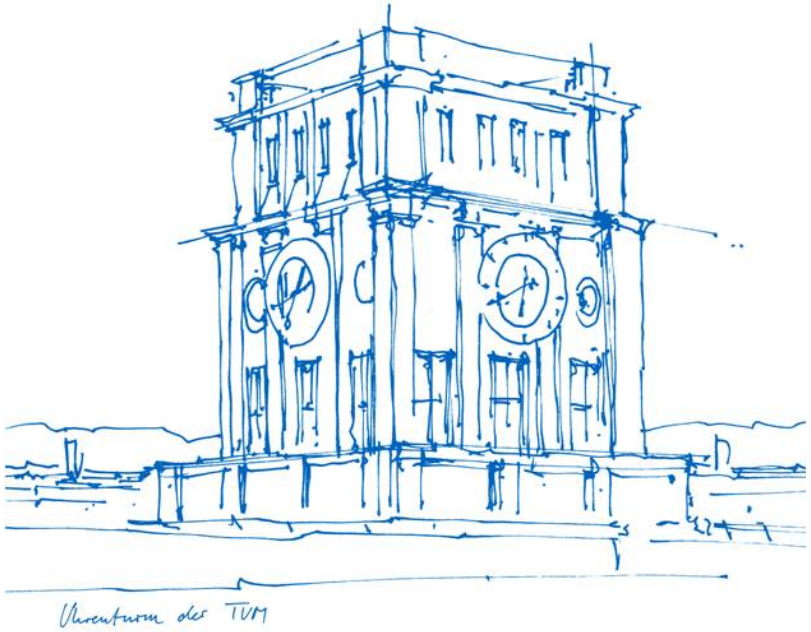
Load Testing and Performance Evaluation of the Theia Online IDE

Bachelor Thesis Presentation
Author: Tobias Klingenberg
Supervisor: Prof. Dr. Stephan Krusche
Advisor: Matthias Linhuber

Technische Universität München
TUM School of Computation, Information and Technology
Applied Education Technologies
Garching, 23. October 2025



Outline

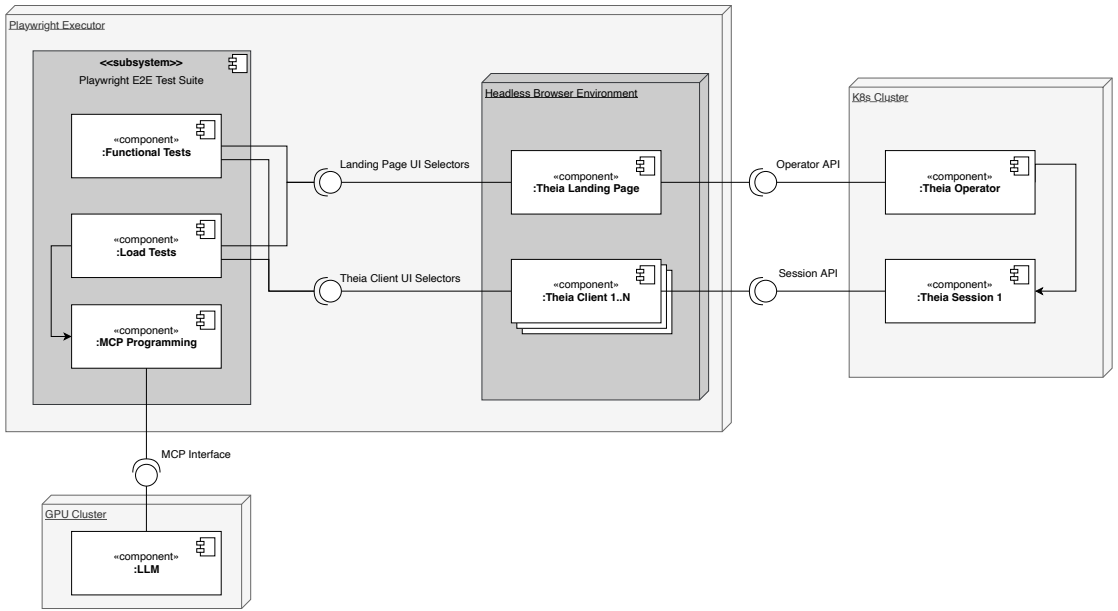
1. Introduction
2. Problem and Motivation
3. Objectives
4. Functionality Tests
5. Load Testing
6. Evaluation
7. MCP Testing / Demo
8. Status
9. Questions

