2024 Meridian Capstone

Project Supervisor: Juan Robledo

Team: Oscar Vazquez and Mary Kait Heeren

Date: May 28th, 2024 – July 31st, 2024

Table of Contents

1 Introduction

1.1 Objectives

This document aims to guide the process/development of our web application, highlighting each major feature and ensuring that the web application performs reliably. Through this section, we intend to identify and break down multiple integrated features, enhancing its overall quality before release.

1.2 Background

Our web application is a dynamic platform for recommending unique shows, viewing/creating dashboards to visualize progress, and providing details of a desired TV show such as Name, summary, image, etc. It supports a variety of user interactions, including adding, viewing, editing, and deleting recommended shows, managing dashboards for performance analysis, and monitoring state (currently watching, not watching, not finished, finished) and urgency (low, medium, high) of TV shows. Designed with user-friendliness, the app aims to facilitate an efficient system for recommending shows.

1.3 Purpose

To create an advanced, interactive show recommendation system with multiple integrated features including virtual agent, flow design, scripting, performance analytics, REST API integration, service portal, and user feedback mechanisms.

1.4 Scope

The scope of this document is confined to the web application. It covers an extensive evaluation of the user interface, backend operations, integration with external services, and performance on desktop devices. The testing will scrutinize all primary functionalities outlined to ensure they meet our usability and performance standards.

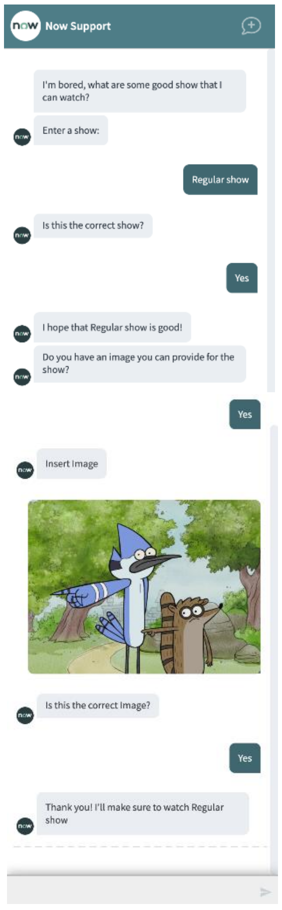
2 Tasks and Processes

2.1 Virtual Agent

The first task accomplished was creating a Virtual Agent that asks for show recommendations, provides an option to add a photo, and creates a record in the "Shows" table with the show name and photo. To achieve this, the steps are as follows:

1. **Navigated to the Virtual Agent Designer**: Accessed the Virtual Agent Designer through the ServiceNow interface to begin creating the new Virtual Agent.
2. **Created a New Virtual Agent Topic**: Initiated a new topic named "Show Recommendations" to handle user inputs for recommending shows.
3. **Designed the Conversation Flow for the Virtual Agent**: Developed the conversation flow to include a friendly welcome message, prompts for show recommendations, and an option to upload a photo. Configured input fields to capture the show name and photo.
4. **Tested the Virtual Agent**: Ran thorough tests on the Virtual Agent by simulating all possible user interactions and variations of the flowchart to ensure all scenarios were handled correctly and the data was accurately stored in the "Shows" table.

This process ensured that the Virtual Agent operated smoothly, capturing user inputs effectively and storing them in the designated table. A visual of the Virtual Agent is displayed above in **Figure 1**.



I'm bored, what are some good show that I can watch?

Enter a show:

Regular show

Is this the correct show?

Yes

I hope that Regular show is good!

Do you have an image you can provide for the show?

Yes

Insert Image

Is this the correct Image?

Yes

Thank you! I'll make sure to watch Regular show

Figure 1: Virtual Agent

2.2 Flow Designer

The second task involves utilizing Flow Designer to manage duplicate show recommendations and set a Due Date for each show. The steps are as follows:

1. **Create a New Field**: Added a "Due Date" field to the "Shows" table.
2. **Configure Flow Designer**: Using Flow Designer, set up a flow to automatically set the "Due Date" to 20 days after the current date or the record's creation date.
3. **Check for Duplicate Recommendations**: Incorporated a condition within the flow to verify if a similar show has already been recommended. If a duplicate is identified, the flow will delete the new record instead of the existing one.

