VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JNANA SANGAMA", Belagavi-590018

Mini Project Report on

"Automation Testing on campus.uno/sjbit website using Selenium "

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR

THE AWARD OF DEGREE OF

BACHELOR OF ENGINEERING IN
INFORMATION SCIENCE AND ENGINEERING

SUBMITTED BY

NISHANT MANJUNATH HEGDE (1JB21IS073)

Under the Guidance of

Mrs. Shivani N
Assistant Professor,
Dept. of ISE, SJBIT
Bengaluru-60

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING
SJB INSTITUTE OF TECHNOLOGY

BGS HEALTH AND EDUCATION CITY, KENGERI, BENGALURU-560060, KARNATAKA, INDIA.

2023-2024

|| Jai Sri Gurudev ||
Sri Adichunchanagiri Shikshana Trust ®

SJB INSTITUTE OF TECHNOLOGY

BGS Health & Education City, Kengeri, Bengaluru – 560 060

Department of Information Science & Engineering

CERTIFICATE

Certified that the Mini project work entitled "**Automation Testing on campus.uno/sjbit website using Selenium"** carried out by **NISHANT MANJUNATH HEGDE** bearing **USN 1JB21IS073 is** a bonafide student of **SJB Institute of Technology** in partial fulfillment for 6th semester Software Testing Laboratory with Mini Project in **INFORMATION SCIENCE AND ENGINEERING** of the **VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI** during the academic year **2023-24.** It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The mini project report has been approved as it satisfies the academic requirements in respect of Mini Project prescribed for the said degree.

Mrs. Shivani N **Assistant Professor** **Dept. of ISE, SJBIT**	**Dr. Shashidhara HR** **Professor & Head** **Dept. of ISE, SJBIT**

EXTERNAL VIVA

Name of the Examiners	**Signature with Date**
1. _____	_____
2. _____	_____

ACKNOWLEDGEMENT

I would like to express my profound thanks to **His Divine Soul Padmabhushan Sri Sri Sri Dr. Balagangadharanatha MahaSwamiji** and His Holiness **Jagadguru Sri Sri Sri Dr. Nirmalanandanatha Swamiji** for providing me an opportunity to pursue my academics in this esteemed institution.

I would also like to express my profound thanks to **Revered Sri Sri Dr. Prakashnath Swamiji**, **Managing Director**, **SJB Institute of Technology**, for his continuous support in providing amenities to carry out this mini project in this admired institution.

I express our gratitude to **Dr. K. V. Mahendra Prashanth**, Principal, SJB Institute of Technology, for providing excellent facilities and academic ambience, which have helped me in satisfactory completion of mini project work.

I extend our sincere thanks to **Dr. Babu N V**, Academic Dean, SJB Institute of Technology for providing us constant support throughout the period of Mini Project.

I extend our sincere thanks to **Dr. Shashidhara H.R**, Professor & Head, Dept. of Information Science and Engineering, for providing us invaluable support throughout the period of Mini Project work.

I express our heartfelt gratitude to our guide**, Mrs. Shivani N,** Assistant Professor, Dept. of Information Science and Engineering for her valuable guidance, suggestions and encouragement during the entire period of our mini project work.

Finally, I take this opportunity to extend my earnest gratitude and respect to our parents, Teaching & Non-teaching staffs of the department, the library staff and all our friends, for their continuous support and encouragement.

NISHANT MANJUNATH HEGDE
(1JB21IS073)

DECLARATION

I hereby declare that the entire work embodied in this Mini Project report has been carried out under the supervision of Mrs. Shivani N, Assistant Professor, Dept. of Information Science and Engineering, SJB Institute of Technology in partial fulfilment for the award of "BACHELOR OF ENGINEERING" in INFORMATION SCIENCE AND ENGINEERING as prescribed by VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI during the academic year 2023-24.

NISHANT MANJUNATH HEGDE
(1JB21IS073)

ABSTRACT

The document presents a comprehensive exploration of Selenium, a powerful framework for web application automation. It begins with an in-depth introduction, highlighting Selenium's core components, key features, and advantages in software testing and automation. Detailed software and hardware requirements are provided to ensure optimal performance of Selenium scripts. The project section offers a practical demonstration of Selenium's capabilities, focusing on automating tasks within a campus management system. It includes a step-by-step setup process, code walk through, and explanations of crucial functions. The script showcases automation of various tasks including login, navigation, document downloads, fee payment simulations, attendance checks, and accessing the proctor system. Through code snippets and figures, the document illustrates Selenium's effectiveness in streamlining routine administrative tasks, significantly improving efficiency and reducing time investment. The results and discussion section quantifies these benefits, providing concrete evidence of Selenium's value in real-world applications. The conclusion acknowledges the changing landscape of web interactions, particularly the increasing implementation of security measures against automated access. It proposes a forward-thinking approach, suggesting a hybrid strategy that combines Selenium's browser automation capabilities with API usage. This balanced method addresses the challenges posed by modern web security while maintaining the benefits of automation. Overall, this document serves as a comprehensive guide for understanding and implementing Selenium in web automation tasks. It offers both theoretical knowledge and practical insights, making it a valuable resource for developers and testers looking to leverage Selenium's capabilities in their projects.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction to Selenium

Selenium is an open-source framework used for automating web applications across different browsers and platforms. It is a powerful tool for testing web applications, allowing developers and testers to write tests in various programming languages such as Java, C#, Python, Ruby, and JavaScript.

