Nadya Pena
CSC415 – Assignment 4
March 31st, 2015

1) Detailed design class diagrams:
    a. **Figure 1.** CommentExtractor Package – Extracts and organizes comments from Crunchyroll so the episode classification modules can easily access and parse them.
    b. **Figure 2.** EpisodeClassifier – Uses the extracted comments to decide whether the current episode for a given show is a filler or not, as well as positive or negative.
    c. **Figure 3.** Database Structure – Shows the database and what it stores
    d. **Figure 4.** High-Level overview of the system
2) Statechart that model the behavior of the algorithms
    a. **Figure 5.** Filler episode classifier – shows the steps of the algorithm classifying whether an episode is filler or not
    b. **Figure 6.** Sentiment classifier – shows the steps taken by the Sentiment classifier to determine if commenters liked the episode (positive sentiment) or disliked the episode (negative)
3) Test Case Design document
    a. Testing Approach
    b. Testing Plan Table
4) **Github repository information:** https://github.com/nadyac/SWE2015-Project/tree/master/src/documentation

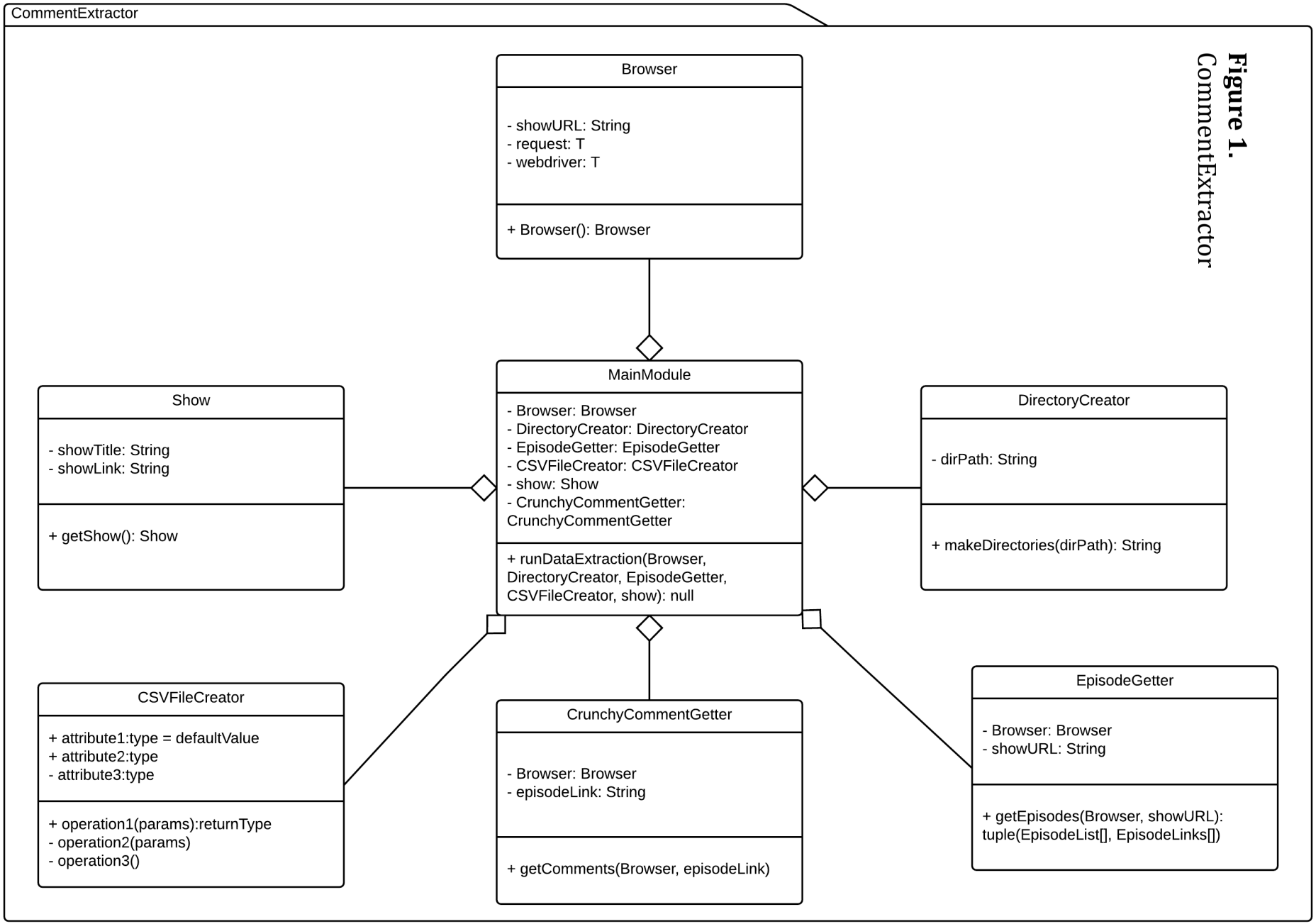