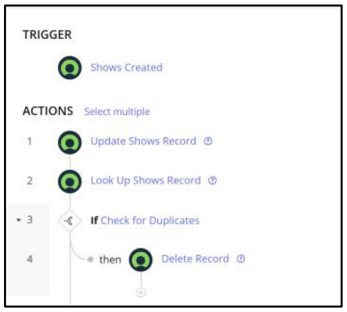


Figure 2: Flow Designer

These steps ensure the efficient management of due dates and prevent duplicate entries in the "Shows" table.

2.3 Scripting

The third task involves adding new columns to the "Shows" table, creating a business rule to improve duplicate checking, and implementing scheduled scripts to manage urgency based on due dates. The steps are as follows:

1. **Create a Business Rule**: Implement a business rule to check for duplicates more effectively (e.g., "OnePiece" and "One Piece"). If duplicates are found, delete the record that is about to be added.
2. **Update the Table**: Add two new columns to the "Shows" table:
   - **State**: Options include "Not Started," "Paused Watching," "Currently Watching," and "Finished Watching."
   - **Urgency**: Options include "Low," "Medium," and "High."
3. **Create a Scheduled Script for Past-Due Records**: Develop a scheduled script that checks for any past-due records, not in the "Finished Watching" state. If any are found, trigger an event to increase the urgency to "High."
4. **Create a Scheduled Script for Upcoming Due Dates**: Develop another scheduled script that checks records within a week of their due date. If found, trigger an event to adjust the urgency to "Medium."

These steps ensure duplicate recommendations are effectively managed and urgency levels are dynamically adjusted based on due dates.

2.4 Performance Analytics

The fourth task involves creating a report source to track all shows that are not in the "Finished Watching" state, along with their urgency levels and due dates. The steps are as follows:

1. **Create a Report Source**: Configure the report source to filter and include only shows that are not in the "Finished Watching" state.
2. **Track Show Status**: Define how many shows are being added, how many are currently being watched, and how many have been finished or not started.

3. **Track Urgency Levels**: Configure additional metrics to track the urgency levels of the shows: "Low," "Medium," and "High."
4. **Track Due Dates**: Set up a mechanism to track how many shows are due within a specified period. Include filters and conditions to ensure accurate tracking of due dates.

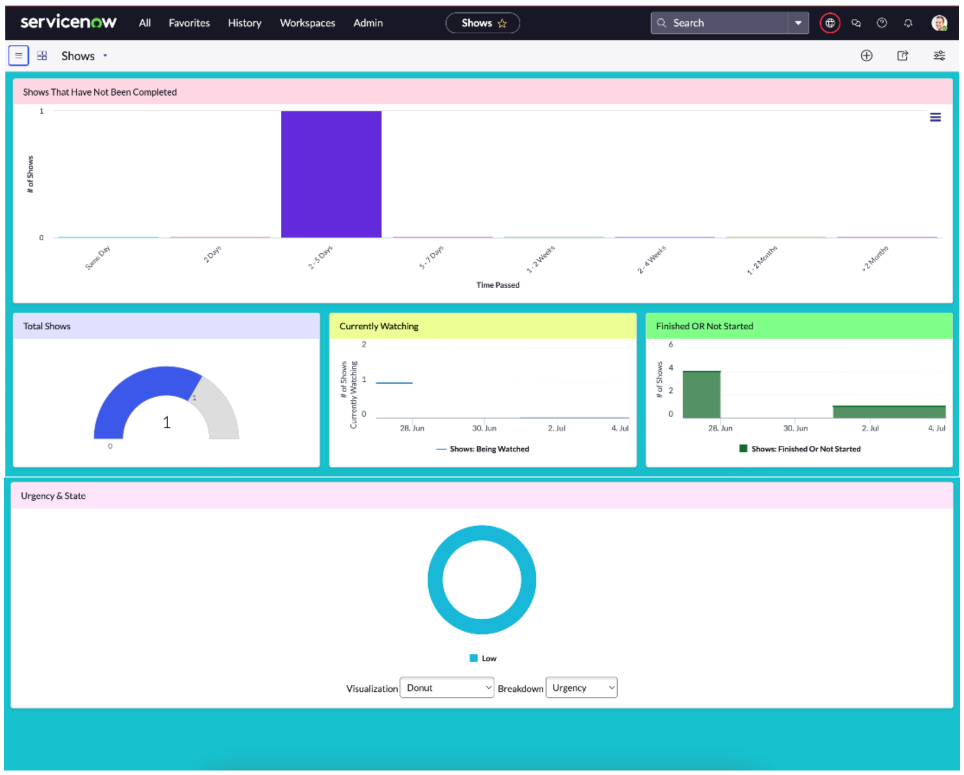


Figure 3: Performance Analytics Dashboard

By following these steps, you can create a comprehensive dashboard that tracks the status and urgency of shows, monitors due dates, and provides valuable insights

