

# ISYS30221 Artificial Intelligence

## Coursework Documentation Template

### 1- About this submission

|                                       |  |
|---------------------------------------|--|
| Student Name                          | Awn Al Bqowr   |
| Student ID                            | N1138321   |
| Chatbot Topic                         | Movies and TV series Chatbot   |
| Tasks implemented in this submission  | <input checked="" type="checkbox"/> Task a<br><input checked="" type="checkbox"/> Task b<br><input checked="" type="checkbox"/> Task c<br><br><input checked="" type="checkbox"/> Task a extra – short title: Voice Interaction<br><input checked="" type="checkbox"/> Task b extra – short title: Fuzzy Fact Checker<br><input checked="" type="checkbox"/> Task c extra – short title: Grad-CAM Visualizer   |
| Files inventory (excluding this file) | <ul style="list-style-type: none"><li>● Python source files (e.g., main.py, similarity_bot.py, logic_reasoning.py, image_classifier.py, voice_interface.py, gradcam_utils.py, tmdb_api.py, tv_api.py, build_dataset.py)</li><li>● AIML files (e.g., mybot-basic.xml)</li><li>● Data/Configuration files (e.g., qa_pairs.csv, knowledge_base.txt, class_indices.pkl)</li><li>● Model file (e.g., my_cnn_model.h5)</li></ul>   |
| Demo video URL                        | <a href="https://youtu.be/Wdl4M2eGC3Q">https://youtu.be/Wdl4M2eGC3Q</a>  |
| Checklist                             | <input checked="" type="checkbox"/> I will submit this file separately (without compression) into DropBox<br><input checked="" type="checkbox"/> All other files are zipped and will be submitted into DropBox<br><input checked="" type="checkbox"/> The demo video is recorded as instructed, and the sharing link is inserted above<br><input checked="" type="checkbox"/> I have made sure that the demo video is shared according to the instructions, so that I allowed everybody in the university to view it.<br><input checked="" type="checkbox"/> All the sections below are populated accordingly. |

## 2- Design notes (shrink/grow as needed, add images where applicable)

|   |   |
|---|---|
| General explanations of the system and its goals  | <p>The chatbot system is designed to serve as an intelligent and interactive assistant for movie and TV series enthusiasts. Its main objective is to provide users with accurate information, reasoned responses, and visual insights through the integration of multiple AI techniques. The system is built around three core functionalities:</p> <ol style="list-style-type: none"> <li><b>1. Enhanced Conversational Capabilities:</b><br/>The chatbot uses an AIML-based response system that has been expanded with over 10 new conversation pairs to address common queries about movies and TV series, such as questions on release dates, directors, and genres. When a user query does not exactly match any predefined pattern, a fallback mechanism employing TF-IDF and cosine similarity is used to retrieve the most relevant answer from a CSV file of Q/A pairs. An additional voice interface allows users to choose between text-based input and voice commands, with the chatbot delivering audible responses.</li> <li><b>2. Rule-Based Logical Reasoning:</b><br/>The system enhances its reasoning capabilities by integrating a rule-based logical reasoning module that leverages first-order logic (FOL). A knowledge base populated with domain-specific facts and rules (e.g., "Director(Christopher_Nolan, Inception)") is used to validate and store movie-related information. Users can add new facts or query existing ones through dual-pattern inputs ("I know that X is Y" and "Check that X is Y"). Additionally, a fuzzy logic-based fact checker, provides approximate matching with confidence scores to handle uncertain or near-miss inputs.</li> <li><b>3. Image Classification:</b><br/>To extend its functionality beyond text, the chatbot incorporates an image classification module. A convolutional neural network (CNN) built with TensorFlow and Keras has been trained on movie posters, enabling the system to predict the movie associated with a given image. When users submit an image—either by file selection or via a URL—the model processes it and returns a predicted label along with a confidence score. Furthermore, a Grad-CAM visualization feature is included to overlay a heatmap on the image, highlighting the regions that most influenced the classification decision.</li> </ol> <p>Overall, the system not only meets the core requirements of handling conversations and reasoning with structured knowledge but also extends its capabilities with voice interaction, fuzzy logic, and visual interpretability. This approach ensures a richer and more engaging user experience, demonstrating the practical application of diverse AI techniques in a single integrated platform.</p> |
| The system requirements, i.e., the list of what the system should do/have from a user's perspective | <ul style="list-style-type: none"> <li>● <b>Comprehensive Conversational Interaction:</b><br/>The system must support natural, topic-specific conversations about movies and TV series. It should respond using pre-defined AIML conversation pairs for common queries (e.g., release dates, directors, genres) and seamlessly fall back on a similarity-based retrieval method when an exact match is unavailable.</li> <li>● <b>Fallback Mechanism Using Similarity-Based Retrieval:</b><br/>When user queries do not match existing AIML patterns, the system should preprocess the input (using lemmatization), compute TF-IDF</li> </ul>   |

