Raw Preview JSON


```
[
    {
        "feedback": {
            "feedbackID": 4,
            "playerID": 2,
            "gameSessionID": 1,
            "feedbackScore": 3,
            "feedbackComment": "This is a test 3",
            "submissionDate": "2019-09-15T11:08:39.894"
        },
        "gamer": {
            "playerID": 2,
            "screenName": "Marcus"
        },
        "gameSession": {
            "gameSessionID": 1,
            "gameID": 1,
            "sessionStartTime": "2019-09-01T07:08:52.468",
            "sessionEndTime": "2019-09-01T12:08:52.468"
        },
        "game": {
            "gameID": 1,
            "gameName": "The Witcher",
            "releaseYear": 2017,
            "publisher": "CD Project"
        }
    },
    {
        "feedback": {
            "feedbackID": 3,
            "playerID": 2,
            "gameSessionID": 4,
            "feedbackScore": 3,
            "feedbackComment": "This is a test 2",
            "submissionDate": "2019-09-15T11:08:28.329"
        },
        "gamer": {
            "playerID": 2,
            "screenName": "Marcus"
        },
        "gameSession": {
            "gameSessionID": 4,
            "gameID": 2,
            "sessionStartTime": "2019-07-08T11:08:52.468",
            "sessionEndTime": "2019-07-08T22:28:52.468"
        },
        "game": {
            "gameID": 2,
            "gameName": "Gears Of War",
            "releaseYear": 2009,
            "publisher": "Microsoft"
        }
    }
]
```

Custom pagination header (x-pagination) for previous/next page of results



3. View a Specific Feedback Entry

HTTPGET ../api/Feedback/1004*

*1004 is the feedbackID of a specific Feedback entry in the SQL DB.

Sample output:

SEED DATA IN THE SQL EXPRESS DB

This is the master data that is used in the application and is set up when the DB is created, instructions in the final section.

Data for Players -

PlayerID	ScreenName
1	Geralt
2	Marcus
3	Watcher
4	Bradford

Data for GameSessions -

GameSessionID	GameID	SessionStartTime	SessionEndTime
1	1	9/1/2019 7:08:52 AM	9/1/2019 12:08:52 PM
2	1	8/22/2019 12:08:52 PM	8/23/2019 12:08:52 PM
3	1	8/12/2019 11:08:52 AM	8/14/2019 1:08:52 PM
4	2	7/8/2019 11:08:52 AM	7/8/2019 10:28:52 PM
5	2	7/13/2019 7:28:52 AM	7/13/2019 1:27:52 PM
6	3	9/8/2019 2:13:52 AM	9/8/2019 3:08:52 PM

Data for Games -

GameID	GameName	ReleaseYear	Publisher
1	The Witcher	2017	CD Project
2	Gears Of War	2009	Microsoft
3	Pillars Of Eternity	2015	Paradox
4	XCOM2	2017	2K

CODE SPECIFICS AND SET UP INSTRUCTIONS

This is a .Net Core C# API solution developed in Visual studio 2017.

a. The solution has 2 projects – FeedbackAPI, the .Net core project and FeedbackAPI.Tests, the Unit Testing project.
b. The project uses SQL express (which comes with a Visual Studio installation) to store data.
   In order to create the tables, please follow these steps –
   i. Open the solution in VS 2017.
   ii. In the VS 2017 menu, click on View -> Other Windows -> Package Manager Console.
   iii. When the console is ready, type “Update-Database” and hit Enter. This will create the tables in a DB called FeedbackDb on your sql express local db – localdb\mssqllocaldb

```
PM> Update-database
```

   Once this operation completes, the Console will say Done and will be ready for new commands.
   iv. The file \FeedbackAPI\FeedbackAPI\Migrations\ 20190911160853_initial.cs is the schema for the tables along with the seed data.
c. Hit Ctrl+F5, which will build the solution and then start the Kestrel Server which will respond to endpoint requests.
d. Use a tool like Postman to test the application. By default, the application is set to respond at http://localhost:5000/api/feedback
e. The endpoints are set up to accept and produce JSON in asynchronous operations.
f. Software packages used:
   i. Standard .Net Core 2.0 and associated packages.
   ii. Entity Framework Core (ORM)
   iii. Xunit and Moq for Unit Testing.