1.2 Components of Selenium

1. Selenium WebDriver:

The core component of Selenium that directly interacts with the web browser.

Supports multiple browsers (Chrome, Firefox, Safari, Edge, etc.).

Allows you to write test scripts in various programming languages.

2. Selenium IDE (Integrated Development Environment):

A browser extension (available for Firefox and Chrome) for record-and-playback of interactions.

Suitable for beginners to create quick test cases without deep programming knowledge.

3. Selenium Grid:

Used for running tests on multiple machines and browsers simultaneously.

Supports parallel execution, reducing the time required for testing.

4. Selenium RC (Remote Control):

An older tool that is now deprecated and replaced by WebDriver.

Was used to write automated web application UI tests.

1.3 Key Features of Selenium:

- **Cross-Browser Testing:** Supports multiple browsers, ensuring compatibility and functionality across different environments.
- **Cross-Platform Testing:** Runs on various operating systems, including Windows, macOS, and Linux.
- **Language Support:** Offers flexibility with multiple programming languages, allowing teams to write tests in their preferred language.
- **Integration:** Integrates with various testing frameworks (JUnit, TestNG) and CI/CD tools (Jenkins, Maven) for continuous integration and delivery.
- **Community Support:** A large and active community provides extensive documentation, tutorials, and support.

1.4 Advantages of Selenium:

- **Open Source:** Free to use, with no licensing costs.
- **Flexibility:** Can be customized and extended to fit specific testing requirements.
- **Re-usability:** Test scripts can be reused across different projects and environments.

- **Scalability:** Selenium Grid enables scaling up test execution across multiple machines and browsers.

1.5 Basic Workflow of Selenium WebDriver

1. Setup Environment:

Install the programming language (e.g., Python, Java).

Install Selenium WebDriver.

Install the browser driver (e.g., ChromeDriver for Chrome).

2. Write Test Script:

Use Selenium WebDriver API to write test scripts for web interactions.

3. Run Tests:

Execute the scripts to run tests on the desired browser and platform.

4. Analyze Results:

Review the test results and logs to identify any issues or failures.

1.6 Problem Statement

Automating the testing of the campus.uno/Sjbit website using Selenium to ensure seamless functionality. This includes verifying user authentication, navigation, course management, attendance tracking, assignments, communication, and profile management.

1.7 Objective

To automate the functional and regression testing of the campus management system hosted at campus.uno/sjbit using Selenium WebDriver. The aim is to ensure that the web application performs as expected across different browsers and devices, providing a seamless experience for students, faculty, and administrators.

CHAPTER 2

REQUIREMENTS SPECIFICATION

There are no systems which can run without hardware and software requirements. So for any system in this world, the hardware and software requirements are the most basic necessity to work. For each and every system there will be different hardware and software requirements. So we shall see the particular requirement of our system. Software requirements concerned with portraying programming asset prerequisites and essentials that should be introduced on a computer to give best working of an application.

2.1 Software Requirements

Operating System:

Windows 7 or later / macOS / Linux (any modern distribution).

Programming Languages:

Python 3.x / Java 8 or later / C# / Ruby / JavaScript.

Selenium WebDriver:

Selenium WebDriver library compatible with the chosen programming language.

Browser Drivers:

ChromeDriver for Google Chrome.

GeckoDriver for Mozilla Firefox.

EdgeDriver for Microsoft Edge.

SafariDriver for Safari (macOS).

IDE (Integrated Development Environment):

PyCharm / IntelliJ IDEA / Eclipse / Visual Studio Code / Visual Studio.

Build Tools:

Maven / Gradle (for Java projects), Pip (for Python projects).

Testing Frameworks:

TestNG / JUnit (for Java), pytest / unittest (for Python), NUnit (for C#).

Continuous Integration Tools:

Jenkins / Travis CI / CircleCI / GitHub Actions.

Additional Libraries:

Browser-specific WebDriver libraries.

Selenium Grid for distributed testing (optional).

2.2 Hardware Requirements

Processor: Multi-core processor (Intel i5 or equivalent).

Memory: Minimum 8 GB RAM (16 GB recommended for running multiple browsers/tests simultaneously)

Storage: Minimum 256 GB SSD (Solid State Drive) for fast read/write operations. Additional storage may be required based on the volume of test data and logs.

