Video Number:	xx_yy
Video Title:	Video title here
Estimated Length:	x min (400-500 words = 2.5-3.5 minutes)

Video learning goal: At the end of this video, the student will learn how to do …(your learning objective).

What will you cover? Add 3 to 5 brief bullet points of what you will cover in this video. This will eventually become the video description, which has the potential to boost search and viewership by 40% because of effective SEO. This also helps you create an outline before scripting.

- Point 1
- Point 2
- Point 3
- Point 4

Use the **table below** to write your dialogue and show what will be on screen while you are talking.

Begin with a good **hook** to get the member's attention and engage them from the start. Perhaps tell a brief story or anecdote. This will vary by subject matter and course type. Do your best if the content is tech heavy.

Next let them know the **learning goal** for the movie in a few sentences. Explain what you'll show them how to do. Or, what they'll be able to do after watching.

Share your content. Refer back to the bullet points you wrote above. Try to break up your script so each paragraph is no longer than 1 **to 3 sentences**. If there is a shift to what is seen or you move on to a new topic, then you can start a new paragraph.

If there is a main point that we want the audience to remember, then bold it.

Make sure you have a conclusion. Consider reminding the member of your key points during the movie. Or give them a call to action. Do not tell them what is coming up. Movies are meant to be standalones as well as part of the overall course.

	Script text or talking points	**Visuals / Actions on Screen**
A	**Data Sturctures:** **Vectors** (`Vec<T>`): Vectors are dynamic arrays that can grow or shrink in size. **Arrays** (`[Type; Number]`**):** In web development, arrays can be employed for tasks that require a fixed number of elements.	Explain bit from this : https://github.com/FatGuyy/LinkedIn-Course/blob/master/std-library/01_02.rs

	Hash Maps (`std::collections::HashMap`): Hash maps are key-value stores that provide fast data retrieval. In web applications, they are used for tasks like storing session data, routing information, or caching, and in calling different API **Queues (`std::collections::VecDeque`):**Deques are valuable for implementing tasks like managing request/response queues in a web server.	
B	**File I/O (`std::fs` and `std::io`):**	Explain **file** from: https://github.com/FatGuyy/LinkedIn-Course/blob/master/std-library/01_02.rs
C	**Error Handling (Result and Option):**	Result example Option example
D	**Concurrency and Multithreading (`std::thread` and `std::sync`):** `std::thread` allows you to create and manage threads in Rust `std::sync` provides synchronization primitives for sharing data between threads safely. `Arc` (Atomic Reference Count) and a `Mutex`. `Arc` allows multiple threads to share ownership of the data safely.	Threading example
E		
F		
G		
H		
I		
J		
K		