Introduction

Motivation and Problem

Objectives

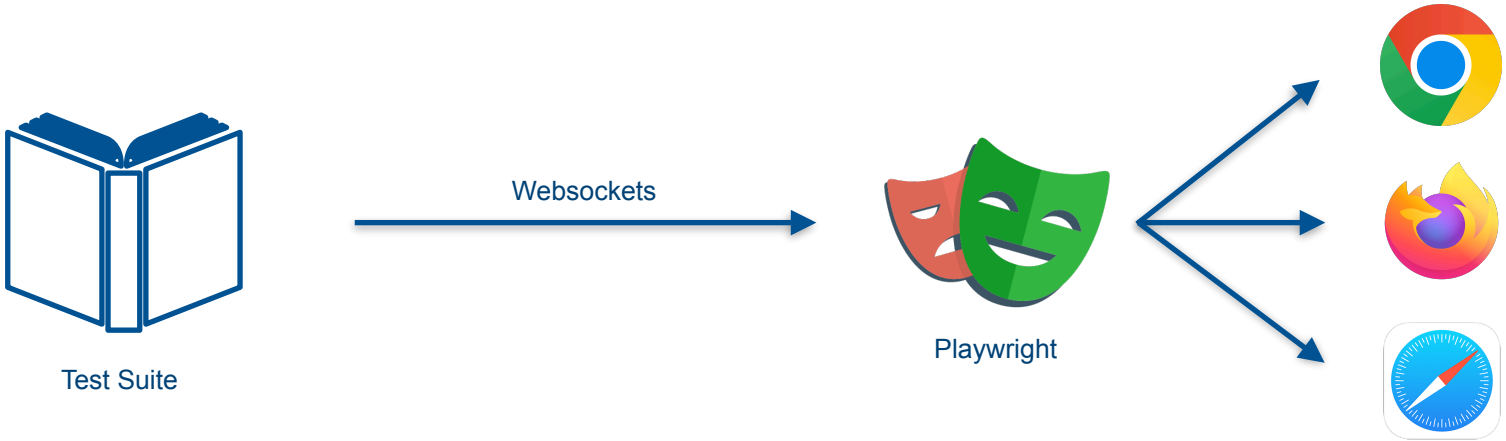
Top Level Design

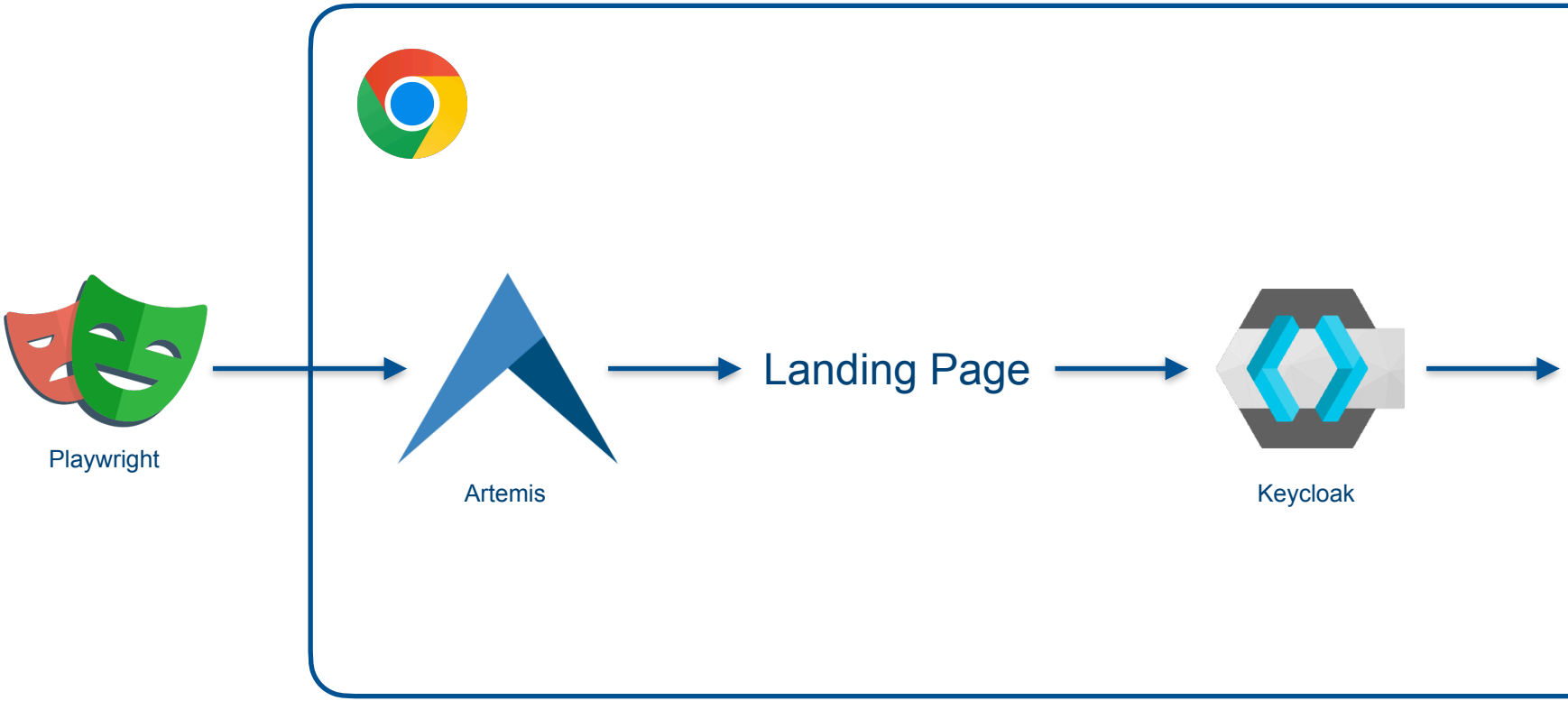


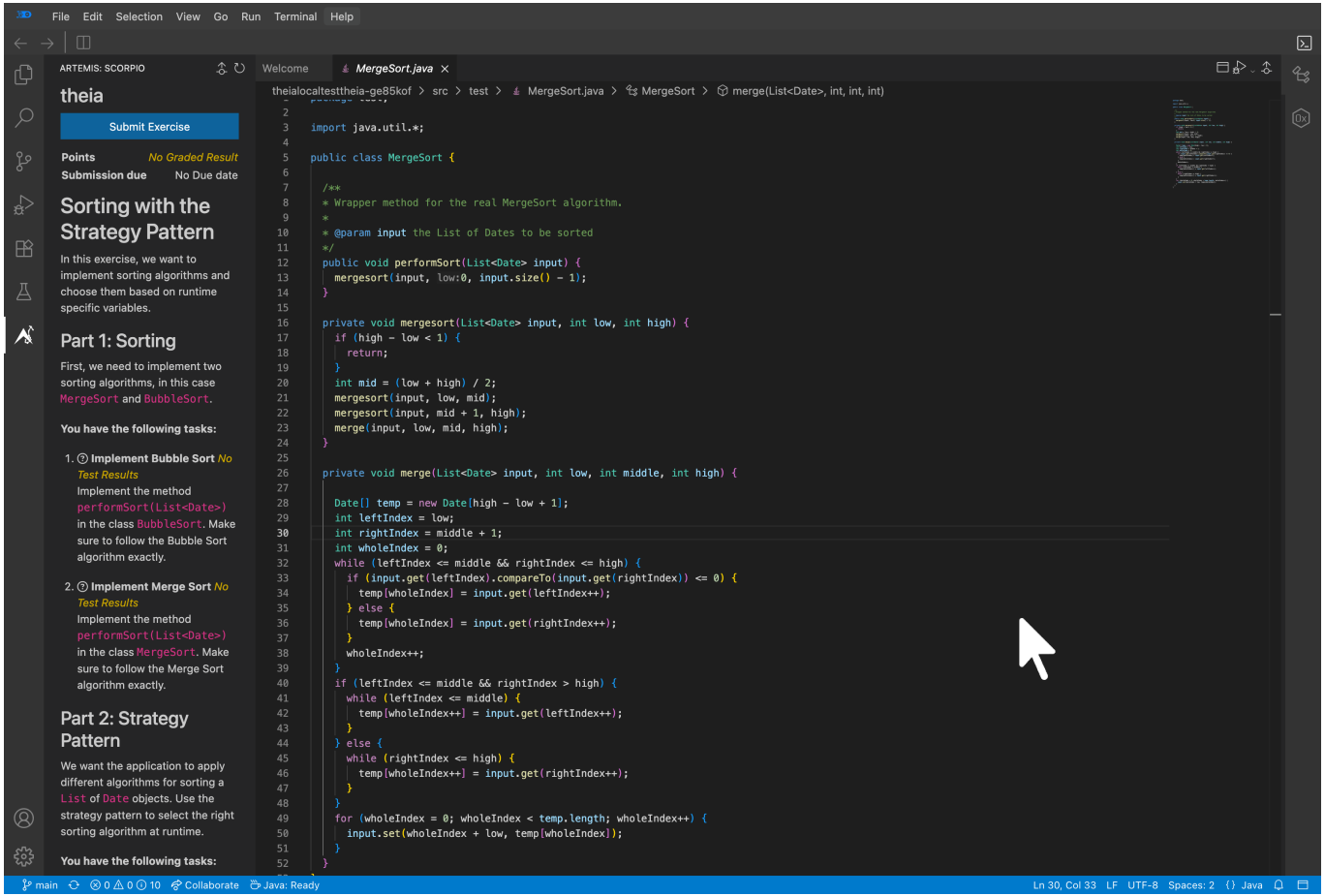
Functionality Tests

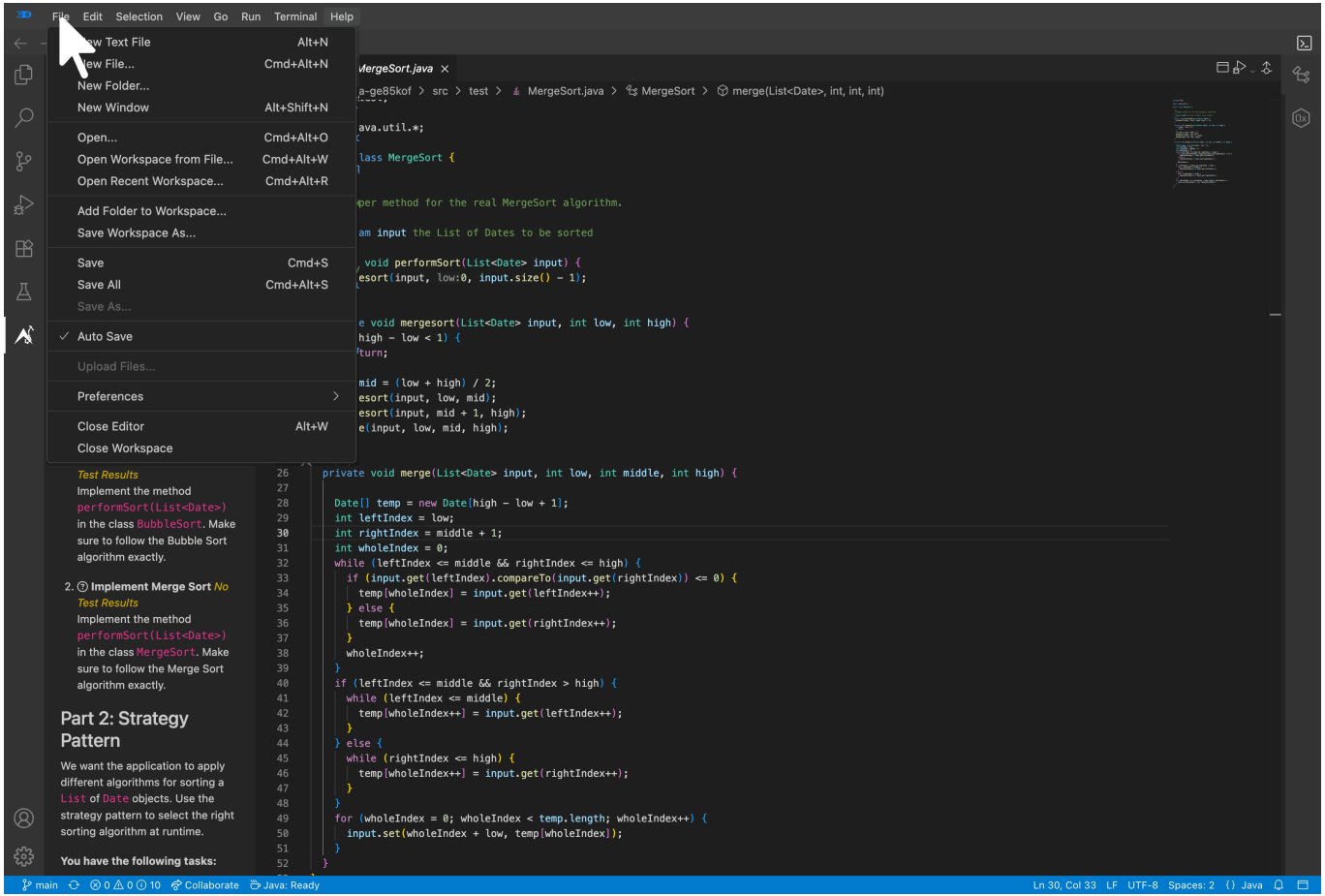
Functionality Tests

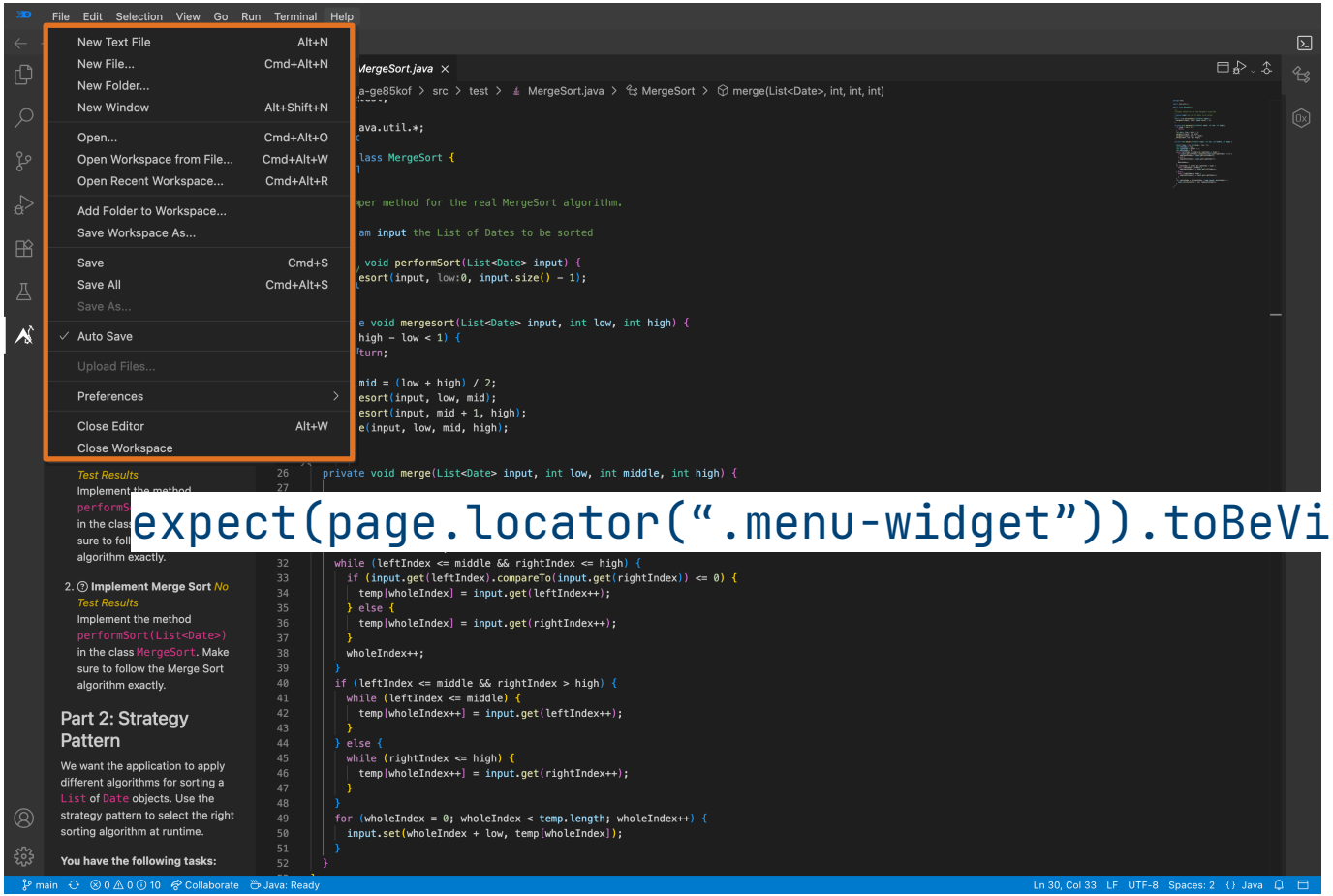
Playwright




Playwright
Artemis
Landing Page
Keycloak

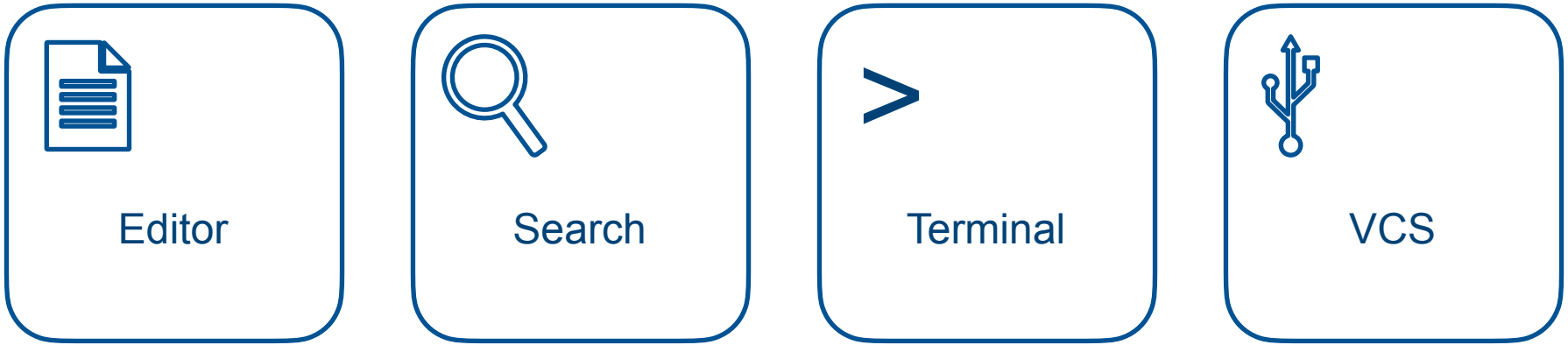

File Edit Selection View Go Run Terminal Help
ARTEMIS: SCORPIO
theia
Submit Exercise
Points No Graded Result
Submission due No Due date
Sorting with the Strategy Pattern
In this exercise, we want to implement sorting algorithms and choose them based on runtime specific variables.
Part 1: Sorting
First, we need to implement two sorting algorithms, in this case MergeSort and BubbleSort.
You have the following tasks:
1. Implement Bubble Sort No Test Results
Implement the method performSort(List<Date>) in the class BubbleSort. Make sure to follow the Bubble Sort algorithm exactly.
2. Implement Merge Sort No Test Results
Implement the method performSort(List<Date>) in the class MergeSort. Make sure to follow the Merge Sort algorithm exactly.