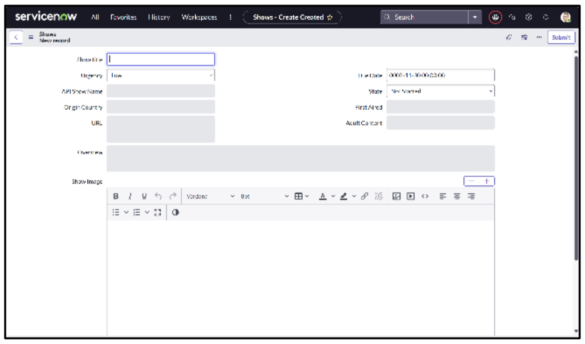
2.5 REST Application Programming Interface (API)

Task 5 involves integrating data from themoviedb.org into ServiceNow by creating seven new columns on the "Shows" table. These columns include "Overview," "API Show Name," "Origin Country," "Adult Content," "URL," "First Aired," and "Show Image." The steps are as follows **:**

1. **Create New Columns on the Shows Table**: Add the following columns to the "Shows" table:
    1. **Overview**: Read-only field to store show descriptions.
    2. **API Show Name**: Read-only field for the show's name from the API.
    3. **Origin Country**: Read-only field for the country of origin.
    4. **Adult Content**: Read-only boolean field to indicate if the show contains adult content.
    5. **URL**: Read-only field for the URL linking to more information about the show.

6. **First Aired**: Read-only field for the first aired date of the show.
7. **Show Image**: Non-read-only field to store the show's image.

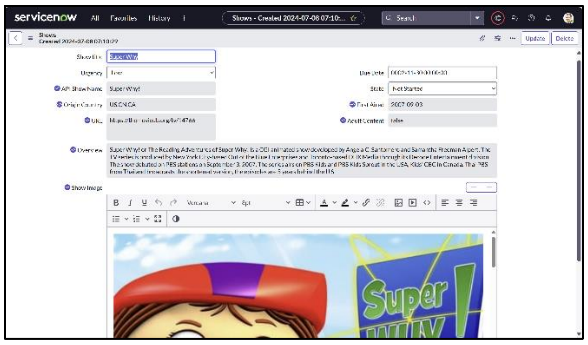
2. **Set Up REST Message**: Create a new REST message in ServiceNow to connect to themoviedb.org's API. Configure the REST message with the HTTP GET method to retrieve show information. Refer to the themoviedb.org API documentation for details on endpoints and parameters.
3. **Create Script Include**: Develop a Script Include in ServiceNow that utilizes the REST message created earlier. Implement methods in the Script Include to call the REST message and handle data retrieval from themoviedb.org.
4. **Implement Business Rule**: Create a Business Rule on the "Shows" table in ServiceNow. Configure the Business Rule to trigger insert or update actions. Use the Script Include to fetch data from themoviedb.org via the REST message and populate the newly created columns based on the show's information retrieved from the API.
5. **Testing:** Test the integration thoroughly to ensure the REST message fetches accurate data from themoviedb.org.



Shows
New record

Show title			
Urgency	Low	Due Date	0002-11-30 00:00:00
API Show Name		State	Not Started
Origin Country		First Aired	
URL		Adult Content	
Overview			
Show Image			

Figure 5:Before API Population



Shows
Created 2024-07-08 07:10:29

Show title	Super Why!		
Urgency	Low	Due Date	0002-11-30 00:00:00
API Show Name	Super Why!	State	Not Started
Origin Country	US,CN,CA	First Aired	2007-09-03
URL	https://themoviedb.org/tv/14766	Adult Content	false

Overview: Super Why! or The Reading Adventures of Super Why!, is a CGI animated show developed by Angela C. Santomero and Samantha Freeman Alpert. The TV series is produced by New York City-based Out of the Blue Enterprises and Toronto-based DHX Media through its Decode Entertainment division. The show debuted on PBS stations on September 3, 2007. The series airs on PBS Kids and PBS Kids Sprout in the USA, Kids' CBC in Canada. Thai PBS from Thailand broadcasts the shortened version, the episodes are 5 years behind the US.