Network:

Stable internet connection for accessing the web application, downloading dependencies, and integrating with CI/CD tools.

Display:

Minimum 1080p resolution for viewing test execution and results clearly.

CHAPTER 3

INTRODUCTION TO PROJECT

3.1 Overview

This script automates various actions on a web page using Selenium WebDriver. It performs operations such as logging in, navigating menus, filling forms, and interacting with payment processes. The script is designed to automate interactions with a specific educational institution's portal, performing tasks like submitting grievances, making fee payments, and accessing attendance records.

3.2 Setup

1. Install Selenium package. Using,

pip install selenium

2. Download the appropriate version of ChromeDriver from ChromeDriver Downloads and place it in the project directory (./chromedriver.exe).

3.3 Code Walk through

3.3.1 Initialization and Setup

Imports necessary modules. Sets up the Chrome WebDriver. Defines a helper function highlight_element() to visually highlight interacted elements.

```
1   from selenium import webdriver
2   from selenium.webdriver.common.keys import Keys
3   from selenium.webdriver.common.by import By
4   import time
5   from selenium.webdriver.chrome.service import Service
6
7   # Initialize Chrome WebDriver (replace with Firefox WebDriver if needed)
8   chromedriver_path = "./chromedriver.exe"  # Adjust for your path
9   service = Service(chromedriver_path)
10  driver = webdriver.Chrome(service=service)
11
12  def highlight_element(element):
13      driver.execute_script("arguments[0].setAttribute('style', 'border: 6px solid red;');", element)
14      time.sleep(2)
15      driver.execute_script("arguments[0].setAttribute('style', 'border: 0px;');", element)
16
17  # Open campus.uno/sjbit
18  driver.get("https://campus.uno/sjbit")#use ur website link
19  driver.maximize_window()
20  time.sleep(2)
21
```

Figure 3.1: Initialization & Setup Code Snippet

3.3.2 Login Process

It locates the login fields using XPATH. Enters username and password. Submits the login form.

```
email = driver.find_element(By.XPATH, '//*[@id="LoginId"]')
highlight_element(email)
email.send_keys("1JB21IS073")
time.sleep(1)
pwd = driver.find_element(By.XPATH, '//*[@id="Password"]')
highlight_element(pwd)
pwd.send_keys("HEGDE@581322")
time.sleep(1)
proceed = driver.find_element(By.XPATH, '//*[@id="intro"]/div/div/div[2]/section/footer/button')
highlight_element(proceed)
proceed.click()
time.sleep(3)

```

Figure 3.2: Login Process Code Snippet

3.3.3 Navigation and Interaction

The script clicks through menus, selects options, and fills out forms according to the task requirements.

```
menu = driver.find_element(By.XPATH, '/html/body/nav/div[1]/ul/li/a')
highlight_element(menu)
menu.click()
time.sleep(3)
```

Figure 3.3: Navigation & Interaction Code Snippet

3.3.4 Course Materials Download

Navigates through the menu to the course section. Selects a specific course (SEPM). Downloads course materials.

```
#notes download
cource = driver.find_element(By.XPATH, '/html/body/nav/div[1]/ul/li/ul/li/ul/li[13]/a')
highlight_element(cource)
cource.click()
time.sleep(3)
sepm = driver.find_element(By.XPATH, '/html/body/div[1]/form/div[3]/div[1]/div/a[1]')
highlight_element(sepm)
sepm.click()
time.sleep(3)
dwd = driver.find_element(By.XPATH, '/html/body/div[1]/form/div[3]/div[2]/div[2]/div[5]/a[1]')
highlight_element(dwd)
dwd.click()
time.sleep(3)
```

Figure 3.4: Course Materials Download Code Snippet

3.3.5 Fee Payment Process

Navigates to the fee payment section. Initiates a new transaction. Fills in payment details (amount, payment type). Simulates canceling the payment.

```
payment = driver.find_element(By.XPATH, '//*[@id="tab-1"]/div/div[2]/div[5]/div/a[3]')
highlight_element(payment)
payment.click()
time.sleep(5)
amt = driver.find_element(By.XPATH, '//*[@id="tab-1"]/div/div[2]/div[3]/div/input')
highlight_element(amt)
amt.send_keys("1680")
time.sleep(1)
opt = driver.find_element(By.XPATH, '//*[@id="tab-1"]/div/div[2]/div[3]/div/input')
highlight_element(opt)
opt.click()
time.sleep(1)
fee = driver.find_element(By.XPATH, '//*[@id="tab-1"]/div/div[2]/div[4]/div/select/option[2]')
highlight_element(fee)
fee.click()
time.sleep(1)
make_payment = driver.find_element(By.XPATH, '//*[@id="tab-1"]/div/div[2]/div[5]/div/a[1]')
highlight_element(make_payment)
make_payment.click()
time.sleep(1)
note = driver.find_element(By.XPATH, '//*[@id="tab-1"]/div/div[3]/div[1]/div/div[2]/div/input')
highlight_element(note)
note.send_keys("EXAM FEE")
time.sleep(1)
term = driver.find_element(By.XPATH, '//*[@id="tab-1"]/div/div[3]/div[2]/div/input')
highlight_element(term)
term.click()
time.sleep(1)
cancel = driver.find_element(By.XPATH, '//*[@id="tab-1"]/div/div[3]/div[3]/div/a[2]')
highlight_element(cancel)
cancel.click()
time.sleep(1)
```