|  |   |
|--|---|
|  | <p>values, and calculate cosine similarity to find and return the closest matching answer from a CSV file of question–answer pairs.</p> <ul style="list-style-type: none"> <li>● <b>Multi-Modal Interaction:</b><br/>Users should be able to interact with the chatbot through both text-based input and voice commands. The system must support voice recognition for input (using SpeechRecognition) and provide audio responses (using pyttsx3) as an extra functionality.</li> <li>● <b>Rule-Based Logical Reasoning:</b><br/>The chatbot must maintain a knowledge base (KB) containing movie-related facts and rules in first-order logic (FOL). Users should be able to: <ul style="list-style-type: none"> <li>○ Add new facts using the phrase “I know that X is Y.”</li> <li>○ Verify existing facts using the phrase “Check that X is Y.”</li> <li>○ The system should check for contradictions, validate statements using a resolution algorithm, and respond with “Correct,” “Incorrect,” or “Unknown.” Additionally, it should support fuzzy matching with a confidence score for near-miss inputs.</li> </ul> </li> <li>● <b>Image Classification and Visual Interpretability:</b><br/>The system should allow users to submit images (via file selection or URL) for classification. It must: <ul style="list-style-type: none"> <li>○ Process the image through a pre-trained CNN model to predict the associated movie.</li> <li>○ Return the predicted movie label along with a confidence score.</li> <li>○ Provide visual feedback using Grad-CAM to overlay a heatmap on the image, highlighting the regions most influential in the classification decision.</li> </ul> </li> <li>● <b>User-Friendly Interface:</b><br/>All interactions should be intuitive with clear prompts and dialogs, ensuring that both novice and advanced users can easily navigate the system's functionalities.</li> </ul> |
| The employed AI techniques, and the explanation of program codes and the supplied files. | <p><b>1. Conversational Capabilities (Task A):</b></p> <ul style="list-style-type: none"> <li>● <b>AIML for Pattern-Based Responses:</b><br/>The core conversational module is built on AIML. Over 10 additional AIML conversation pairs have been implemented to cover common queries related to movies and TV series. These patterns enable rapid responses for frequently asked questions such as release dates, directors, and genres.</li> <li>● <b>Similarity-Based Retrieval:</b><br/>When a user's input does not exactly match an AIML pattern, the system processes the text by performing lemmatization and then computes TF-IDF representations. Cosine similarity is calculated against a pre-prepared CSV file of question–answer pairs to retrieve the closest match.</li> <li>● <b>Voice Interface Extra Functionality:</b><br/>The chatbot includes an extra layer of interaction through voice commands using Python's SpeechRecognition library for converting spoken input to text, and pyttsx3 for synthesizing speech as an output.</li> <li>● <b>Supplied Files:</b></li> </ul>  |

