

J.N.T.U.H. UNIVERSITY COLLEGE OF ENGINEERING HYDERABAD
(Autonomous)

KUKATPALLY, HYDERABAD – 500 085



Certificate

Certified that this is the bonafide record of the practical work done during

the academic year _____ by

Name _____

Roll Number _____ Class _____

in the Laboratory of _____

of the Department of _____

Signature of the Staff Member

Signature of the Head of the Department

Date of Examination _____

Signature of the Examiner/s

Internal Examiner

External Examiner

J.N.T.U.H. UNIVERSITY COLLEGE OF ENGINEERING HYDERABAD
(Autonomous)

KUKATPALLY, HYDERABAD – 500 085

Name _____ **Roll Number** _____

Class _____ *Year* _____ *Laboratory* _____

List of Experiments

[illegible]

1) What is WinRunner explain about it's functionalities.

WinRunner is an automated functional testing tool that was developed by Mercury Interactive (later acquired by HP/Micro Focus). It's primarily used for testing Windows-based applications through GUI automation.

Core Functionalities

Record and Playback Testing WinRunner can record user interactions with an application's interface and then replay those actions automatically. This includes mouse clicks, keyboard inputs, menu selections, and other GUI operations.

Script-Based Testing The tool uses Test Script Language (TSL), a C-like programming language that allows testers to create more sophisticated test scripts beyond simple record/playback. TSL supports variables, loops, conditional statements, and functions.

Object Recognition WinRunner maintains an object repository that maps GUI objects (buttons, text fields, menus, etc.) to logical names. It can identify objects using various properties like class, text, location, and other attributes, making tests more maintainable when the interface changes.

Data-Driven Testing The tool supports parameterized testing where test data can be read from external sources like Excel files, databases, or text files. This allows running the same test with multiple data sets.

Synchronization Points WinRunner can wait for specific conditions before proceeding with test execution, such as waiting for a window to appear, text to change, or objects to become available.

2) Write a Selenium program using Node.js to automate a Google search.

```
const { Builder, By, Key, until } = require('selenium-webdriver');  
const chrome = require('selenium-webdriver/chrome');
```

```
async function googleSearch() {  
  // Set up Chrome driver  
  let driver = await new Builder()  
    .forBrowser('chrome')  
    .setChromeOptions(new chrome.Options())  
    .build();
```

```

try {
  // Navigate to Google
  await driver.get('https://www.google.com');

  // Find the search input, enter query, and submit
  const searchQuery = 'Selenium WebDriver';
  await driver.findElement(By.name('q')).sendKeys(searchQuery, Key.RETURN);

  // Wait for results to load
  await driver.wait(until.elementLocated(By.css('h3')), 10000);

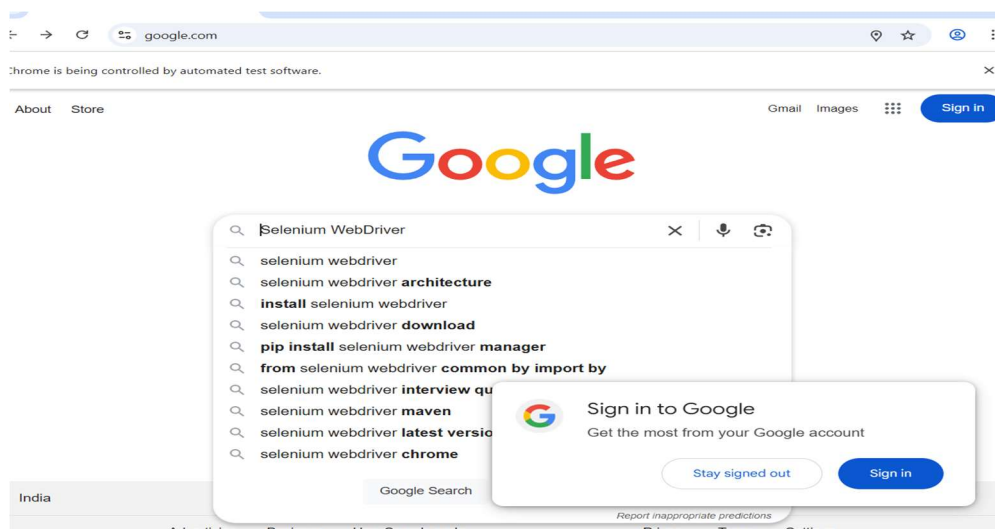
  // Get the title of the first result
  const firstResult = await driver.findElement(By.css('h3'));
  const resultTitle = await firstResult.getText();
  console.log('First result title:', resultTitle);

} catch (error) {
  console.error('Error during search:', error);
} finally {
  // Close the browser
  await driver.quit();
}
}

// Run the function
googleSearch();