Part 2: Strategy Pattern
We want the application to apply different algorithms for sorting a List of Date objects. Use the strategy pattern to select the right sorting algorithm at runtime.
You have the following tasks:
Welcome
MergeSort.java
thealocaltesttheia-ge85kof > src > test > MergeSort.java > MergeSort > merge(List<Date>, int, int, int)
import java.util.*;
public class MergeSort {
/**
* Wrapper method for the real MergeSort algorithm.
*
* @param input the List of Dates to be sorted
*/
public void performSort(List<Date> input) {
mergesort(input, low:0, input.size() - 1);
}
private void mergesort(List<Date> input, int low, int high) {
if (high - low < 1) {
return;
}
int mid = (low + high) / 2;
mergesort(input, low, mid);
mergesort(input, mid + 1, high);
merge(input, low, mid, high);
}
private void merge(List<Date> input, int low, int middle, int high) {
Date[] temp = new Date[high - low + 1];
int leftIndex = low;
int rightIndex = middle + 1;
int wholeIndex = 0;
while (leftIndex <= middle && rightIndex <= high) {
if (input.get(leftIndex).compareTo(input.get(rightIndex)) <= 0) {
temp[wholeIndex] = input.get(leftIndex++);
} else {
temp[wholeIndex] = input.get(rightIndex++);
}
wholeIndex++;
}
if (leftIndex <= middle && rightIndex > high) {
while (leftIndex <= middle) {
temp[wholeIndex++] = input.get(leftIndex++);
}
} else {
while (rightIndex <= high) {
temp[wholeIndex++] = input.get(rightIndex++);
}
}
for (wholeIndex = 0; wholeIndex < temp.length; wholeIndex++) {
input.set(wholeIndex + low, temp[wholeIndex]);
}
}
main 0 0 10 Collaborate Java: Ready
Ln 30, Col 33 LF UTF-8 Spaces: 2 Java


File Edit Selection View Go Run Terminal Help
New Text File Alt+N
New File... Cmd+Alt+N
New Folder...
New Window Alt+Shift+N
Open... Cmd+Alt+O
Open Workspace from File... Cmd+Alt+W
Open Recent Workspace... Cmd+Alt+R
Add Folder to Workspace...
Save Workspace As...
Save Cmd+S
Save All Cmd+Alt+S
Save As...
Auto Save
Upload Files...
Preferences
Close Editor Alt+W
Close Workspace
Test Results
Implement the method performSort(List<Date>) in the class BubbleSort. Make sure to follow the Bubble Sort algorithm exactly.
2. Implement Merge Sort No Test Results
Implement the method performSort(List<Date>) in the class MergeSort. Make sure to follow the Merge Sort algorithm exactly.