|  |   |
|--|---|
|  | <ul style="list-style-type: none"> <li>○ <b>AIML File:</b> Contains all the conversation patterns, including the newly added ones.</li> <li>○ <b>CSV File:</b> Holds the Q/A pairs used for similarity-based fallback responses.</li> <li>○ <b>Python Scripts:</b> Contain the code for pre-processing user input, TF-IDF vectorization, cosine similarity calculation, and the voice interface integration.</li> </ul> <p><b>2. Rule-Based Logical Reasoning (Task B):</b></p> <ul style="list-style-type: none"> <li>● <b>First-Order Logic (FOL) with NLTK:</b><br/>The chatbot's reasoning module leverages first-order logic (FOL) to store and validate facts related to movies and TV series. The knowledge base (KB) is created using NLTK's FOL syntax, with domain-specific facts and rules (e.g., "Director(Christopher_Nolan, Inception)").</li> <li>● <b>Dual-Pattern Input Handling:</b><br/>Two input patterns are supported: <ul style="list-style-type: none"> <li>○ "I know that X is Y" for adding new facts.</li> <li>○ "Check that X is Y" for validating existing facts. The system uses NLTK's resolution algorithm to check for contradictions and infer the truth value of the statements.</li> </ul> </li> <li>● <b>Fuzzy Fact Checking:</b><br/>An extra functionality was implemented using Python's difflib module to perform approximate matching. This provides a confidence score when the user's input is similar—but not identical—to stored facts, handling uncertainty in a user-friendly manner.</li> <li>● <b>Supplied Files:</b> <ul style="list-style-type: none"> <li>○ <b>Knowledge Base File (CSV):</b> Contains the initial set of facts and rules.</li> <li>○ <b>Python Scripts:</b> Implement the input parsing, logical reasoning, and fuzzy matching mechanisms.</li> </ul> </li> </ul> <p><b>3. Image Classification (Task C):</b></p> <ul style="list-style-type: none"> <li>● <b>CNN Model with TensorFlow and Keras:</b><br/>A convolutional neural network (CNN) was developed for classifying movie posters. The model comprises two convolutional blocks (Conv2D and MaxPooling2D), followed by a flattening layer, a dense layer with dropout, and a final dense layer with softmax activation.</li> <li>● <b>Image Preprocessing and Prediction:</b><br/>User-submitted images (via file selection or URL) are preprocessed and then passed to the CNN model, which outputs a predicted movie label along with a confidence score.</li> <li>● <b>Extra Functionality – Grad-CAM Visualization:</b><br/>To improve interpretability, Grad-CAM is applied to generate a heatmap overlay on the input image, highlighting the regions that most influenced the model's decision. This required modifying the model architecture slightly (e.g., converting the Sequential model into a functional model) to ensure proper gradient flow.</li> <li>● <b>Supplied Files:</b> <ul style="list-style-type: none"> <li>○ <b>CNN Model File (.h5):</b> Contains the trained CNN model.</li> <li>○ <b>Class Mapping File (Pickle):</b> Maps class indices to movie names.</li> <li>○ <b>Python Scripts:</b> Include code for image preprocessing, prediction, and Grad-CAM visualization.</li> </ul> </li> </ul> |
|--|---|

|  |   |
|--|---|
|  | <ul style="list-style-type: none"><li>○ <b>Sample Image Files:</b> Provided for testing the image classification module (excluding training/testing datasets). Overall, the source code is written in Python with extensive in-line comments to ensure clarity and maintainability. The project is structured into separate modules for conversational capabilities, logical reasoning, and image classification, with each module interacting seamlessly within the integrated chatbot system. All supplementary files and scripts are organized and submitted as part of the project.</li></ul> |
|--|---|

### 3- Conversation log (insert text, screenshots and/or images as required)

Task A:

```
> How many episodes does Breaking Bad have?  
Breaking Bad has 62 episodes.
```

This response is generated by the AIML pattern that directly matches the query.

```
> How many frames per second are in a standard movie?  
Most movies are shot at 24 frames per second (fps), which provides a smooth cinematic experience
```

This user query exactly matches one of the predefined Q/A pairs in the CSV file, so the chatbot responds with the direct answer without needing the TF-IDF similarity check.

```
> How many frames are usually taken per second in a movie?  
Most movies are shot at 24 frames per second (fps), which provides a smooth cinematic experience
```

Here, the question is phrased differently and does not match any of the predefined Q/A pairs word-for-word. The chatbot falls back on the TF-IDF and cosine similarity mechanism to find the closest question in the CSV file. It correctly identifies that this question is similar to the original frames-per-second query and returns the same answer.

```
> I love pizza  
Sorry, I don't know. Can you rephrase?
```

This user input neither matches any predefined AIML pattern nor aligns closely with any Q/A pair in the CSV. Because the similarity score falls below the acceptable threshold, the chatbot triggers its default "I don't know" response, prompting the user to rephrase.

```
Welcome to the Movie & TV Chatbot! Ask me anything. Type 'exit' or 'quit' to stop  
➤ Enable voice interaction? (yes/no): yes  
➤ Listening...
```

The user opts to enable voice interaction. The chatbot now uses the SpeechRecognition library to capture and process voice commands.

```
> what is CGI in  
CGI (Computer-Generated Imagery) refers to digital effects used in movies to create realistic or fantastical visuals.  
➤ Listening...
```

Even though the user's query is slightly incomplete, the chatbot detects the keyword "CGI" and matches it through the TF-IDF fallback mechanism to a stored answer about CGI.

```
➤ Listening...  
> exit  
Goodbye!
```

The user issues an exit command, and the chatbot terminates the session gracefully.