Figure 1. CommentExtractor

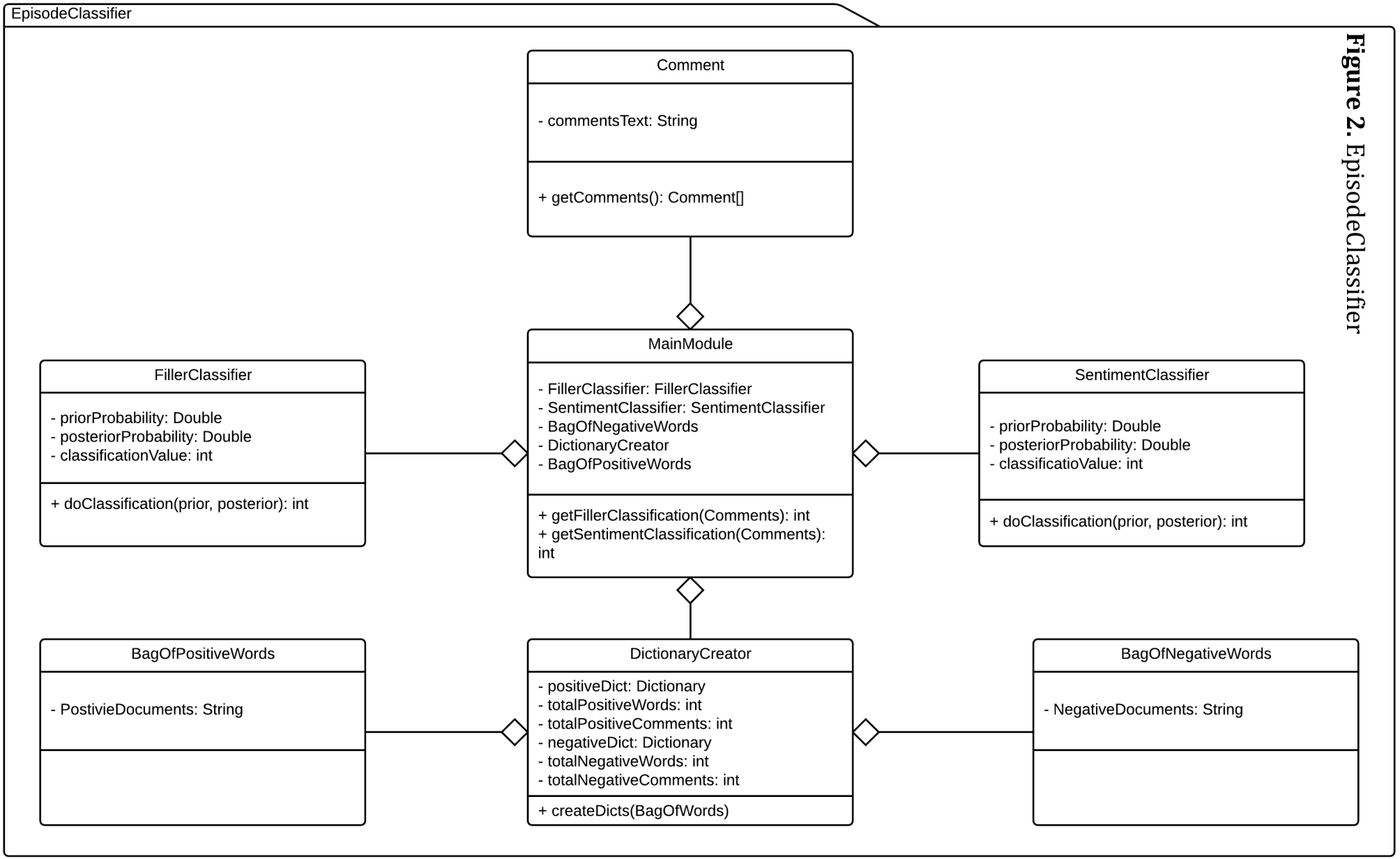


Figure 2. EpisodeClassifier

Figure 3. The Database

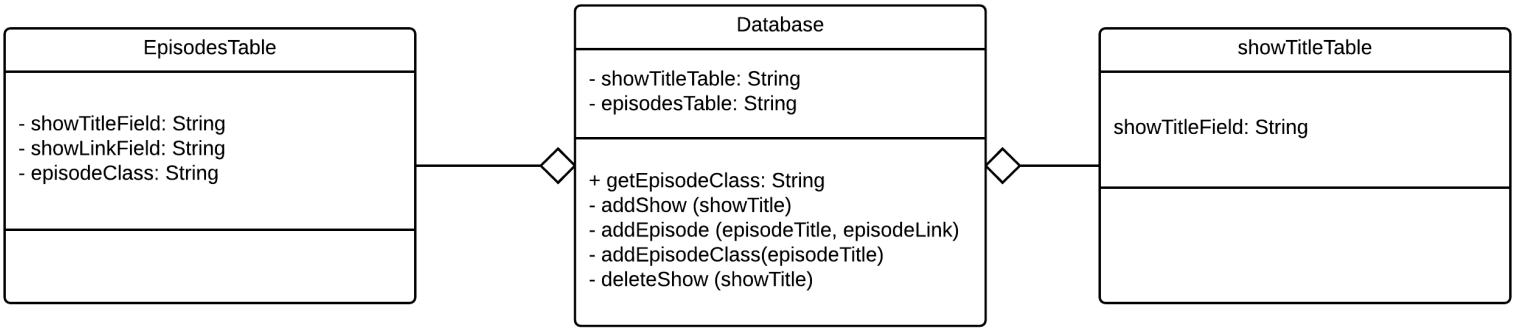


Figure 4. High-Level Overview of System

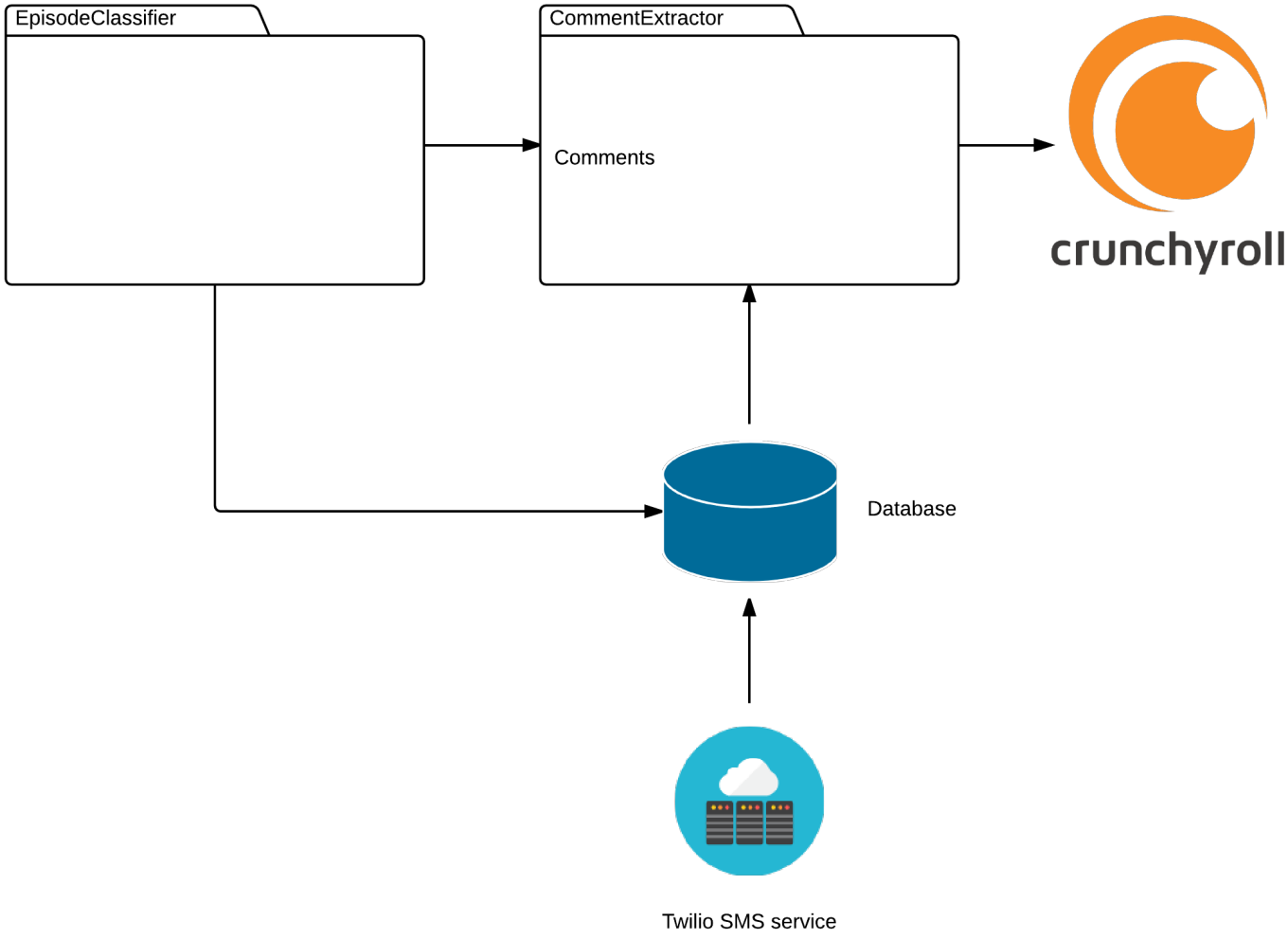


Figure 5. Filler Classifier State Chart

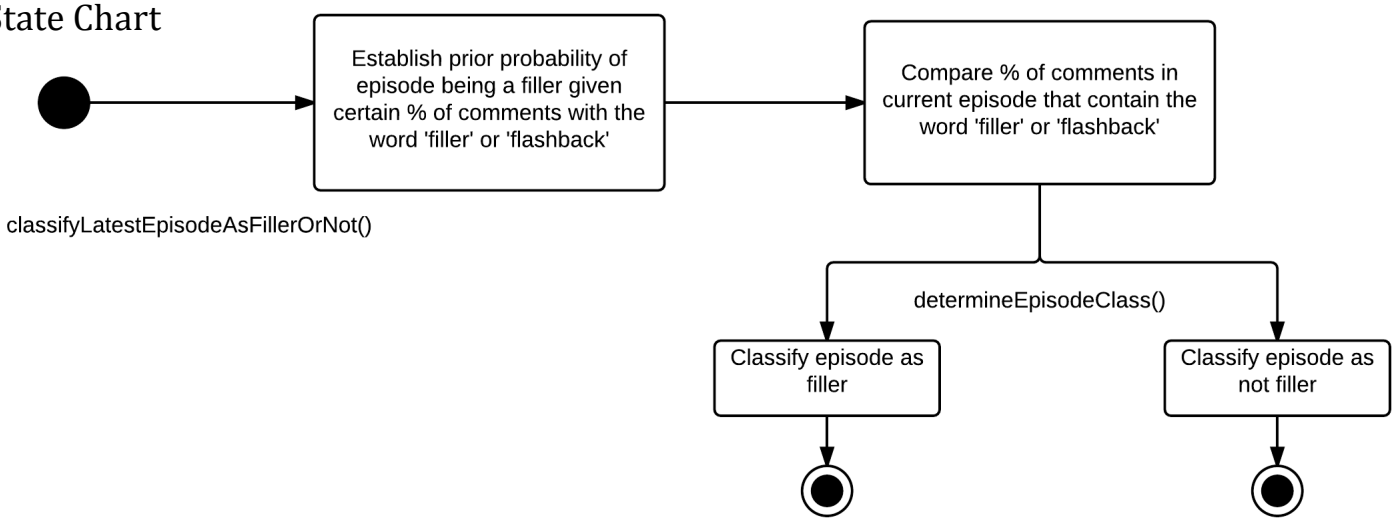
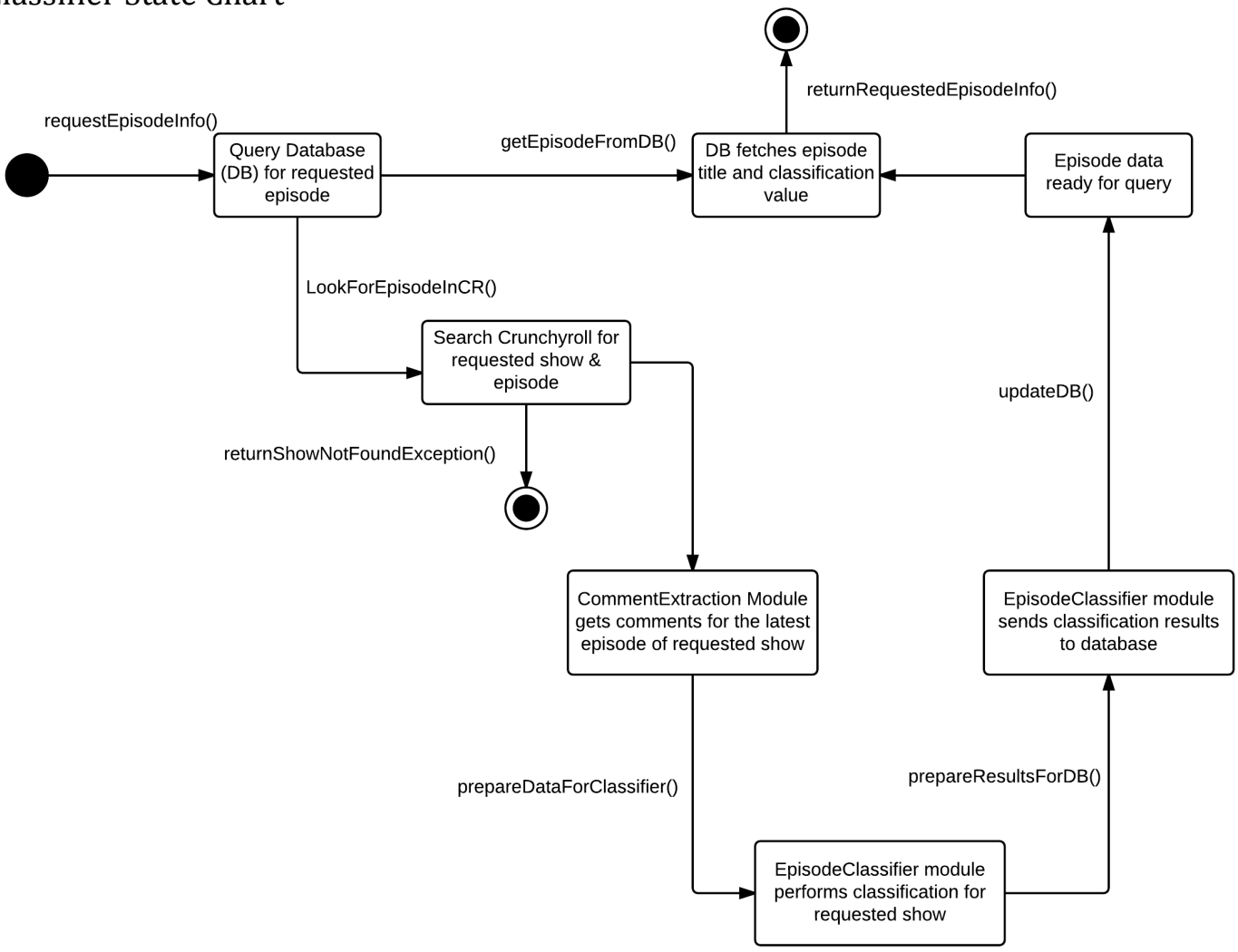