Part 2: Strategy Pattern
We want the application to apply different algorithms for sorting a List of Date objects. Use the strategy pattern to select the right sorting algorithm at runtime.
You have the following tasks:
MergeSort.java
MergeSort > merge(List<Date>, int, int, int)
private void merge(List<Date> input, int low, int middle, int high) {
Date[] temp = new Date[high - low + 1];
int leftIndex = low;
int rightIndex = middle + 1;
int wholeIndex = 0;
while (leftIndex <= middle && rightIndex <= high) {
if (input.get(leftIndex).compareTo(input.get(rightIndex)) <= 0) {
temp[wholeIndex] = input.get(leftIndex++);
} else {
temp[wholeIndex] = input.get(rightIndex++);
}
wholeIndex++;
}
if (leftIndex <= middle && rightIndex > high) {
while (leftIndex <= middle) {
temp[wholeIndex++] = input.get(leftIndex++);
}
} else {
while (rightIndex <= high) {
temp[wholeIndex++] = input.get(rightIndex++);
}
}
for (wholeIndex = 0; wholeIndex < temp.length; wholeIndex++) {
input.set(wholeIndex + low, temp[wholeIndex]);
}
}
main
Collaborate
Java: Ready
Ln 30, Col 33 LF UTF-8 Spaces: 2 Java



expect(page.locator(“.menu-widget”)).toBeVisible()

