1.Web Automation Using Selenium:

• Deploy Selenium to access an e-commerce site.

• Execute a product search for &quot;Hindi Books&quot; and scrape details such as

the name, price, and user rating of the books.

• Organize the scraped data in a structured JSON or CSV format.

**Instructions for Setting Up and Running Selenium Script in PyCharm**

**Prerequisites:**

1. **Download and Installation**:
   * Download the project zip file and extract it to your preferred location on your system.
   * Ensure Python is installed on your system and its path is set up in the environment variables.
2. **IDE Setup**:
   * Open PyCharm and create a new project or open an existing one.

**Installing Dependencies**:

* Navigate to the project directory in PyCharm's terminal.
* Install the dependencies listed in requirements.txt using the following command

pip install -r requirements.txt

**Running the Selenium Script:**

1. **Configuration**:
   * Ensure ChromeDriver executable is downloaded and placed in a directory accessible to the script. Update the path in the script if necessary.
2. **Execution**:
   * Open the selenium.py script in PyCharm.
   * Optionally, if you prefer headless mode (without opening a browser window), uncomment chrome\_options.add\_argument("--headless") in the script.
   * Run the script by clicking the Run button in PyCharm or using the terminal.

python selenium.py

**Output**:

* After successful execution, the script will scrape data from Flipkart for Hindi Books.
* It will save the scraped data into hindi\_books.json (JSON format) and hindi\_books.csv (CSV format) files in the project directory.

Mobile Interaction Using Appium:

• Set up Appium to engage with a mobile application displaying a productlist with prices.

• Capture a screenshot of this product list directly from the app.

**Step 1: Install Java Development Kit (JDK)**

1. **Download JDK**: Visit [Oracle JDK download page](https://www.oracle.com/java/technologies/javase-jdk11-downloads.html) and download the JDK installer for Windows.
2. **Install JDK**: Run the downloaded installer and follow the installation instructions. After installation, set the JAVA\_HOME environment variable:
   * Right-click on This PC or Computer → Properties → Advanced system settings.
   * Click Environment Variables....
   * Under System Variables, click New....
   * Variable name: JAVA\_HOME
   * Variable value: Path to your JDK installation directory (e.g., C:\Program Files\Java\jdk-11).
3. **Add JDK to PATH**: Append %JAVA\_HOME%\bin to the Path variable in the same Environment Variables window.
4. **Verify Installation**: Open a command prompt and type java -version and javac -version to ensure Java is installed correctly.

**Step 2: Install Node.js and npm**

1. **Download Node.js**: Visit Node.js download page and download the Windows installer.
2. **Install Node.js**: Run the downloaded installer and follow the installation instructions. This will also install npm (Node Package Manager).
3. **Verify Installation**: Open a command prompt and type node -v and npm -v to verify Node.js and npm installation.

**Step 3: Install Android Studio**

1. **Download Android Studio**: Visit [Android Studio download page](https://developer.android.com/studio) and download the Windows installer.
2. **Install Android Studio**: Run the downloaded installer and follow the installation instructions. Make sure to install Android SDK and Android Virtual Device (AVD) during setup.
3. **Set Android SDK Path**: After installation, open Android Studio and navigate to Tools → SDK Manager. Note the Android SDK Location as you will need this path later.

**Step 4: Set Up Virtual Device (AVD)**

1. **Create Virtual Device**: Open Android Studio, click on Configure → AVD Manager.
2. **Create a New Virtual Device**: Click Create Virtual Device, choose a device definition, and follow the wizard to create a new virtual device (emulator). Make sure to select a device with Google Play support if you want to install apps like Flipkart.

**Step 5: Install Flipkart App on Emulator**

1. **Download Flipkart APK**: You can download the Flipkart APK from trusted sources. Make sure it's compatible with the Android version of your emulator.
2. **Install APK on Emulator**: Drag and drop the downloaded APK file onto the emulator screen, or use adb install command from Android SDK tools (adb should be in your PATH, located in <Android SDK Location>\platform-tools).