Figure 3.5: Fees Payment Code Snippet

3.3.6 Attendance Status Check

Accesses the attendance section. Selects a specific semester. Retrieves attendance information.

```
#check attendence status
menu = driver.find_element(By.XPATH, '/html/body/nav/div[1]/ul/li/a')
highlight_element(menu)
menu.click()
time.sleep(3)
attendence = driver.find_element(By.XPATH, '/html/body/nav/div[1]/ul/li/ul/li/ul/li[4]/a')
highlight_element(attendence)
attendence.click()
time.sleep(3)
option = driver.find_element(By.XPATH, '//*[@id="dropdown-class-section"]/button')
highlight_element(option)
option.click()
time.sleep(3)
sem2 = driver.find_element(By.XPATH, '//*[@id="dropdown-class-section"]/div/ul/li[5]/a')
highlight_element(sem2)
sem2.click()
time.sleep(10)
go = driver.find_element(By.XPATH, '//*[@id="btn-go"]')
highlight_element(go)
go.click()
time.sleep(10)

```

Figure 3.6: Attendance Status Check Code Snippet

3.3.7 Proctor System View

Navigates to the proctor system. Cycles through different sections: Activities, Meetings, Grading, Academic, and Interaction.

```
#view proctor system
menu = driver.find_element(By.XPATH, '/html/body/nav/div[1]/ul/li/a')
highlight_element(menu)
menu.click()
time.sleep(3)
proctor = driver.find_element(By.XPATH, '/html/body/nav/div[1]/ul/li/ul/li/ul/li[19]')
highlight_element(proctor)
proctor.click()
time.sleep(3)
Activities = driver.find_element(By.XPATH, '/html/body/div[1]/form/div[4]/div/div[1]/div/a[1]')
highlight_element(Activities)
Activities.click()
time.sleep(6)
meeting = driver.find_element(By.XPATH, '/html/body/div[1]/form/div[4]/div/div[1]/div/a[2]')
highlight_element(meeting)
meeting.click()
time.sleep(6)
grading = driver.find_element(By.XPATH, '/html/body/div[1]/form/div[4]/div/div[1]/div/a[3]')
highlight_element(grading)
grading.click()
time.sleep(6)
academic = driver.find_element(By.XPATH, '/html/body/div[1]/form/div[4]/div/div[1]/div/a[4]')
highlight_element(academic)
academic.click()
time.sleep(6)
Interaction = driver.find_element(By.XPATH, '/html/body/div[1]/form/div[4]/div/div[1]/div/a[5]')
highlight_element(Interaction)
Interaction.click()
time.sleep(6)
```

Figure 3.7: Proctor System View Code Snippet

3.3.8 Logout

Performs the logout action.

```
#logout
out = driver.find_element(By.XPATH, '/html/body/nav/div[1]/ul/li/ul/li/ul/li[22]')
highlight_element(out)
out.click()
time.sleep(3)
```

Figure 3.8: Logout Code Snippet

3.4 Important Functions

highlight_element(element)

This function visually highlights an element on the page by temporarily adding a red border. It's used throughout the script to visually indicate which element is being interacted with.

```
def highlight_element(element):
    driver.execute_script("arguments[0].setAttribute('style', 'border: 6px solid red;');", element)
    time.sleep(2)
    driver.execute_script("arguments[0].setAttribute('style', 'border: 0px;');", element)

```

Figure 3.9: highlight_element() function Code Snippet

3.5 Notes and Considerations

- **Credentials:** The script includes hardcoded login credentials. For security reasons, it's recommended to use environment variables or a secure configuration file instead.
- **Wait Times:** The script uses time.sleep() for waiting. In a production environment, it's better to use Selenium's built-in waits (e.g., WebDriverWait) for more reliable automation.
- **Error Handling:** The script doesn't include error handling. In a real-world scenario, try-except blocks should be added to handle potential exceptions.
- **Commented Code:** There's a section of commented-out code related to a grievance form. This could be uncommented and used if needed.
- **XPath Selectors:** The script heavily relies on XPath selectors. These might break if the website's structure changes. Consider using more robust selectors where possible.
- **Chromedriver Path:** The script assumes the chromedriver is in the same directory. Adjust the path if necessary.
- **Website Specificity:** This script is tailored for a specific website (campus.uno/sjbit). It may need modifications to work with other sites.