```

Output



3) Create a Selenium script in Node.js to open a web page, extract the text

```
const { Builder, By } = require('selenium-webdriver');

// Sleep function using Promise and setTimeout
const sleep = (ms) => new Promise(resolve => setTimeout(resolve, ms));

async function extractHeaderText() {
  let driver = await new Builder().forBrowser('chrome').build();

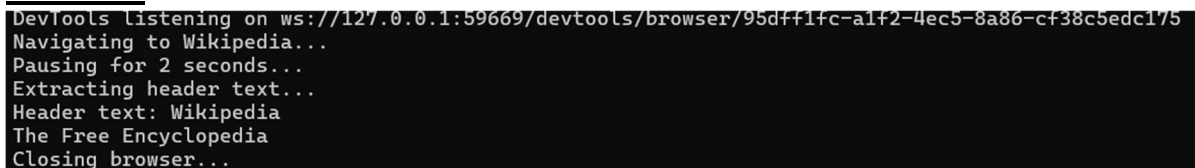
  try {
    console.log('Navigating to Wikipedia...');
    await driver.get('https://www.wikipedia.org');

    console.log('Pausing for 2 seconds...');
    await sleep(2000); // Pause for 2000ms (2 seconds)

    console.log('Extracting header text...');
    const header = await driver.findElement(By.tagName('h1'));
    const text = await header.getText();
    console.log('Header text:', text);
  } catch (error) {
    console.error('Error:', error.message);
  } finally {
    console.log('Closing browser...');
    await driver.quit();
  }
}

extractHeaderText();
```

OUTPUT



```
DevTools listening on ws://127.0.0.1:59669/devtools/browser/95d4f1fc-a1f2-4ec5-8a86-cf38c5edc175
Navigating to Wikipedia...
Pausing for 2 seconds...
Extracting header text...
Header text: Wikipedia
The Free Encyclopedia
Closing browser...
```

4) Write a Selenium script to handle browser navigation (back, Forward, refresh)

```
const { Builder, By } = require('selenium-webdriver');

// Sleep function for timing
const sleep = (ms) => new Promise(resolve => setTimeout(resolve, ms));

async function browserNavigation() {
  let driver = await new Builder().forBrowser('chrome').build();
```

```

try {
  // Step 1: Navigate to Wikipedia
  console.log('Navigating to Wikipedia...');
  await driver.get('https://www.wikipedia.org');
  let title = await driver.getTitle();
  console.log('Current page title:', title);
  await sleep(2000); // Pause for 2 seconds
  // Step 2: Navigate to English Wikipedia
  console.log('Navigating to English Wikipedia...');
  await driver.findElement(By.id('js-link-box-en')).click();
  title = await driver.getTitle();
  console.log('Current page title:', title);
  await sleep(2000);
  // Step 3: Go back
  console.log('Going back...');
  await driver.navigate().back();
  title = await driver.getTitle();
  console.log('Current page title after back:', title);
  await sleep(2000);
  // Step 4: Go forward
  console.log('Going forward...');
  await driver.navigate().forward();
  title = await driver.getTitle();
  console.log('Current page title after forward:', title);
  await sleep(2000);
  // Step 5: Refresh page
  console.log('Refreshing page...');
  await driver.navigate().refresh();
  title = await driver.getTitle();
  console.log('Current page title after refresh:', title);
} catch (error) {
  console.error('Error:', error.message);
} finally {
  console.log('Closing browser...');
  await driver.quit();
}
}
browserNavigation();

```

OUTPUT

```
DevTools listening on ws://127.0.0.1:59713/devtools/browser/9f95cb3e-61e7-40d8-989c-400346478bc7
Navigating to Wikipedia...
Current page title: Wikipedia
Navigating to English Wikipedia...
Current page title: Wikipedia, the free encyclopedia
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1749656254.216091    3224 voice_transcription.cc:58] Registering VoiceTranscriptionCapability
Going back...
Current page title after back: Wikipedia
Going forward...
Current page title after forward: Wikipedia, the free encyclopedia
Refreshing page...
Current page title after refresh: Wikipedia, the free encyclopedia
Closing browser...
```

