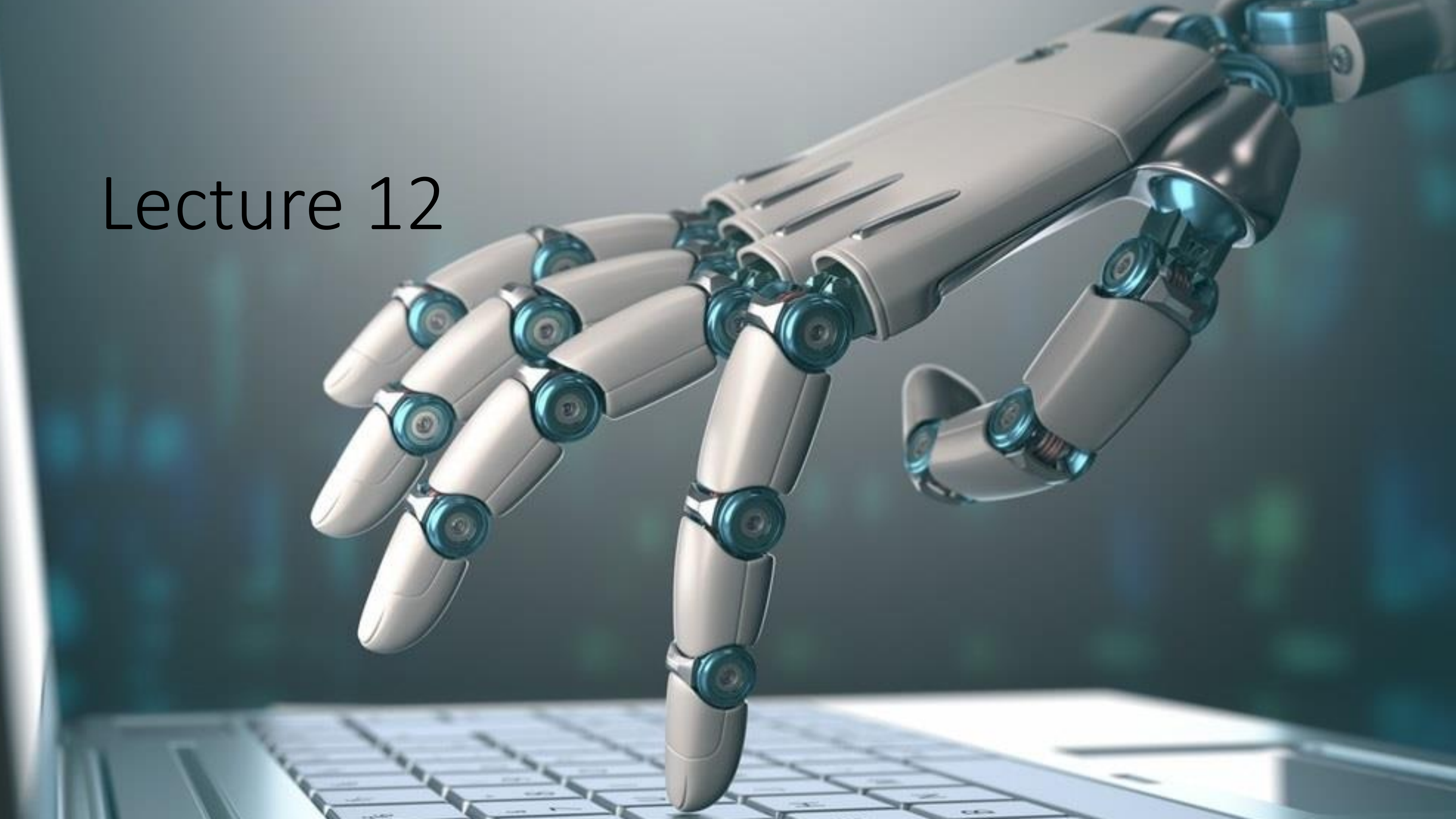


Lecture 12

THE END IS NEAR

Demo time

Tasks that are given for interviews

Task:

Create automation test for www.cinamonkino.lv , including following steps:

1. Sign in

2. Open User Profile

3. Create (check if exists) Name and Surname (Blabla & Tech)

4. Choose one movie with random date and time (+5 days from current date) in Cinema: Cinamon Alfa Riga

5. Ticket confirmation steps:

5.1 Add 2 tickets

5.2 Add random text in coupon field and check validation

5.3 Store Sum for 2 tickets(will be needed to validation in last step)

5.3 Move to next step

5.4 Check random seats from row: 5 or 6 (if availabe).

If not then choose any random seats

5.5 Store selected seats

5.5 Move to next step

5.6 Check stored sum from step 1 in Ticket Confirmation Steps.

5.7 Check stored seats from step 2 in Ticket Confirmation Steps.

5.8 Change the Order

5.9 Select 1 ticket

5.10 Store sum for 1 ticket -> Move to step 2 in Ticket Confirmation Steps

5.11 Leave randomly selected seat -> Move to step 3 in Ticket Confirmation Steps

5.12 Check stored sum from step 1 in Ticket Confirmation Steps.

5.13 Check availability of 5 payment methods(do not select).

5.14 Logout

- **For this task please use Java. You can implement it using any known to you technology: Selenium, Selenide, Cucumber, just simple Junit tests or anything else.**
- The task to implement:
- - Test will have to open Chrome browser and navigate to https://stackoverflow.com webpage.
- - It should be possible somewhere in your code/scenarios to provide a list of search words. An example: [“python”, ”java”, ”android”, ”api”, ”2013”]
- - Your program would have to go through the list and find if there is such word in one of the questions on the page. If there is, it would have to save stackoverflow question. An example: I’ve attached a screenshot, where I hightlighted the very first option. So for the word “2013” it would notice there is such topic and save the whole topic (“crm onpremise 2013 domain change”).
-
- - These topics (aka questions) have to be saved into the text file, separated by newline symbol. Bonus points if you are able to use SQLite database to store this data, instead of a plain text file.
- - In addition, no matter what solution you use for the previous task. You have to form a proper JSON object and save it to text file. The text file should look like:
- {
-     “python”: [“This is one question about python”, “python is bad”, “python is the best”],
-     “java”: [],
-     “android”: [“is android better than ios?”],
-     “api”: [],
-     “2013”: [“crm onpremise 2013 domain change”]
- }
- So basically, it’s an object, which has **KEY** as a single search word from the initial list, and then an array containing all found questions (aka topics). Note, if none was found for the search word, you still have to include it in the JSON but just with an empty array.
- You can save your homework in Github or bitbucket. Just make the repository public. Or, you can just attach a zip file and send it to me via email. Just be careful **do not** attach any .class files or executables J

1. Create a simplified Webdriver-based framework (Java), following the best practices and using any BDD framework (Gherkin language).

**Use the Chrome browser in tests.*

---

2. In the test

*Given I authenticate at "**https://postman-echo.com/basic-auth**" using username "**postman**" and password "**password**"*

*Then I get "**authenticated:true**" response*

authenticate on given website using any method, **excluding** basic auth headers passed in URL like https://postman:password@postman-echo.com/basic-auth and not invoking any third-party bytecode from Java.

---

3. In the test

*When I navigate to "**https://www.google.com/**"*

*Then I verify that main image matches "**test.jpg**" image*

verify that "**test.jpg**" image (take a screenshot of your actual Google logo) exists on the Google webpage when it is opened by the Webdriver.