Show Image

Figure 6: After API Population

By following these steps, the table effectively integrates data from themoviedb.org into ServiceNow using REST APIs, ensuring that show information is dynamically updated and available for use within your instance.

2.6 Implementing API Features into Virtual Agent

Task 6 involves enhancing the Virtual Agent to display show information when a user recommends a show. The steps are as follows:

1. **Enhance Virtual Agent Interaction**: Modify the Virtual Agent topic related to recommending shows to include displaying show information after receiving a recommendation from the user.
2. **Integration with API Data**: Utilize the integration with themoviedb.org API (already set up in Task 5) to fetch show information such as overview, first aired date, origin country, etc.
3. **Display Information**: Update the conversation flow in the Virtual Agent to fetch and display relevant show information retrieved from the API in response to the user's recommendation.
4. **Testing:** Test the Virtual Agent thoroughly to ensure that it accurately displays information when a recommendation is made.

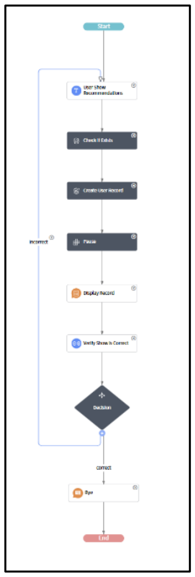
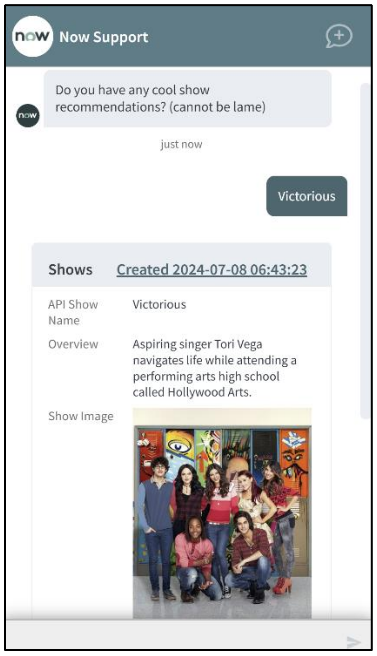


Figure 6: VA Conversation Flow



Do you have any cool show recommendations? (cannot be lame)

just now

Victorious

Shows Created 2024-07-08 06:43:23

API Show Name: Victorious

Overview: Aspiring singer Tori Vega navigates life while attending a performing arts high school called Hollywood Arts.

Show Image

Figure 7: Updated Virtual Agent

By following these steps, you can enhance the Virtual Agent to dynamically display show information when users recommend shows, integrating real-time data from themoviedb.org to enrich the user interaction experience.

2.7 Recap, Documentation Overview, and Verdict on Personal Developer Instance (PDI)

This task involved documenting all implemented features, selecting a PDI for the final tasks, and setting up new admin users. The steps are as follows:

1. **Document All Implemented Features**: Review and document every feature and enhancement implemented so far.
2. **Review and Compare Individual PDIs**: Discuss and compare the PDIs to determine which instance is most suitable for continuing the final tasks.
3. **Select a PDI**: Based on the review and discussion, choose one PDI to be used for the remaining tasks.
4. **Create New Users with Admin Privileges**: In the selected PDI, create new user accounts for each team member.

Now, we have created a unified PDI for final development and a secure setup with individual admin accounts for each team member.

2.8 Service Portal

This task involved creating a new service portal and customizing it using HTML and CSS. This portal will display most of the features implemented in the previous tasks, including the Virtual Agent, Show Table with records, and Dashboard. The major task is to design the portal using principles from Client Server, with creative freedom for the name and design. This portal will be the foundation of our overall project, and any new features will be add-ons to this task. The steps are as follows:

1. **Add Containers and Determine Layout:** Decided what kind of layout to design as well as added containers to hold widgets on the homepage.
2. **Add Virtual Agent Widget:** Incorporated the Virtual Agent service portal widget to allow access to the Virtual Agent on the portal.
3. **Show Table Integration:** Added the Shows table using a Data Table from Instance Widget to display the records of shows.
4. **Dashboard Widget Creation:** Created a new widget to display the dashboard and all the indicators it includes.
5. **Carousel Widget Development:** Developed a widget named "Carousel" due to its transitioning behavior. This widget allows users to transition through all the shows in a specific state before moving to the next state, thus displaying all the shows by the end of its rotation.
6. **Welcome Page Widget Development:** Developed a widget named “test_welcome” that displays a personalized greeting for the logged in user. The widget also contains a button that redirects the logged in user to their Shows Homepage with the carousel widget, table data, and dashboard.