5) Write a Node.js program using Selenium to count the total number of hyperlinks on the page

```
const { Builder, By } = require('selenium-webdriver');
// Sleep function for timing
const sleep = (ms) => new Promise(resolve => setTimeout(resolve, ms));
async function countHyperlinks() {
  let driver = await new Builder().forBrowser('chrome').build();
  try {
    console.log('Navigating to Wikipedia...');
    await driver.get('https://www.wikipedia.org');
    await sleep(2000); // Pause for 2 seconds to ensure page loads
    console.log('Counting hyperlinks...');
    const links = await driver.findElements(By.tagName('a'));
    const totalLinks = links.length;
    console.log(`Total number of hyperlinks: ${totalLinks}`);
  } catch (error) {
    console.error('Error:', error.message);
  } finally {
    console.log('Closing browser...');
    await driver.quit();
  }
}
```

countHyperlinks();

OUTPUT

```
DevTools listening on ws://127.0.0.1:59768/devtools/browser/8d8c6fe6-a652-4c57-b74d-1a6f99bc615d
Navigating to Wikipedia...
Counting hyperlinks...
Total number of hyperlinks: 371
Closing browser...
```

6) Develop a Selenium script to automate login testing

```
const { Builder, By } = require('selenium-webdriver');
// Sleep function for timing
const sleep = (ms) => new Promise(resolve => setTimeout(resolve, ms));
```

```

async function loginTest() {
  let driver = await new Builder().forBrowser('chrome').build();
  try {
    console.log('Navigating to login page...');
    await driver.get('https://the-internet.herokuapp.com/login');
    await sleep(2000); // Pause for 2 seconds to ensure page loads
    console.log('Entering credentials...');
    await driver.findElement(By.id('username')).sendKeys('tomsmith');
    await driver.findElement(By.id('password')).sendKeys('SuperSecretPassword!');
    await driver.findElement(By.css('button[type="submit"]')).click();
    await sleep(2000); // Pause for login processing
    console.log('Checking login result...');
    const successMessage = await driver.findElement(By.id('flash')).getText();
    if (successMessage.includes('You logged into a secure area!')) {
      console.log('Login successful!');
    } else {
      console.log('Login failed:', successMessage);
    }
  } catch (error) {
    console.error('Error during login test:', error.message);
  } finally {
    console.log('Closing browser...');
    await driver.quit();
  }
}
loginTest();

```

OUTPUT

```

DevTools listening on ws://127.0.0.1:59889/devtools/browser/a66d8e55-f9de-4f11-b998-0a63d80b6f22
Navigating to login page...
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1749656546.855024    17808 voice_transcription.cc:58] Registering VoiceTranscriptionCapability
Entering credentials...
Checking login result...
Login successful!
Closing browser...

```

7) Write Selenium script that captures a screenshot

```

const { Builder, By } = require('selenium-webdriver');
const fs = require('fs');
// Sleep function for timing
const sleep = (ms) => new Promise(resolve => setTimeout(resolve, ms));
async function captureScreenshot() {
  let driver = await new Builder().forBrowser('chrome').build();

  try {

```



```
console.log('Navigating to Wikipedia...');
await driver.get('https://www.wikipedia.org');
await sleep(2000); // Pause for 2 seconds to ensure page loads

console.log('Capturing screenshot...');
const screenshot = await driver.takeScreenshot();
fs.writeFileSync('wikipedia_screenshot.png', screenshot, 'base64');
console.log('Screenshot saved as wikipedia_screenshot.png');

} catch (error) {
  console.error('Error during screenshot capture:', error.message);
} finally {
  console.log('Closing browser...');
  await driver.quit();
}
}
captureScreenshot();
```

OUTPUT

```
DevTools listening on ws://127.0.0.1:59949/devtools/browser/e8aac9f4-616e-401c-aeec-039ff1279220
Navigating to Wikipedia...
Capturing screenshot...
Screenshot saved as wikipedia_screenshot.png
Closing browser...
```