### Step 6: Install Appium

1. **Install Appium**: Open a command prompt and install Appium using npm:

npm install -g appium

1. **Install Appium Doctor**: Appium Doctor helps diagnose and fix common setup issues.

npm install -g appium-doctor

1. **Check Appium Installation**: Run appium-doctor to verify Appium installation and diagnose issues.

### Step 7: Install UIAUTOMATOR

 **Install UI Automator Viewer**: Use Android Studio SDK Manager to ensure Android SDK Platform-Tools is installed.

 **Locate UI Automator Viewer**: Navigate to <Android SDK Path>\tools\bin in Command Prompt.

 **Launch UI Automator Viewer**: Run uiautomatorviewer to inspect UI elements.

 **Check Emulator Details**: Use adb devices -l to list connected emulators and note down the emulator name.

 **Verify Setup**: Ensure emulator is running with emulator -avd <emulator\_name> and check connection with adb devices. Adjust paths as per your installation directories.

**Explanation:**

* **Imports**: Imported necessary modules (webdriver, MobileBy, time, Image, BytesIO, TouchAction) for Appium automation and image processing.
* **Capabilities**: Defined Appium capabilities including platformName, automationName, deviceName, app, appPackage, and appActivity.
* **Driver Initialization**: Established a connection to the Appium server using webdriver.Remote.
* **Exception Handling**: Added try-except blocks to handle potential exceptions during operations like clicking on elements and capturing screenshots.
* **Search and Interaction**: Interacted with elements on the Flipkart app such as search input, search result, and handling popups.
* **Screenshot Capture**: Captured screenshots of each book element during the scrolling process.
* **Scrolling**: Performed scroll actions using TouchAction to navigate through multiple pages of search results.
* **Session Termination**: Closed the driver session (driver.quit()) at the end of the script to release resources.

Run the script by clicking the Run button in PyCharm or using the terminal.

python appium.py

NOTE: Despite encountering compatibility issues on my older system with Android Studio, I've diligently crafted code based on my own research from documentation and YouTube tutorials. While alternatives have been explored, concerns remain about execution reliability. Optimal performance is expected on systems meeting or exceeding 4GB RAM specifications. I am committed to troubleshooting and seeking solutions, leveraging community insights to overcome challenges in mastering Appium and mobile automation.

Text Extraction with OCR:

 Apply OCR technology on the screenshot to extract product names and

prices.

 Match the OCR-derived data with the Selenium-scraped data to identify

any price discrepancies.

 **OCR Text Processing**:

* After using pytesseract.image\_to\_string(image), split the text into lines (lines = text.splitlines()).
* If lines is not empty (if lines:), strip the first line (product\_name = lines[0].strip()). This assumes the first line contains the product name.
* If no text is found (else:), assign "Unknown" to product\_name.

 **Regex Adjustments**:

* The regular expressions for price and ratings remain the same.
* Adjusted ratings\_match.group(1) to directly get the digits (ratings count) from the match.

 **CSV Writing**:

* Writes product\_name, price, and ratings to the CSV file instead of filename.

 **Output**:

Optionally prints product\_name, price, and ratings for each image processed.

Navigate to project terminal and run

python OCR.py

Data Comparison and Report Generation:

• Analyze the data to determine matching and mismatching prices between

the two methods.

• Generate a detailed report summarizing:

• Products with matching prices.

• Products with discrepancies in prices.

The script imports the csv module and defines dummy image\_prices. It initializes result\_dict to store comparison results: "matching\_prices" for matching OCR-extracted prices and "price\_discrepancies" for discrepancies. It reads book details from "book\_details.csv" using csv.DictReader, appending each row to extracted\_data. It compares OCR-extracted prices with expected prices (price extracted from CSV) and categorizes results based on equality, appending to respective lists in result\_dict. Finally, it writes results to "comparison\_results.csv", handling exceptions like FileNotFoundError and other unexpected errors to ensure robust operation.

Navigate to terminal and run the commands.

python compare.py