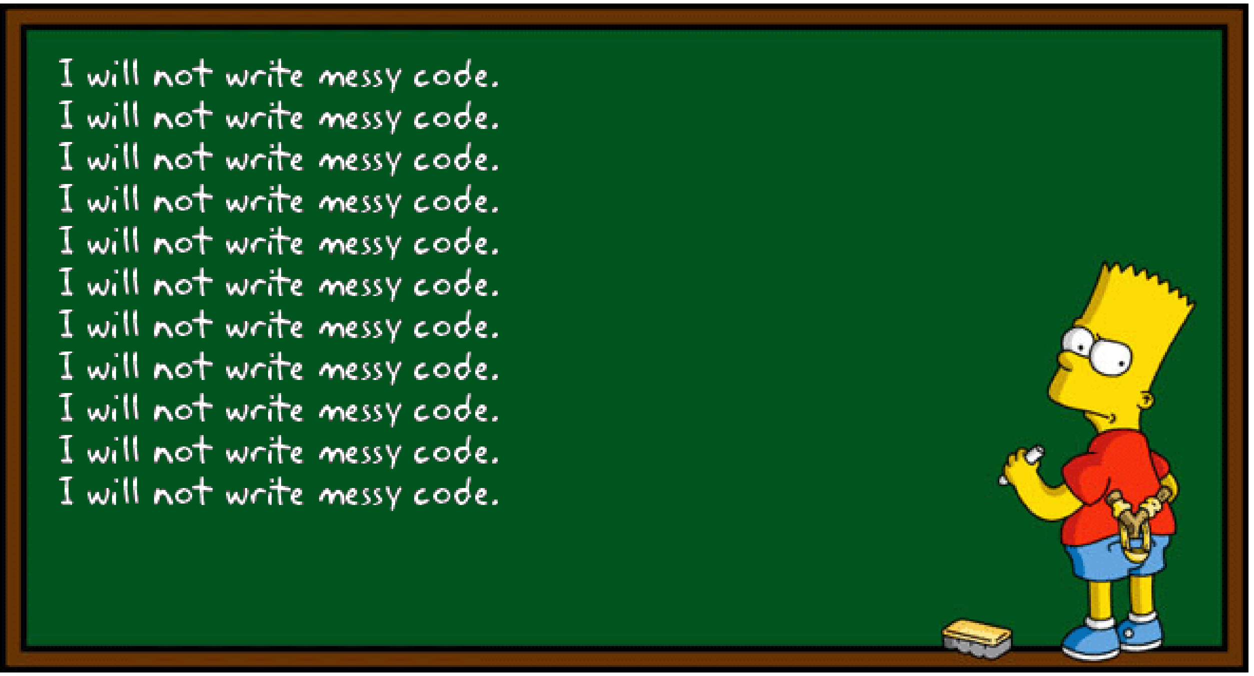
Proximity is up to 80% (we can state that actual image is at least 80% similar to the source image).

Notes:
a. Comparison of the checksum for both downloaded images (or taking a screenshot of certain area on a website and matching bytecode for both images) is not a solution since images can come from different sources and might have different image compression noise and artifacts.
b. You can invoke any third-party code from Java, and use any additional libraries for image recognition.

---

4. Compress the result to a zip file, add a detailed instruction on how to compile and run the code, add the required libraries (if some of them will not be fetched during project assembly).

Now you have portfolio


I will not write messy code.
I will not write messy code.
I will not write messy code.
I will not write messy code.
I will not write messy code.
I will not write messy code.
I will not write messy code.
I will not write messy code.
I will not write messy code.
I will not write messy code.
I will not write messy code.

Friends don't let Friends not use Git

- Git is your friend
- GitHub is your best friend

More Octocats at http://octodex.github.com

What to study next

- Javascript or Python
- Javascript – Cypress or Jasmine
- Appium –mobile app testing
- Rest - RestAssured

Where to look for Job

- TestDevLab
- Accenture
- Visma
- Beetrootlab
- Evolution

QA 3

- Database testing
- Rest testing
- Soap Testing
- Mobile testing – Android – App testing and mobile browser testing
- TestUI – overview – Appium + Selenide framework overview
- Visual Testing, Game Testing
- Jenkins and Reporting
- Final project of your choice

THANK YOU FOR YOUR ATTENTION

GV.com