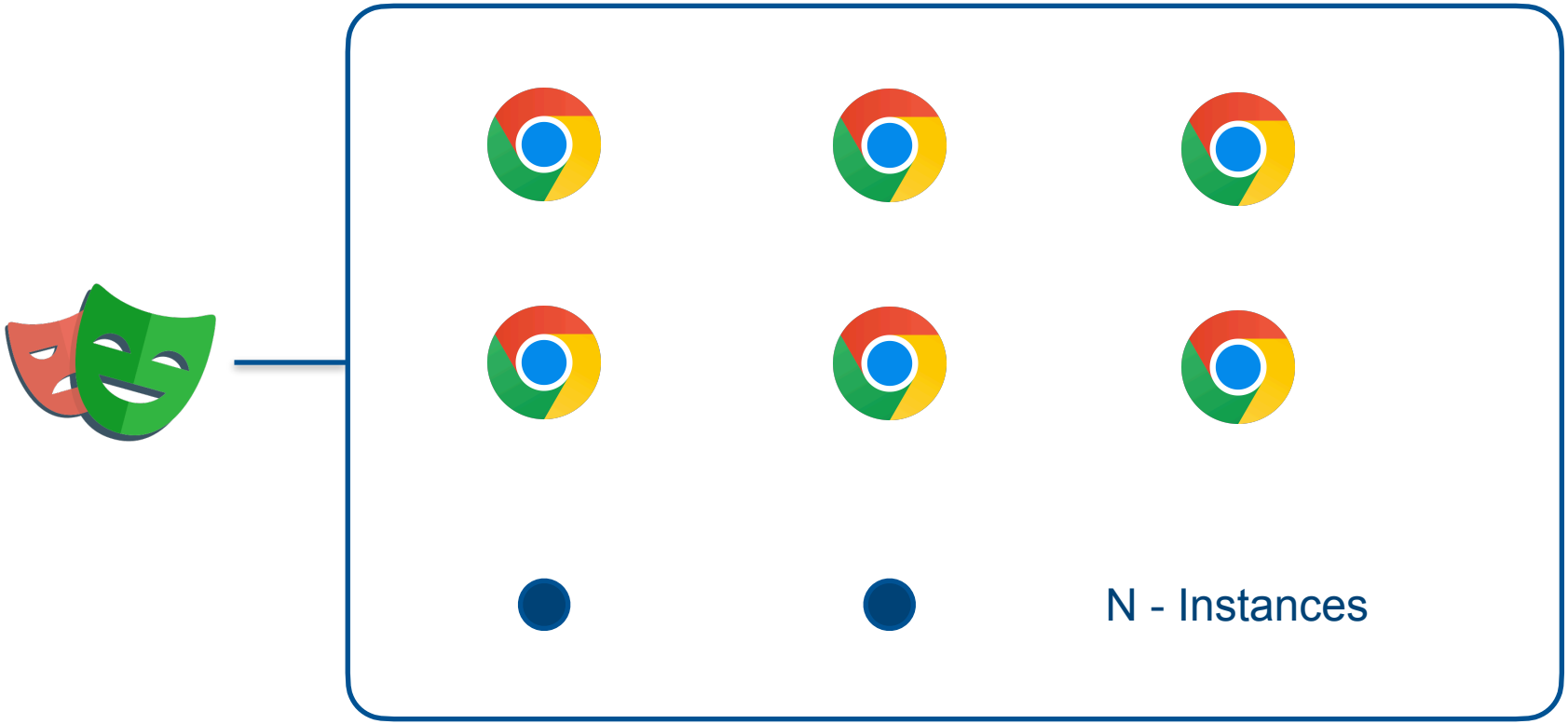
Functionality in Test

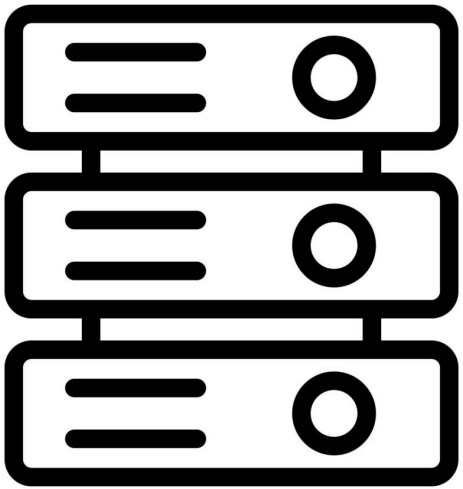


Load Testing

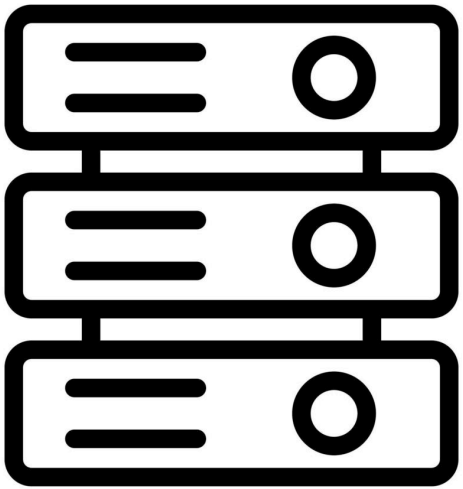
Load Testing

Load Testing

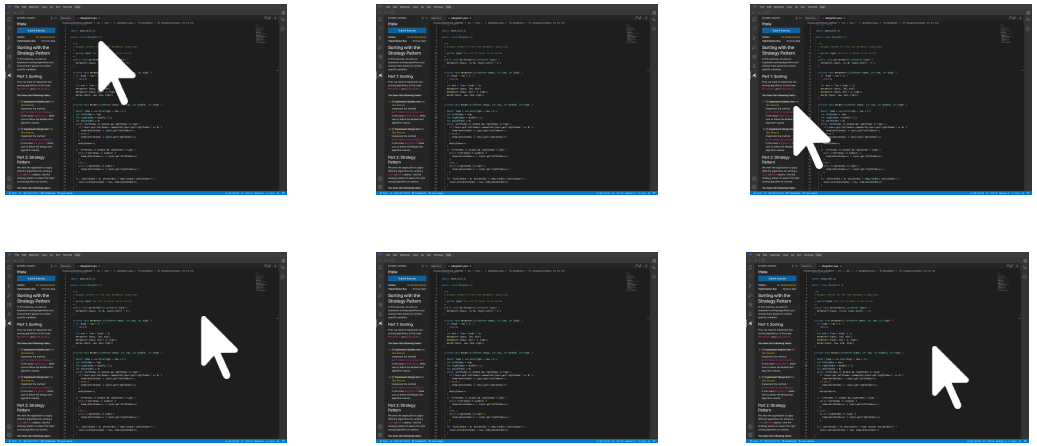

N - Instances



Prewarming (Preview…)

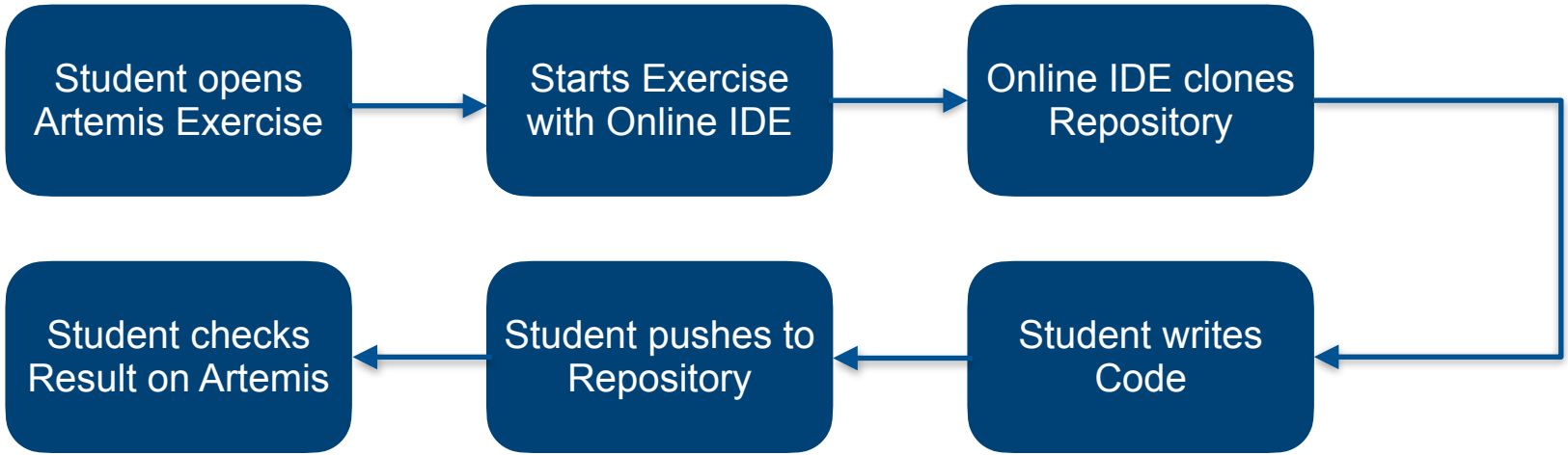


Prewarming (Preview…)