The newly created service portal successfully integrates the Virtual Agent, Shows table, Dashboard, and Carousel widgets, providing a comprehensive and interactive user experience. This portal serves as the foundation for future enhancements and feature additions.

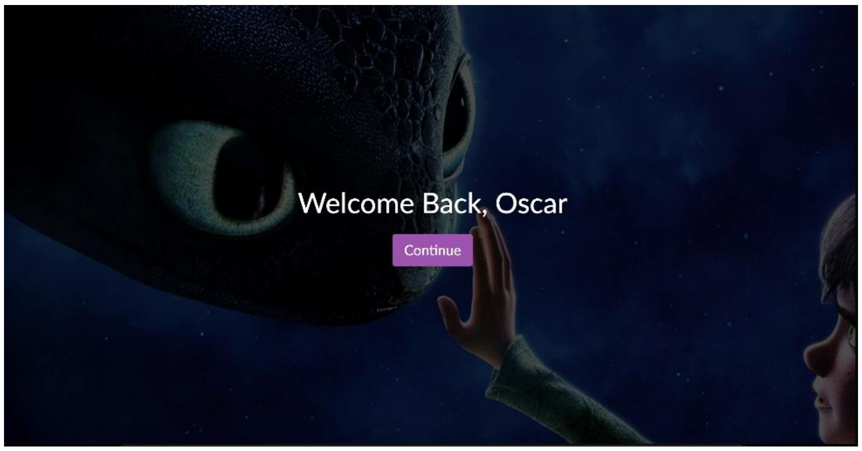
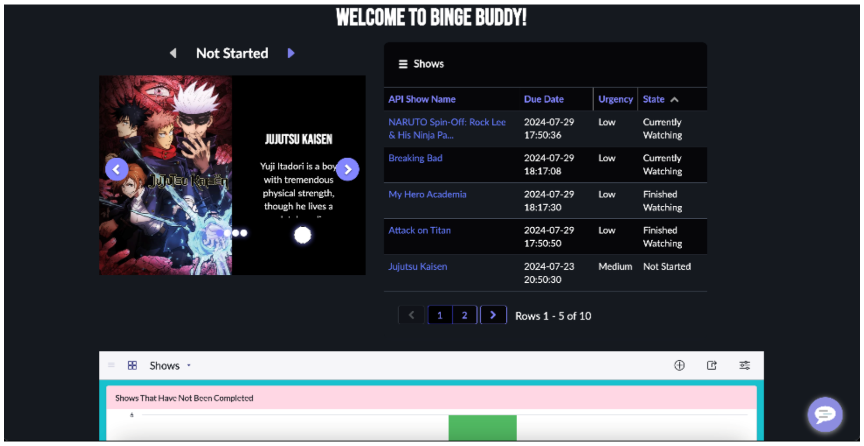


Figure 8: Welcome Page for Service Portal


WELCOME TO BINGE BUDDY!
Not Started
JUJUTSU KAISEN
Yuji Itadori is a boy with tremendous physical strength, though he lives a
Shows
Rows 1 - 5 of 10
Shows That Have Not Been Completed

API Show Name	Due Date	Urgency	State
NARUTO Spin-Off: Rock Lee & His Ninja Pa...	2024-07-29 17:50:36	Low	Currently Watching
Breaking Bad	2024-07-29 18:17:08	Low	Currently Watching
My Hero Academia	2024-07-29 18:17:30	Low	Finished Watching
Attack on Titan	2024-07-29 17:50:50	Low	Finished Watching
Jujutsu Kaisen	2024-07-23 20:50:30	Medium	Not Started