Task B:

```
> I know that The Godfather is a movie  
OK, I will remember that the Godfather is movie.
```

Demonstrates the "I know that ... is ..." pattern. The chatbot checks the knowledge base for contradictions. None are found, so it adds "movie(the\_Godfather)" to the KB. The chatbot confirms the addition with "OK, I will remember that the Godfather is movie."

```
> Check that The Godfather is a movie  
Correct!
```

Demonstrates the “Check that … is …” pattern. The system runs a resolution proof on the statement. It finds the fact “movie(the\_Godfather)” in the KB. The chatbot responds with “Correct!” indicating the statement is in the KB.

```
> Fuzzy check that The Godfather is a movie  
Fuzzy check: The fact 'a_movie(the_Godfather)' is similar to 'movie(the_Godfather)' with a confidence of 0.95.
```

Demonstrates the fuzzy check. Because the user typed “a\_movie” or had spacing/typos, the system calls the difflib-based fuzzy logic. The chatbot reports a confidence score of 0.95, indicating a near match to “movie(the\_Godfather).”

```
> Fuzzy check that The Godfather is movie  
Exact match found: Correct! (Truth value: 1.0)
```

Same fuzzy check command, but now the input closely matches the existing KB entry. The chatbot detects it’s essentially identical to “movie(the\_Godfather)” and reports an exact match with a truth value of 1.0.

```
> Check that The Godfather is a series  
Incorrect!
```

A “Check that … is …” pattern for a contradictory fact. Since the KB states “The Godfather is a movie,” this new statement conflicts with existing knowledge. The chatbot responds with “Incorrect!” to show it found a contradiction.

```
> Check that Lala Land is a director  
I don't know.
```

Another “Check that … is …” pattern. The KB has no info about “Lala Land” or a matching “director” fact, and there’s no direct contradiction either. The chatbot responds “I don’t know.” because it can’t prove or disprove the statement from its KB.

Task C:

```
Welcome to the Movie & TV Chatbot! Ask me anything. Type 'exit' or 'quit' to stop  
➤ Enable voice interaction? (yes/no): no  
> Tell me what is in this image
```

User initiates an image classification request. The chatbot logic checks for the keyword “image,” triggering the CNN classification workflow.

```
Would you like to see a Grad-CAM visualization as well? (yes/no): yes
```

User opts in for Grad-CAM, meaning the chatbot will generate a heatmap showing which parts of the image contributed most to the classification.

```
Do you want to select a LOCAL file or provide a URL? (file/url): url  
Please enter the image URL: https://screencritix.com/wp-content/uploads/2014/03/frozen.jp  
1/1 ━━━━━━━━ 0s 16ms/step
```

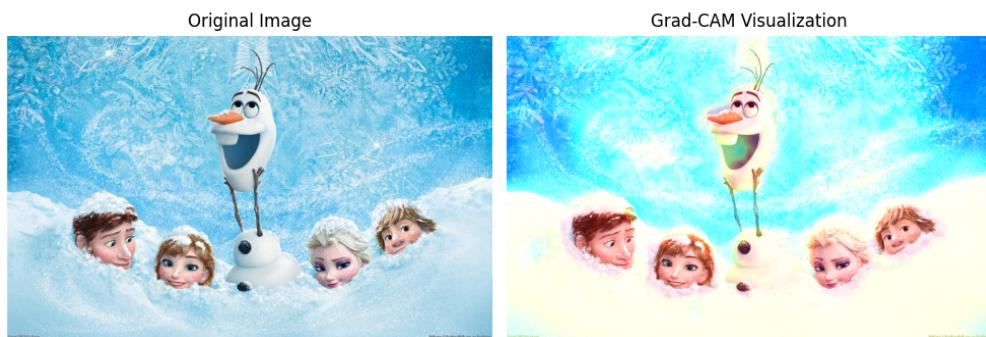
The chatbot downloads the image from the provided URL and preprocesses it: resizing to 128x128, converting to NumPy array, normalizing pixel values. Then it runs model.predict() to classify the image.

```
Using layer: conv2d_1
Generating Grad-CAM visualization...
Grad-CAM visualization saved to gradcam_output.jpg
I see Frozen in this image (Confidence: 0.97).
```

1. **Layer Selection:** “conv2d\_1” is the final convolutional layer used for Grad-CAM.
2. **Grad-CAM:** The chatbot computed gradients from the predicted class (“Frozen”) back to the conv2d\_1 feature maps, then overlaid them on the original image.
3. **Confidence 0.97:** The CNN is 97% certain that the poster belongs to “Frozen.”
4. **gradcam\_output.jpg:** The system saved the heatmap overlay to this file.)



