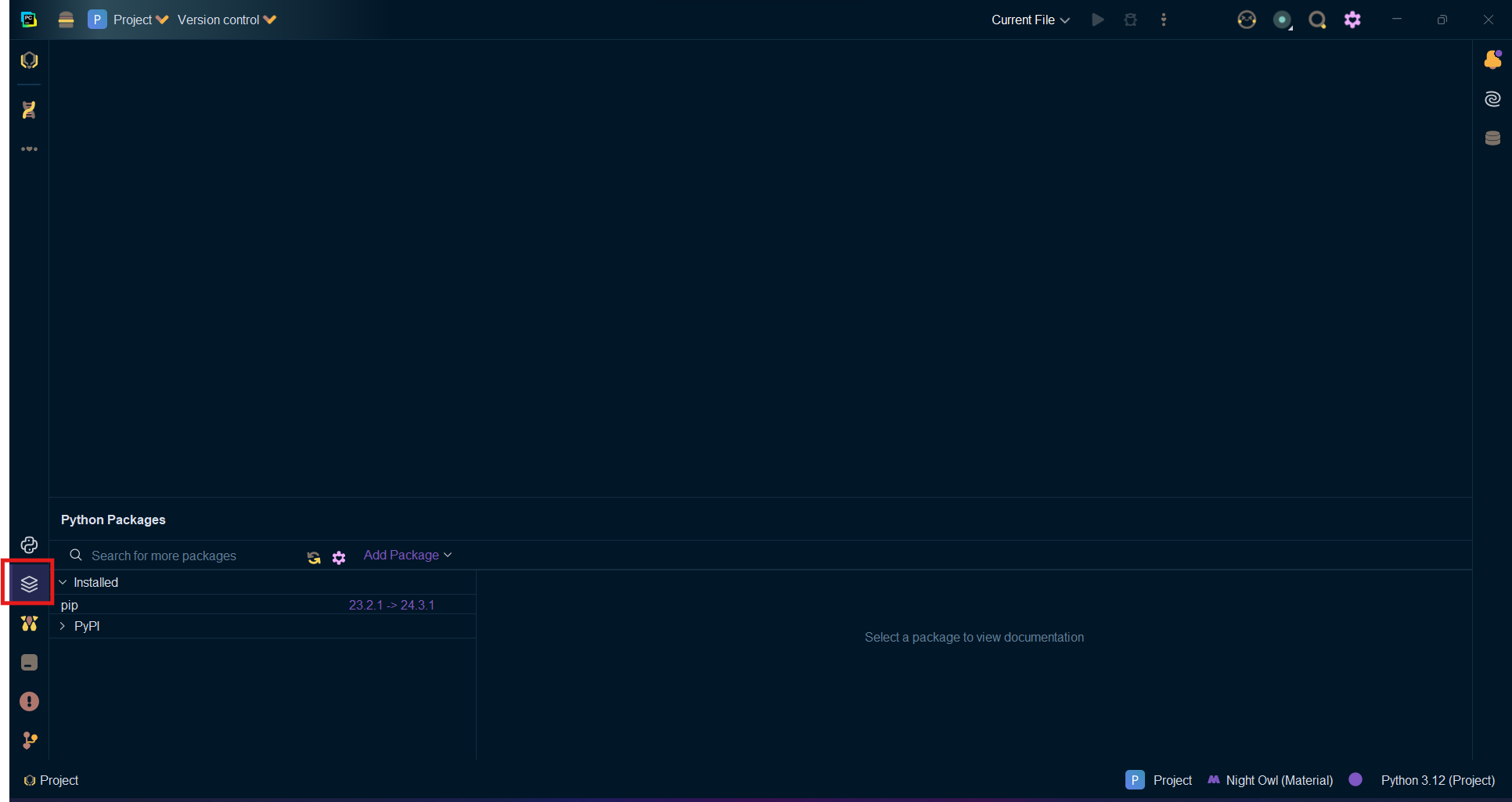
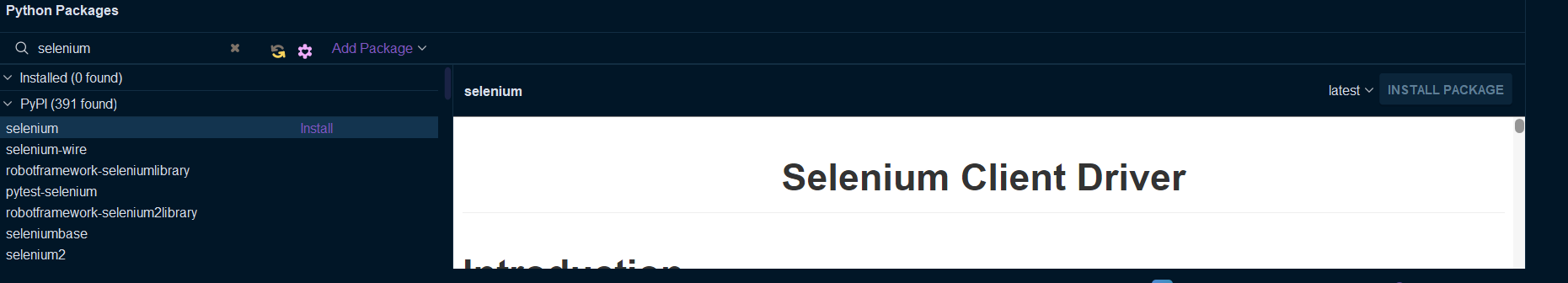
**Report on Selenium and PyAutoGUI Integration for Automated YouTube Navigation and Recording**