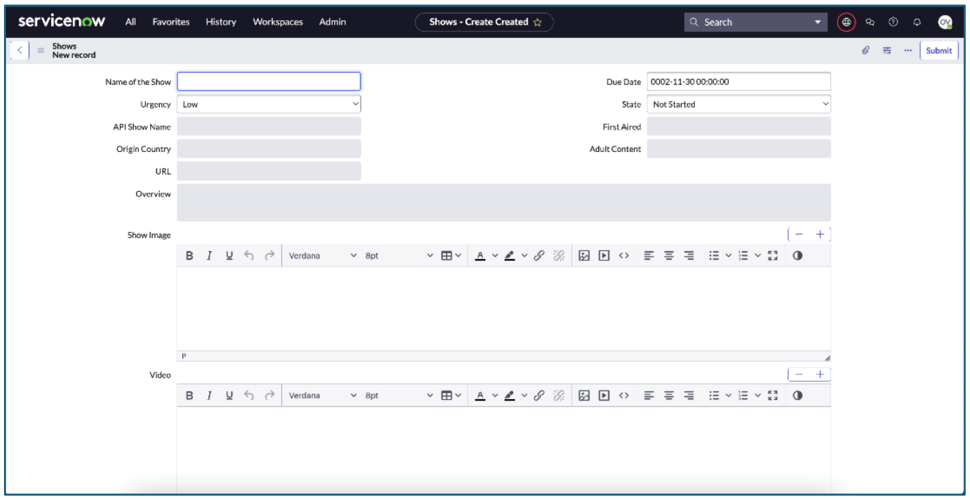
Figure 9: Service Portal Homepage

2.9 Add API Video Functionality

Task 9 involves integrating data from **Youtube.com** into ServiceNow by creating one new column on the "Shows" table. This new column is "Video." The steps are as follows:

1. **Create a new Column on the Shows Table: Add the following column to the "Shows" table:**
    1. **Video**: Non-read-only field to store the show's trailers.
2. **Set Up REST Message:** Create a new REST message in ServiceNow to connect to Youtube.com's API. Configure the REST message with the HTTP GET method to retrieve show trailers. Refer to the Youtube.com. API documentation for details on endpoints and parameters.
3. **Create Script Include:** Develop a Script Include in ServiceNow that utilizes the REST message created earlier. Implement methods in the Script Include to call the REST message and handle data retrieval from Youtube.com

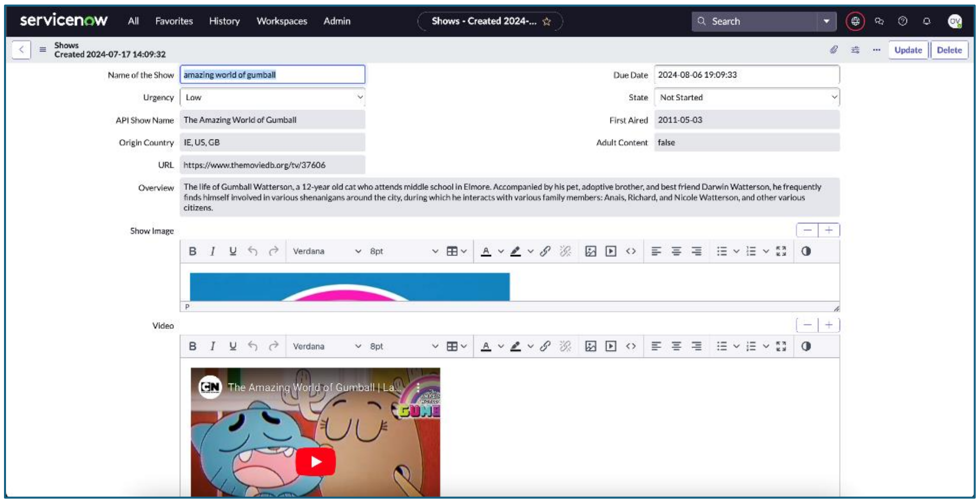
4. **Implement Business Rule:** Create a Business Rule on the "Shows" table in ServiceNow. Configure the Business Rule to trigger insert or update actions. Use the Script Include to fetch data from Youtube.com via the REST message and populate the newly created column based on the show's information retrieved from the API.
5. **Testing:** Test the integration thoroughly to ensure the REST message fetches accurate data from Youtube.com.



Shows
New record

Name of the Show	
Urgency	Low
API Show Name	
Origin Country	
URL	
Overview	
Due Date	0002-11-30 00:00:00
State	Not Started
First Aired	
Adult Content	

Figure 10: Before API Population (Video Column)



Shows
Created 2024-07-17 14:09:32

Name of the Show	amazing world of gumball
Urgency	Low
API Show Name	The Amazing World of Gumball
Origin Country	IE, US, GB
URL	https://www.themoviedb.org/tv/37606
Overview	The life of Gumball Watterson, a 12-year old cat who attends middle school in Elmore. Accompanied by his pet, adoptive brother, and best friend Darwin Watterson, he frequently finds himself involved in various shenanigans around the city, during which he interacts with various family members: Anais, Richard, and Nicole Watterson, and other various citizens.
Due Date	2024-08-06 19:09:33
State	Not Started
First Aired	2011-05-03
Adult Content	false

Figure 11: After API Population (Video Column)

By following these steps, the table effectively integrates data from Youtube.com into ServiceNow using REST APIs, ensuring that the show trailer is dynamically updated and available for use within your instance.