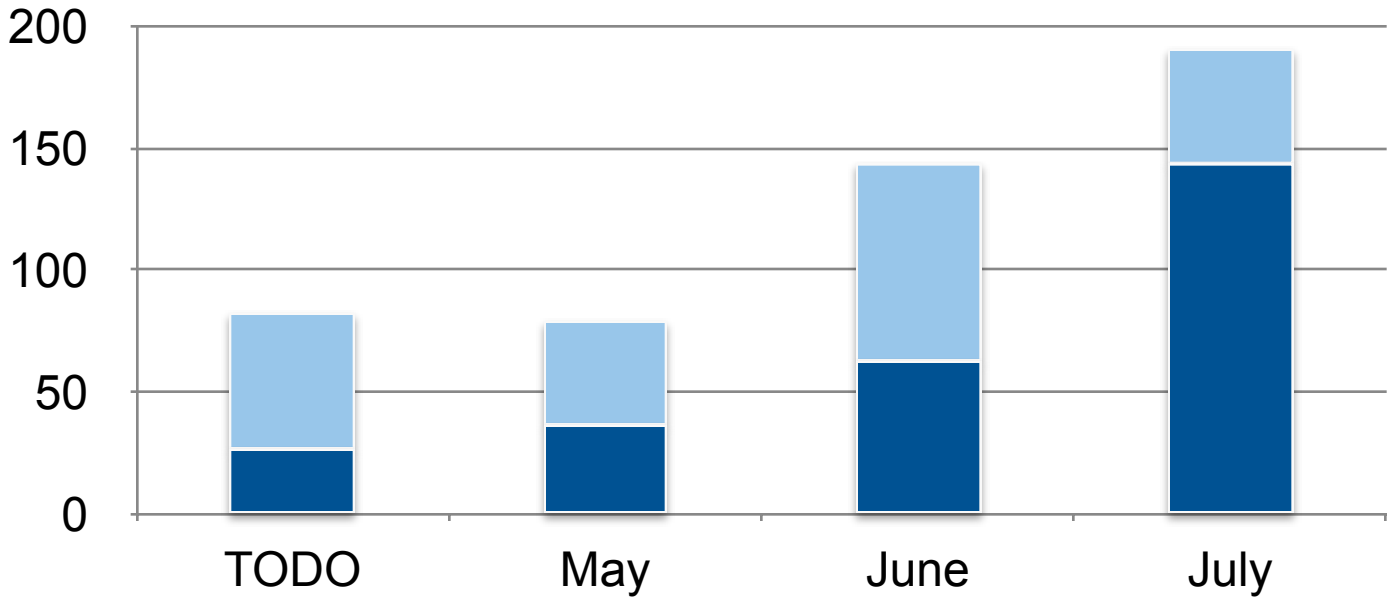


N - Instances

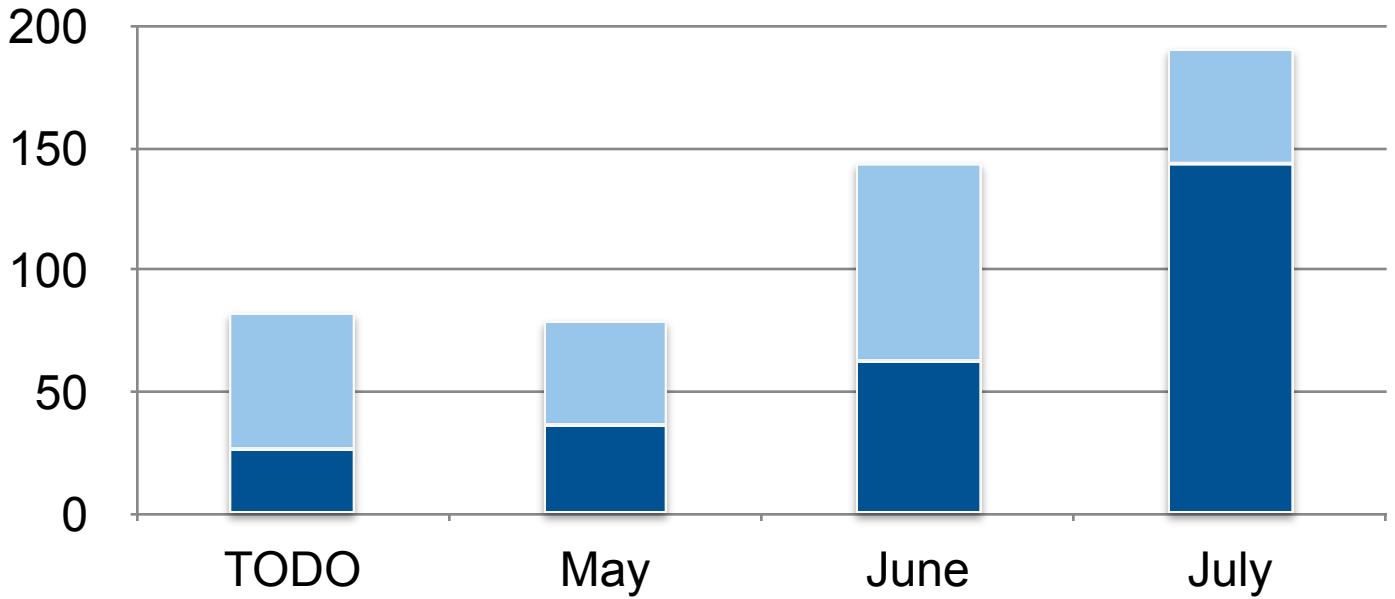
Example Workflow



Evaluation



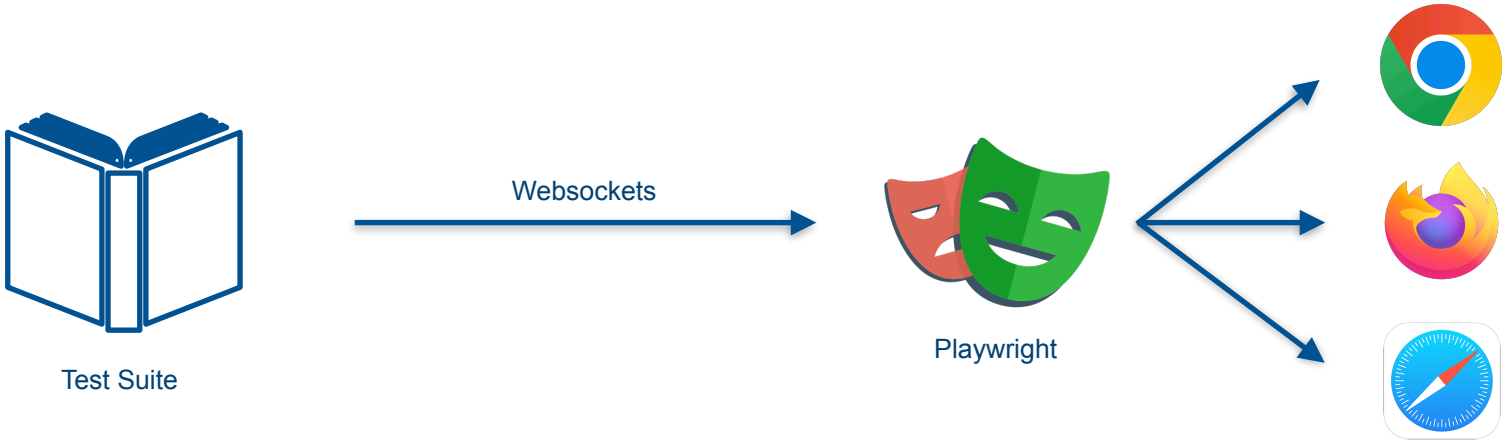
Evaluation



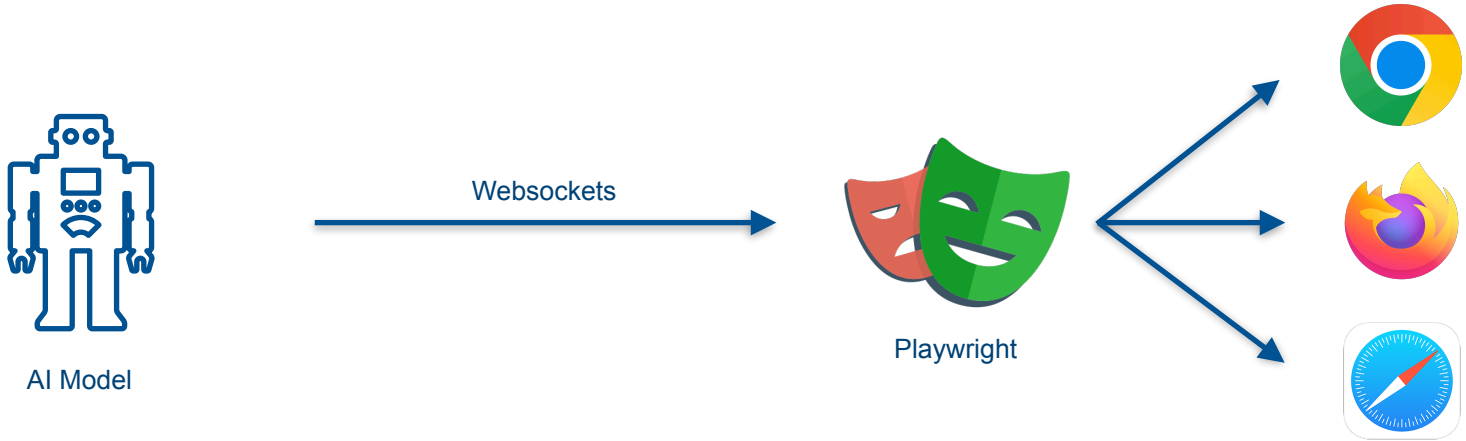
What is MCP

Model Context Protocol

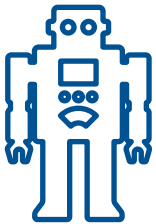
How can we utilize MCP for testing



How can we utilize MCP for testing



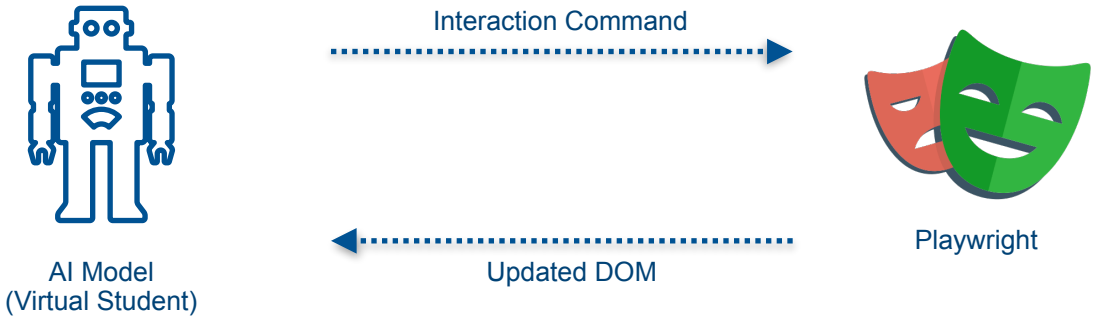
How can we utilize MCP for testing



AI Model

Prompt:
Imagine you are a student with a programming exercise. You are given an Online IDE with all the known functionality. Interact with the Online IDE and behave like a student trying to solve the task.

How can we utilize MCP for testing



Status

- Functionality Tests 
- Load Testing 
- Performance Evaluation 
- MCP Testing 

DEMO

Questions

Thank you for listening!