CHAPTER 4

SNAPSHOTS

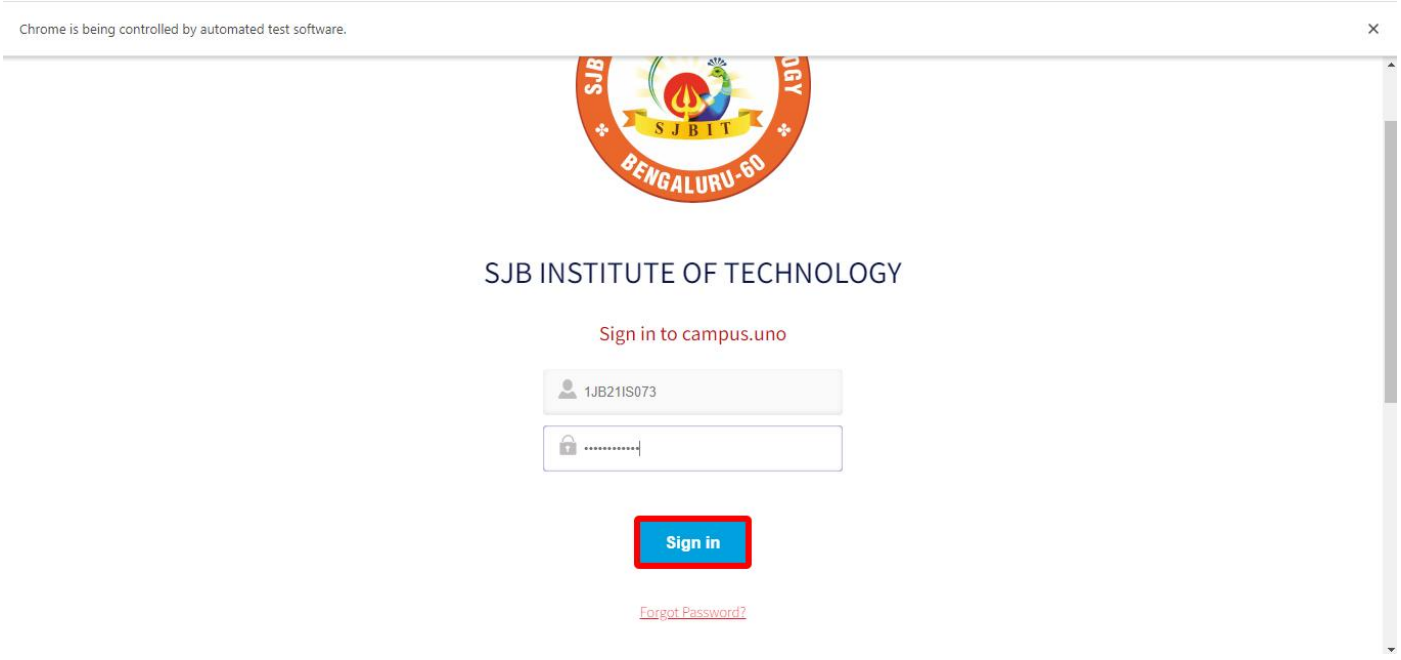

Figure 4.1: Login Process

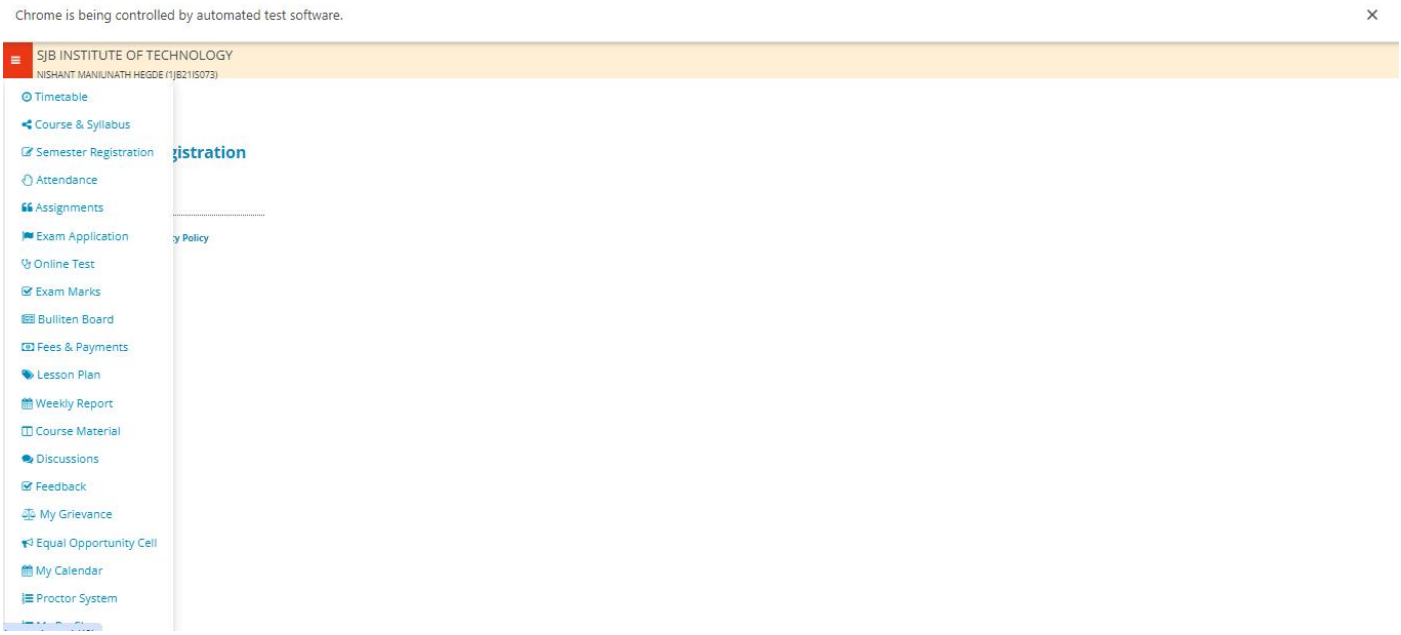

Figure 4.2: Navigation & Interaction

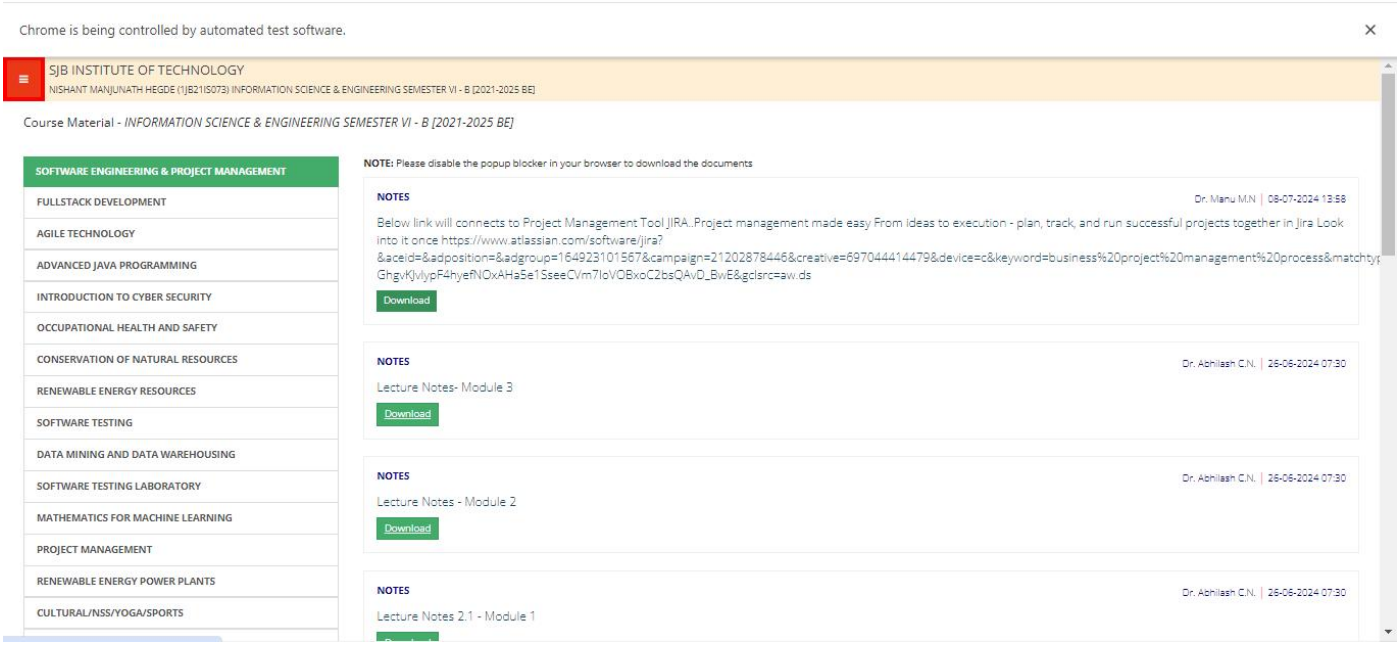

Figure 4.3: Course Materials Download

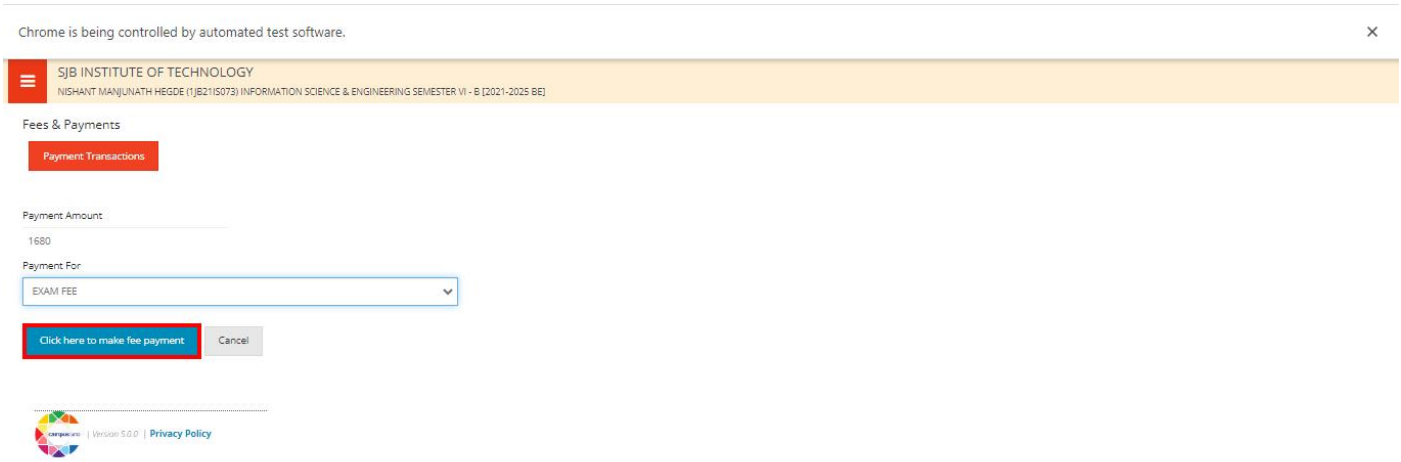

Figure 4.4: Fees Payment

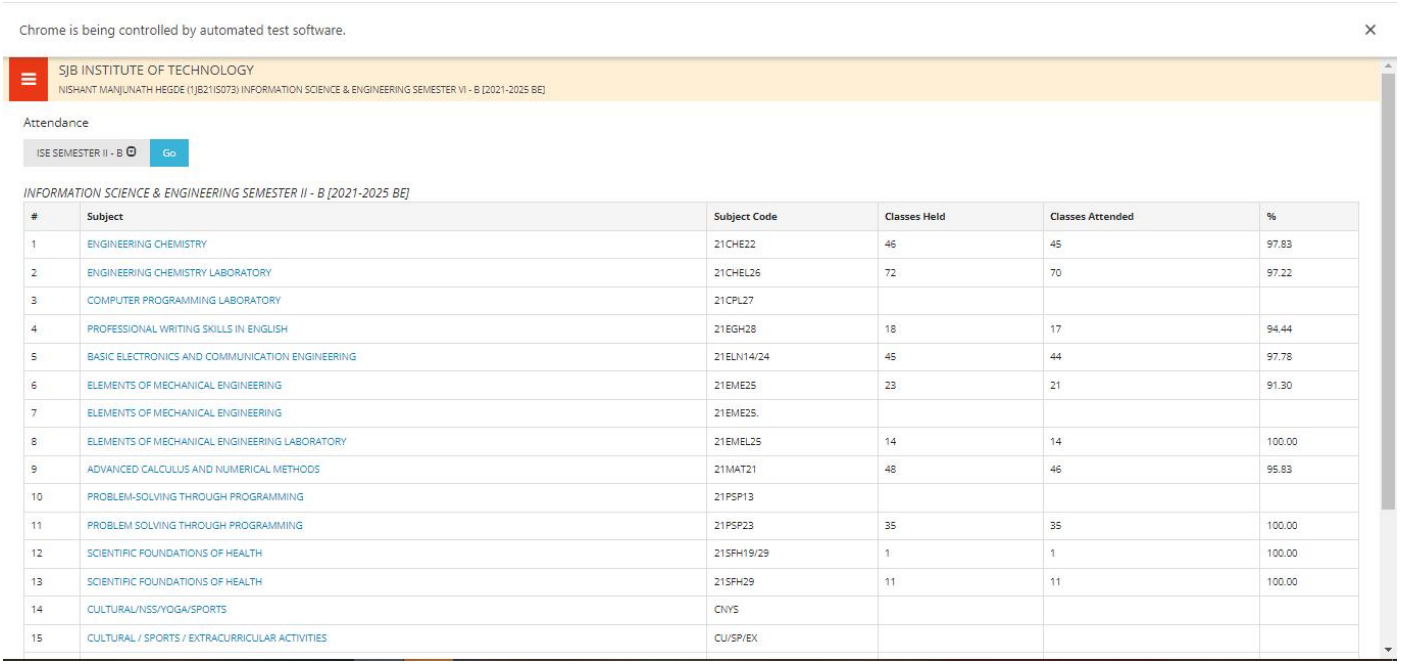

Figure 4.5: Attendance Status Check

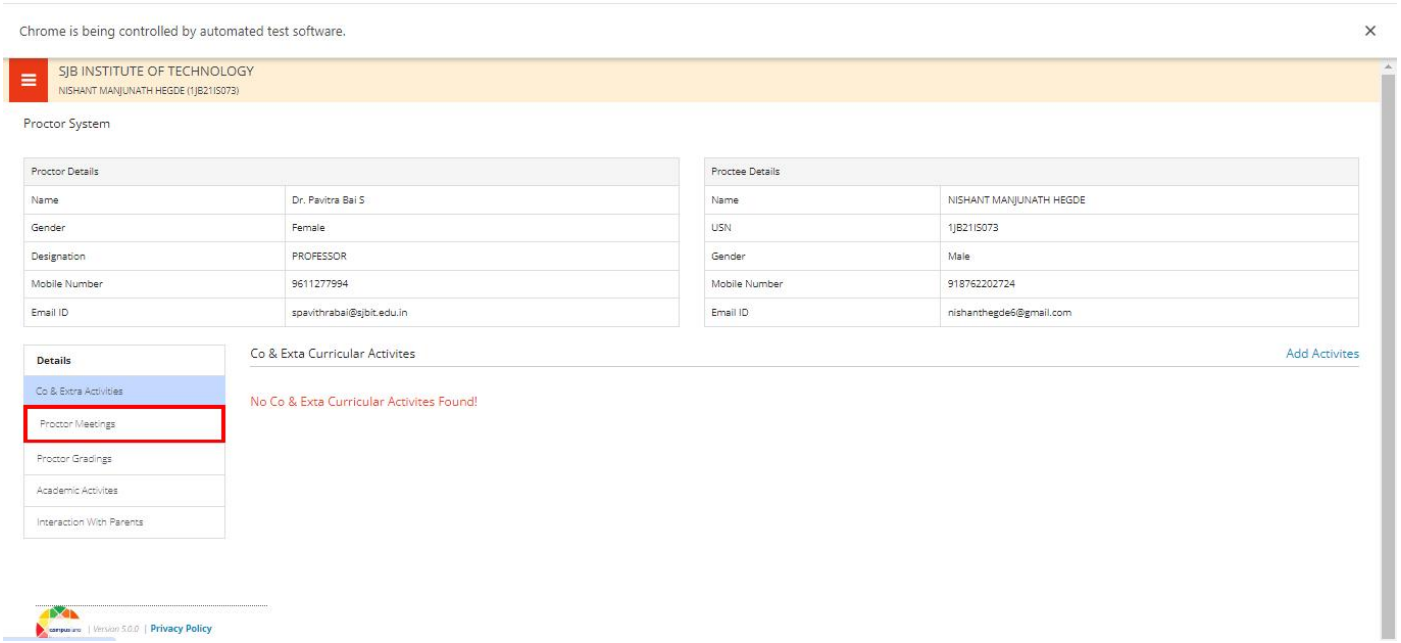

Figure 4.6: Proctor System View

CHAPTER 5

CONCLUSION

This Selenium script successfully automates key tasks in a campus management system, demonstrating significant potential for time savings and efficiency improvements in routine student and administrative