2.10 System Tests

2.10.1 Functional Requirement: Add Shows

Description: Users shall be able to recommend shows with details such as entering the Name of the Show and setting the Urgency and State of the show they are creating.

Linked Test Case:

- **Test ID:** AddShow-TestCase
- **Test Description:** This test case verifies that users can successfully add a new show recommendation by entering the name of the show, setting the urgency level, and selecting the state. The test will check that the "Name of the Show" field is correctly filled out, with the Urgency and State fields having set default values of "Low" and "Not Started" respectively if the user chooses not to change those two fields. The TMDB API will then populate the "Overview," "API Show Name," "Origin Country," "Adult Content," "URL," "First Aired," and "Show Image". The YouTube API will then populate the "Video" field with the first 5 trailers of the added show. The new show is saved in the system and will appear in the list of recommendations with these details.

2.10.2 Functional Requirement: View Shows

Description: Users shall be able to view their shows within the "Shows" table.

Linked Test Case:

- **Test ID:** ViewShows-TestCase
- **Test Description:** This test case verifies that users can successfully view their shows within the "Shows" table. The test will ensure that users can access the "Shows" table and see a list of all shows they have recommended.

2.10.3 Functional Requirement: Delete Shows

Description: Users shall be able to delete shows within the "Shows" table.

Linked Test Case:

- **Test ID:** DeleteShow-TestCase
- **Test Description:** This test case verifies that users can successfully delete shows from the "Shows" table. The test will ensure that users can select a show and delete it, and that the show is permanently removed from the system.

2.10.4 Functional Requirement: Update Shows

Description: Users shall be able to update the details of a show already recommended. Users will be able to update the “Name of the Show”, “Urgency”, “State”, and “Due Date” fields.

Linked Test Case:

- **Test ID:** UpdateShows-TestCase
- **Test Description:** This test case verifies that users can successfully update the details of an already recommended show. The test will check that users can modify the “Name of the Show”, “Urgency”, “State”, and “Due Date” fields for an existing show and save the changes. It will also ensure that the updated details are correctly reflected in the system.

2.10.5 Functional Requirement: Check for Duplicate Shows

Description: System shall be able to find and prevent the recommendation of shows that have already been recommended (shows that are already in the “Shows” table).

Linked Test Case:

- **Test ID:** DuplicateShow-TestCase
- **Test Description:** This test case verifies that the system (business rule) can identify and prevent duplicate show recommendations. The test will attempt to add a show that already exists in the "Shows" table and confirm that the system correctly identifies the duplicate and prevents the addition.

2.10.6 Functional Requirement: Update Due Soon Shows

Description: System shall be able to find shows where the “Due Date” field is within a week of its assigned due date and increase their “Urgency” level by one (from “Low” to “Medium” and “Medium” to “High”).

Linked Test Case:

- **Test ID:** UpdateDueSoonShows-TestCase
- **Test Description:** This test case verifies that the system (scheduled script execution, event registry, and business rule) correctly identifies shows with a “Due Date” that has passed and updates their “Urgency” level accordingly. The test will set up shows with past due dates, run the system process to check for overdue shows, and confirm that the urgency levels are increased as specified (from low to medium and from medium to high). It will also verify that shows with urgency already set to high are not further updated.

2.10.7 Functional Requirement: Update Overdue Shows

Description: System shall be able to find shows where the “Due Date” field is past its assigned due date and increase their “Urgency” level to “High”.

Linked Test Case:

- **Test ID:** UpdateOverdueShows-TestCase
- **Test Description:** This test case verifies that the system (scheduled script execution, event registry, and business rule) correctly identifies shows with a "Due Date" that has passed and updates their "Urgency" level to "High". The test will set up shows with past due dates, run the system process to check for overdue shows, and confirm that the urgency levels are set to "High" for each overdue show. It will also ensure that shows with future due dates are not affected.