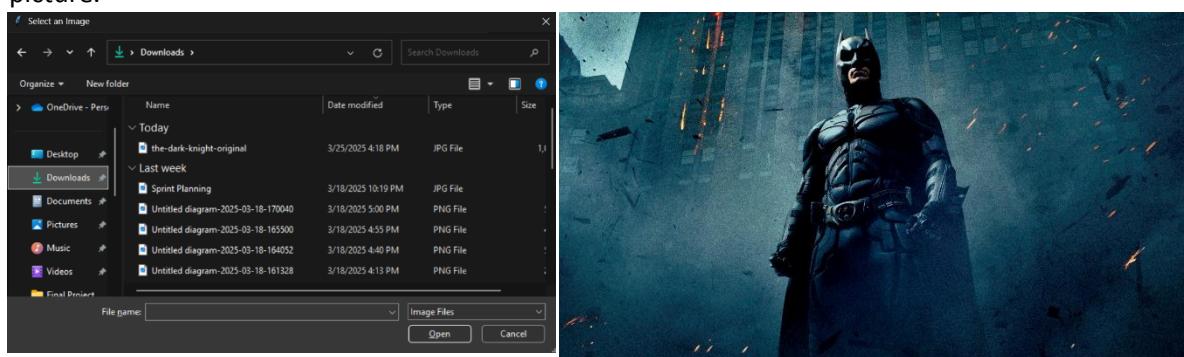
User’s image from the URL—Frozen characters in snow.



The bright areas in the Grad-CAM overlay indicate which parts of the poster, in this case it was the faces of the characters and the snow that contributed most to identifying “Frozen”.

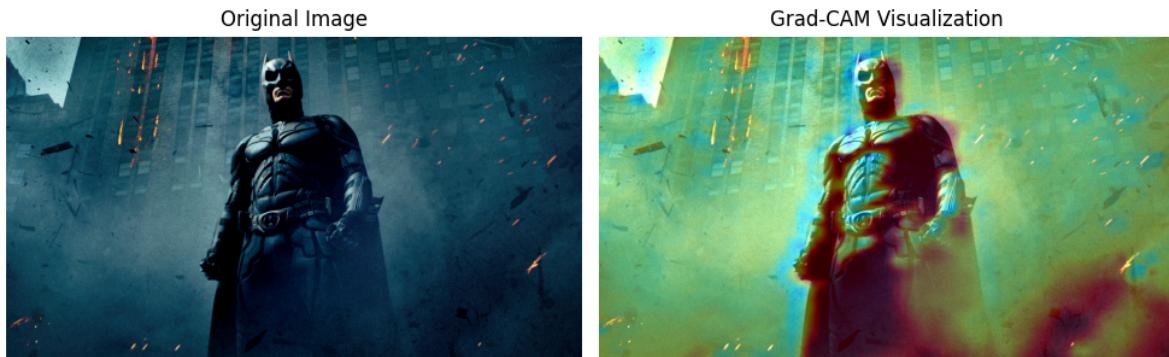
```
Do you want to select a LOCAL file or provide a URL? (file/url): file
1/1 ━━━━━━ 0s 16ms/step
```

A file selection dialog appears, and the user chooses a local image—**The Dark Knight** poster—by browsing their files. See screenshot below showing the pop-up file browser and the original picture.



I see Inception in this image (Confidence: 0.38).

Despite the image actually being *The Dark Knight*, the model guesses “Inception.” This can happen if both posters share visual similarities (e.g., dark color palette, same director, text placement), causing the CNN to confuse them. The confidence is only 0.38—relatively low. This suggests the CNN wasn’t sure which class best matches the poster.



The heatmap might highlight Batman’s figure or background areas. However, the CNN incorrectly mapped these features to “Inception.” The relatively low confidence indicates it wasn’t a strong match.

```
> image
Would you like to see a Grad-CAM visualization as well? (yes/no): no
Do you want to select a LOCAL file or provide a URL? (file/url): file
No file selected.
No image selected or an error occurred.
```

The chatbot popped up a system dialog to pick a local image. The user didn’t select any file. The chatbot detects that file\_path is empty or invalid, printing “No file selected” and a generic error message. The chatbot simply ends the image classification flow without crashing, letting the user continue with other chatbot features.

```
select a LOCAL file or provide a URL? (file/url): url
he image URL: s
image from URL: Invalid URL 's': No scheme supplied. Perhaps you meant https://s?
ted or an error occurred.
```

The chatbot attempts to fetch “s” as a URL, fails due to missing protocol (`http://` or `https://`). The code catches this error, prints a clear error message (“Invalid URL”), and halts classification. The chatbot suggests the user might have meant “`https://s`,” but ultimately it can’t proceed with classification.

(no word count is necessary)