Figure 6. Sentiment Classifier State Chart



About the testing approach
Since this system is heavily comprised of modules, it is important to test each module separately in a series of Unit tests. The tests will be written in python and they will leverage the 'assert' method to check output or result of each module. Integration testing will be done periodically after each new module is added to the base system.

Since users will be interacting with the system via SMS and will be receiving responses through SMS, desktop-based tools like JMeter or Google Test will not be effective. Performance measurements will need to be taken manually for a set of 10 diverse shows. Currently the shows used for testing range from 11 episodes in length, to 72 episodes in length. Retrieval and processing time for each show will be recorded and relevant statistics such as mean response times, failure rates, and extraction rate will be recorded.

Functionality Tested	**Inputs**	**Expected Output**	**Actual Output**
Creation of output directories	Path location	Directories should exist in the specified path location	'Pass ' if the created directories exist 'Fail' otherwise.
Comment Extraction	CSV Files for each episode	CSV file with containing comments	'pass' if file contains comments. 'Fail' if file is blank
CSV File creation	Path location in which to create the CSV file	CSV file at the specified location	'Pass' if the CSV File exists at the specified location. 'Fail' otherwise.
Get the latest episode for the requested show	Show title	List of episodes for the given show	'Fail' if list of episodes retrieved is blank
Word Dictionaries Creation	A small bag of words (in a text file)	Dictionaries with each unique word and their frequency	'Fail' if expected dictionary values are not correct
Classification algorithms	Show episodes of each class	Correct or nearly correct classification	The expected class and the calculated probabilities associated with them
Database storage	Show, related episodes, and classes	Database should contain things put into it	'fail' if the database does not contain the right information
Database deletion	Show title	Database should delete episodes and classes associated with the show	'pass' if show and related records are deleted. 'Fail' otherwise.
Twilio SMS Service responsiveness	Show title	Response with latest episode and the classes associated with it (filler or not, negative or positive)	Test passes if correct response is given within reasonable amount of time