TEST CASE NAME/ID:	AddShow-TestCase
ENTRY CONDITION:	User is logged in and navigated to the "Shows" table. Then, the user clicks the "New" button on the top right and is redirected to a Show Form.
FLOW OF EVENTS:	1. User navigates to the "Shows" page. 2. User clicks the "New" button. 3. User enters "amazing world of gumball" in the "Name of the Show" field. 4. User does not change the Urgency nor State (default to "Low" and "Not Started" when saved). 5. User saves the show. 6. TMDB API populates additional fields: "The life of Gumball Watterson, a 12-year-old cat who attends middle school in Elmore. Accompanied by his pet, adoptive brother, and best friend Darwin Watterson, he frequently finds himself involved in various shenanigans around the city, during which he interacts with various family members: Anais, Richard, and Nicole Watterson, and other various citizens." in the "Overview" field, "The Amazing World of Gumball " in the "API Show Name" field, "IE, US, GB" in the "Origin Country" field, "false" in the "Adult Content" field, https://www.themoviedb.org/tv/37606 in the "URL" field, "2011-05-03" in the "First Aired" field and a poster image in the "Show Image" field. 7. YouTube API populates the "Video" field with the first 5 trailers or relevant videos.
EXIT CONDITION:	The new show is saved and all information from APIs populate all columns.

TEST CASE NAME/ID:	ViewShows-TestCase
ENTRY CONDITION:	User is logged in and has recommended at least one show.
FLOW OF EVENTS:	1. User navigates to the "Shows" table page. 2. User sees “amazing world of gumball” in the “Shows” table. 3. Optionally, add details of the show as columns to the form in order to display to the user.
EXIT CONDITION:	The shows are displayed accurately in the table.

TEST CASE NAME/ID:	UpdateShows-TestCase
ENTRY CONDITION:	User is logged in and has recommended at least one show.
FLOW OF EVENTS:	1. User navigates to the "Shows" table page. 2. User selects “amazing world of gumball” show to update. 3. User modifies the “Name of the Show”, “Urgency”, “State”, and “Due Date” fields to “Amazing World Of Gumball”, “Medium”, “Currently Watching” and “2024-07-16 19:09:33” respectively. 4. User saves the changes by clicking the “Update” button. 5. The system displays the same API information as before.
EXIT CONDITION:	The details shown are updated and accurately displayed.

TEST CASE NAME/ID:	DuplicateShow-TestCase
ENTRY CONDITION:	User is logged in and has recommended at least one show.
FLOW OF EVENTS:	1. User attempts to add a show called "amazing world of gumball” in the "Shows" table. 2. System identifies that “amazing world of gumball” is a duplicate. 3. System prevents the addition of “amazing world of gumball” since “Amazing World Of Gumball” is already recommended. 4. User receives a message that says, “Unable to add amazing world of gumball since the show has already been recommended.”
EXIT CONDITION:	The duplicate show is not added to the system.

TEST CASE NAME/ID:	DeleteShow-TestCase
ENTRY CONDITION:	User is logged in and has recommended at least one show.
FLOW OF EVENTS:	1. User navigates to the "Shows" table page. 2. User selects “amazing world of gumball” to delete. 3. User clicks the “Actions on selected rows” dropdown list 4. User clicks the “Delete” option. 5. User confirms their decision to delete.
EXIT CONDITION:	The selected show is deleted and no longer appears in the list of recommendations.

TEST CASE NAME/ID:	UpdateDueSoonShows-TestCase
ENTRY CONDITION:	User has recommended shows with due dates within a week.
FLOW OF EVENTS:	1. User adds “Stranger Things” to the shows table with a “Low” Urgency field. 2. User changes the due date to 6 days after its creation. 3. System runs scheduled jobs daily at 8 a.m. and finds that “Stranger Things” is due within a week. 4. System updates “Stranger Things” urgency field from “Low” to “Medium”.
EXIT CONDITION:	The urgency levels of due soon shows are correctly updated.

TEST CASE NAME/ID:	UpdateOverdueShows-TestCase
ENTRY CONDITION:	The show table has shows with past due dates.
FLOW OF EVENTS:	1. User changes the due date of “Stranger Things” to a date in the past. 2. System runs scheduled jobs daily at 8 a.m. and finds that “Stranger Things” is past its due date. 3. System updates “Stranger Things” urgency field to “High”.
EXIT CONDITION:	The urgency levels of overdue shows are correctly updated to "High".

Works Cited

ServiceNow developers. (n.d.). ServiceNow Application Developer | ServiceNow Developers

Now learning. ServiceNow. (n.d.). ServiceNow - Now Learning

The Movie Database (TMDB). (n.d.). The Movie Database (TMDB) (themoviedb.org)

Search - TV shows. The Movie Database (TMDB). (n.d.). TV (themoviedb.org)