**1. Setup and Initial Configuration**

The project necessitated the installation of several essential packages in PyCharm, including Selenium, PyAutoGUI, PyAudio, and others. Additionally, the Chrome WebDriver was downloaded and placed in the project directory to facilitate browser automation with Selenium. To use selenium in this project, I needed to download corresponding web driver for web browser. In this case I use Google Chrome, thus I needed to download the Chrome web driver and place it in the project directory (I could place it in other folders but I would need to specify a path to that location, in my case we just type the name of the executable).I have also added ‘reques’ package.

The program is developed under Windows 11 Pro operating systems, IDE of choice is PyCharm and all packages will be installed in the venv (virtual environment) using PyCharm’s built-in package manager.

To access the package manager, I needed to select it form the left side bar, it is the second item in the bottom part of the sidebar, as we can see below:

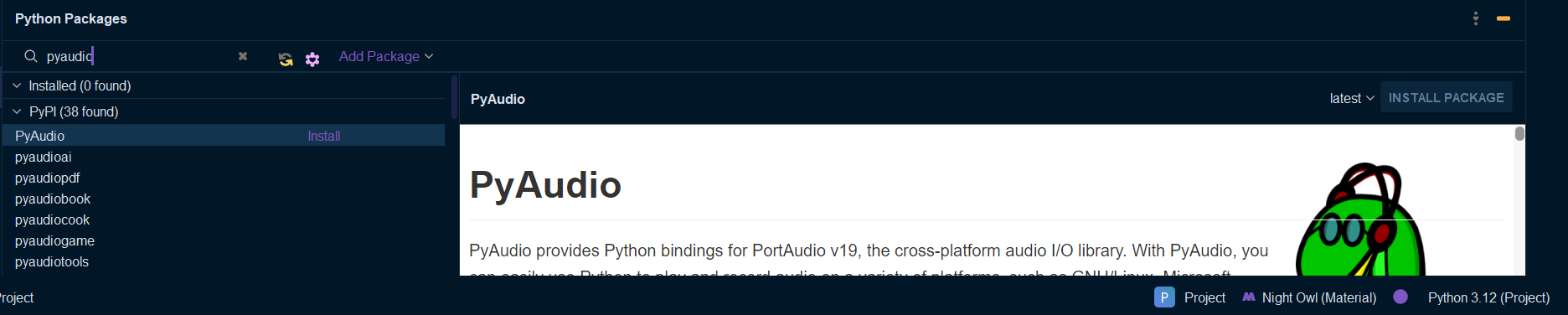
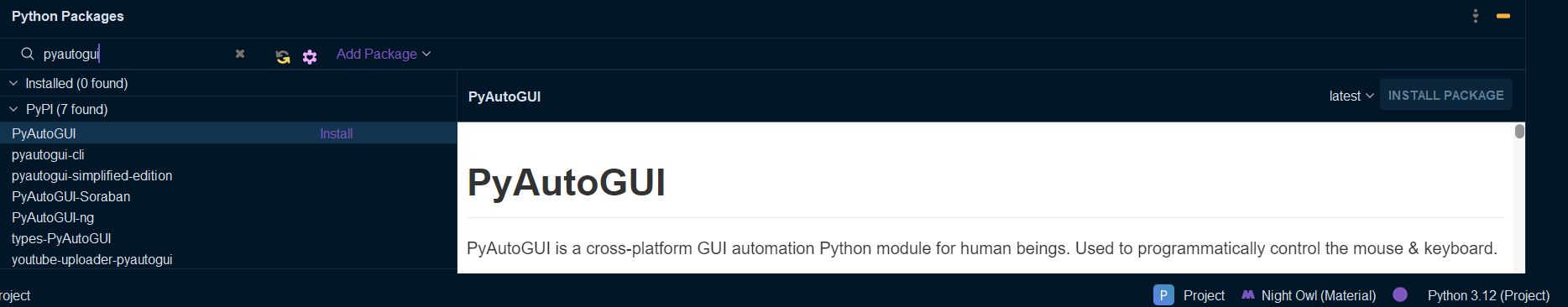
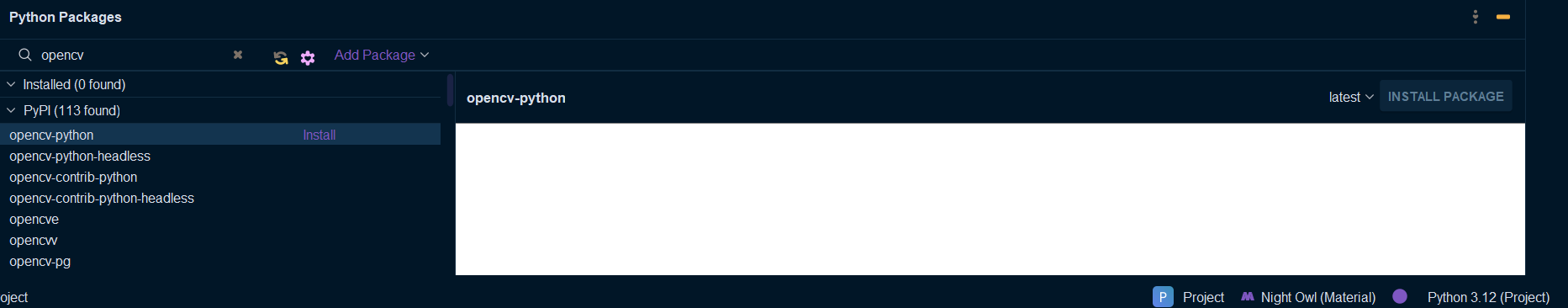
In the search bar I needed to type the name of the package I wanted to install:

Next to the install package button we have a dropdown menu, which allow us to select a specific verion, here I downloaded the latest version of selenium.

We will repeat the same steps for other libraries.

Alternatevly we can use pip to install python packages. Pip is a command line tool, the sytax for it is:

pip install <package name>



No errors were encountered during the installation of these packages.