tasks. It handles login, navigation, document download, fee payment simulation, attendance checking, and proctor system access.

While Selenium has been a cornerstone for web automation, web scraping, and browser-based tasks, the landscape of web interaction is evolving. Many websites now implement stringent measures against automated access, including sophisticated CAPTCHA systems and rate limiting, to protect user data and prevent unauthorized scraping.

In response to these challenges and to balance data accessibility with security, many platforms are shifting towards providing official APIs. These APIs offer a more controlled, efficient, and ethical means of data retrieval and interaction. They allow developers to access necessary data without resorting to potentially disruptive web scraping techniques.

This transition highlights the importance of adaptability in automation strategies. While Selenium remains valuable for certain tasks, developers increasingly need to explore API-based solutions for data mining and web interaction. This approach not only ensures compliance with platform policies but often provides more reliable and scalable access to data.

Moving forward, a hybrid approach combining browser automation for unavoidable UI interactions and API usage for data retrieval may represent the most effective strategy in many scenarios.

REFERENCES

[1] Conboy, K., Coyle, Sh., Wang, X. 2010. People over Process: Key challenges in Agile Development, IEEE Software, Volume: 28, Issue: 4, (July-Aug. 2011).

[2] Limaye M. G., Software Testing 2009, Tata McGraw-Hill Education.

[3] Introduction to Selenium WebDriver --

Selenium Tutorial #8, 2017, http://www.softwaretestinghelp.com/selenium-webdriver-selenium-tutorial-8/

[4] Suman Madan, Aakriti Kakkar. Test Automation as framework for web applications. International Journal of computer science and engineering. 2017;

[5] https://www.softwaretestinghelp.com/selenium-tutorial-1/

[6] https://btreesystems.com/blog/top-10-selenium-project-with-code-examples/#5

[7] https://www.phind.com/search?cache=mrg85yds2vbgfs22ydldpo2z