**2. Code Development for YouTube Navigation and Playback**

I commenced by writing code to navigate to YouTube, search for a specific song, and play it. During this process, we encountered several issues:

**Issue 1: Cookie Pop-Up Dialog**

The first obstacle was the cookie pop-up dialog that appeared upon visiting YouTube. This dialog interfered with the script's execution. To address this, I implemented code within Selenium to automatically close the pop-up:

python

def reject\_cookies(driver):

try:

elements = driver.find\_elements(By.CSS\_SELECTOR, "#dialog button.yt-spec-button-shape-next--mono")

if elements:

elements[0].click()

else:

print("Cookie button not found.")

except NoSuchElementException:

print("Cookie pop-up not found.")

**Issue 2: Search Button Execution**

The program initially failed to execute the click on the search button. Upon review, I identified this as a timing issue. Despite utilizing explicit waits, the issue persisted, prompting us to insert a call to the sleep function after entering the search term. This adjustment successfully resolved the problem.

Additionally, I noted that while developing the ad-skipping function, commenting out code for recording video and audio occasionally caused the search term not to be entered into the search bar. To mitigate this, I added another sleep function call to handle this bug when necessary.

**3. Screen Recording with PyAutoGUI**

Subsequently, I developed a screen recording feature using PyAutoGUI. However, I encountered an issue with missing dependencies, specifically the Pillow package, which was subsequently installed.

**Issue: Video Playback Speed**

Initially, the video playback speed was excessively fast due to the frame rate being set to 30 fps. After some research, we discovered that PyAutoGUI's fps setting affects encoding, and its internal fps is not adjustable. By lowering the fps to 5, the video played slower but became laggy, leading us to switch to the MSS library for recording.

python

frame = cv2.cvtColor(frame, cv2.COLOR\_BGRA2BGR)

This line resolved color inversion issues arising from the difference between the BGRA and BGR color systems used by MSS and OpenCV, respectively.

**4. Audio Recording**

My initial attempt to record audio using PyAudio was unsuccessful. I switched to the soundcard and soundfile libraries, becase I had experienced audio stuttering when using PyAutoGUI for recording, likely due to threading issues, which were resolved by switching to MSS.

**5. Combining Audio and Video**

I utilized FFmpeg to combine audio and video files. Initially, the function for combining files did not work, resulting in a silent video. Several attemps with testing with FFmpeg commands in the command line proved successful, so I incorporated these commands using the os.system() call in our script.

python

os.system("ffmpeg -i video.mp4 -i audio.wav -c:v copy -c:a aac output.mp4")

Then I added code to delete the intermediate audio and video files:

python

os.remove("video.mp4")

os.remove("audio.wav")

**6. Extracting Audio and Writing dB Values**

The final step involved extracting audio from the video file and writing the dB values to a text file using NumPy for calculations. The formula used was sourced from the Internet and requires verification for accuracy.

**Timing and Selenium**

Timing is critical in Selenium, necessitating a detailed explanation of its implementation. Selenium supports two types of waits:

1. **Explicit Waits**: Called explicitly in the code, these waits use conditions such as element visibility or clickability. For example:

python

wait = WebDriverWait(driver, 15)

wait.until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, 'input#search')))

1. **Implicit Waits**: Configured globally, implicit waits instruct Selenium to wait for a specified time for elements to appear:

python

driver.implicitly\_wait(10)

Unlike the sleep function, which pauses execution for a fixed duration, waits only pause as long as necessary. Due to issues with the pop-up dialog and browser behavior, we occasionally had to rely on sleep instead of waits.

**Conclusion**

Throughout this project, we faced and resolved multiple challenges related to timing, dependencies, and integration of various libraries. The final implementation successfully navigates YouTube, plays a selected song, records video and audio, and processes the recorded data to produce a combined media file. Future improvements could focus on refining timing mechanisms and further optimizing the recording process.