

AUTOMATION TESTING ACTITIME
Project based experience learning program

ACTITIME Automation Testing Project

ABOUT ACTITIME :

ActiTIME is a website deals with product for Time Tracking Software for cost-effective projects. actiTIME boosts performance, track work progress and don't leave any billable second untracked and records billable hours for a project resource for easy monitoring the work hours.

we can generate reports based on hours logged and also, we can integrate with other platforms like google calendar, Zapier etc.

actiTIME is not the only option for Time Tracking Software. Explore other competing options and alternatives. Time Tracking Software is a widely used technology, and many people are seeking productive, high quality software solutions with ease of completing timesheets, invoice creation and delivery, and time tracking. Other important factors to consider when researching alternatives to actiTIME include tasks and design. We have compiled a list of solutions that reviewers voted as the best overall alternatives and competitors to actiTIME, including Toggl Track, BigTime, QuickBooks Time, and Harvest.

The Actitime Manual Testing Project aims to thoroughly test the web-based time-tracking and project management software, Actitime. The project team will conduct comprehensive testing of Actitime's features and functionalities to identify any defects, inconsistencies, or usability issues that might impact user experience.

Table of Contents

- Introduction
- Project Overview
- Getting Started
 - ✓ Pre requisites
 - ✓ Installation
 - ✓ Project Setup
- Project Structure
- Writing Test Cases
- Running Tests
- Test Reporting
- Advantages & Disadvantages
- Application
- Troubleshooting
- Conclusion

Introduction

The actiTIME project is a Python automation project that utilizes the Selenium framework for automating web interactions. This documentation provides an overview of the project, instructions for installation and configuration, details about the project structure, and examples of test cases.

actiTIME is a web-based timesheet for companies of any size and any business type. It allows you to enter time spent on different work assignments, register time offs and sick leaves, and then create detailed reports covering almost any management or accounting needs.

Project Overview

The actiTIME project aims to automate various tasks on the Bluestone website. It leverages Selenium, a powerful web automation tool, to interact with web elements and perform actions such as clicking buttons, filling forms, and extracting data.

actiTIME manual testing is the process of verifying the functionality and quality of actiTIME software by executing test cases manually without using any automated tools. It involves checking the features such as time tracking, project management, billing, leave management, and integrations of actiTIME according to the user requirements and expectations. It also involves reporting and re-testing any bugs or issues that are found during the testing process. actiTIME manual testing helps to ensure that the software is bug-free and meets the customer needs.

Milestone 1

Getting Started

Explain the prerequisites and setup required to run the project.

Activity 1: Prerequisites

Before getting started with the actiTIME project, ensure you have the following prerequisites:

List the software and tools that need to be installed before running the project, such as:

- **Python (3.x)**
- **Selenium (3.x)**
- **Pytest (Latest Version)**
- **Any other dependencies**

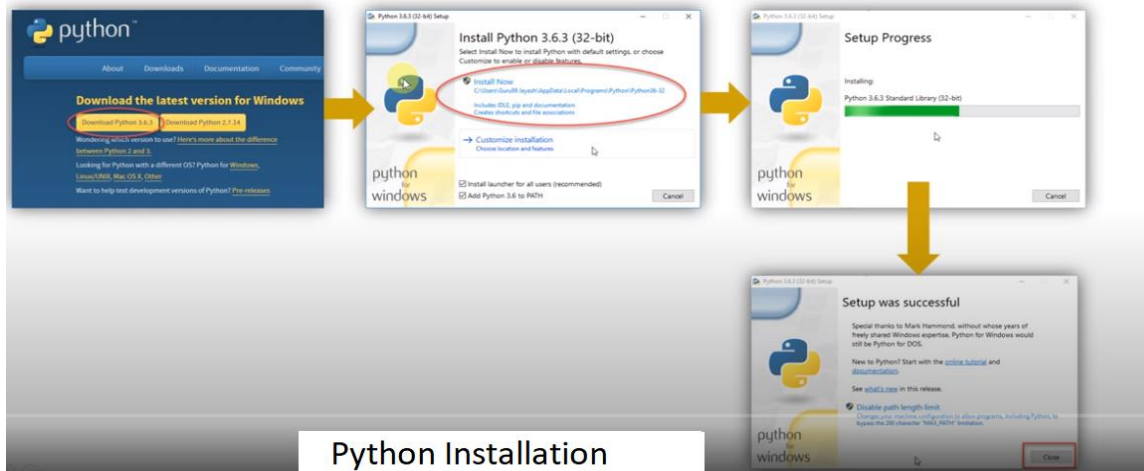
Activity 2 : Installation

To install the necessary dependencies for the actiTIME automation project, follow these steps:

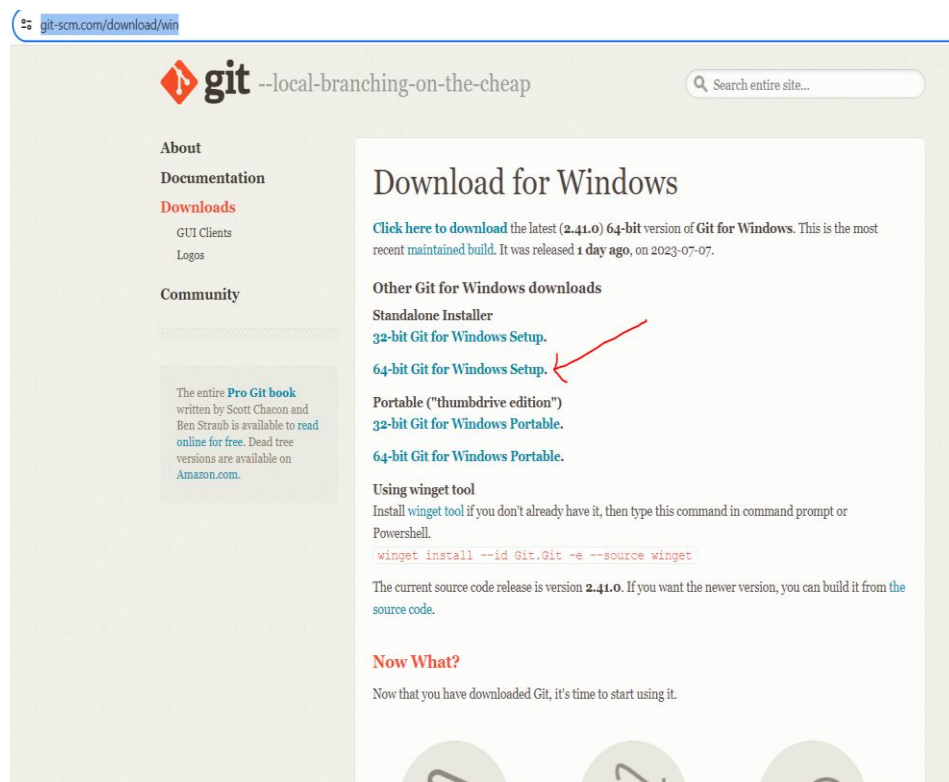
Activity 2.1 : Install Python: Visit the Python website (<https://www.python.org>) and download the latest version of Python. Follow the installation instructions for your operating system.

Download and install python on Windows

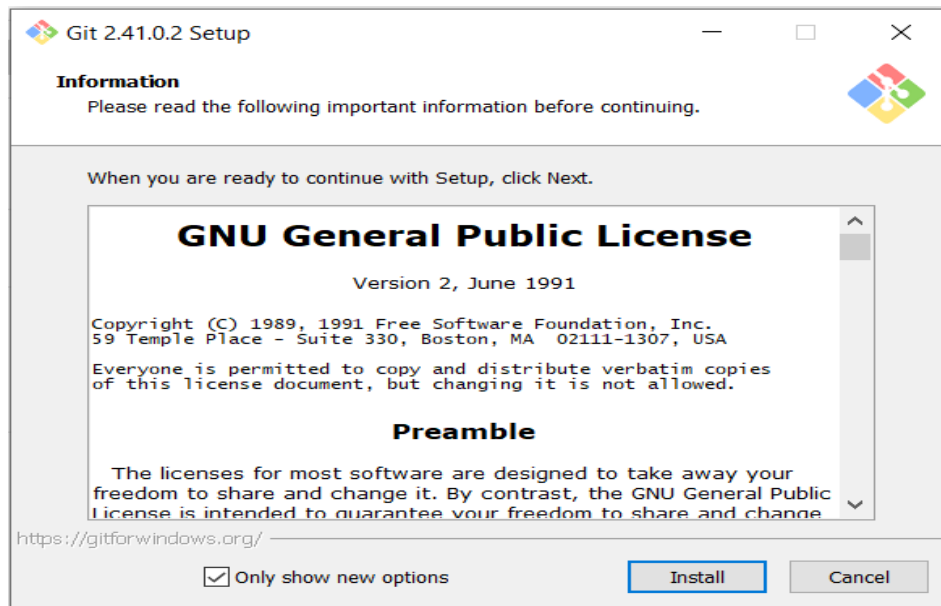
■ <https://www.python.org/downloads/>



Download and install **gitbash** from <https://git-scm.com/download/win>



Activity 2.3 :

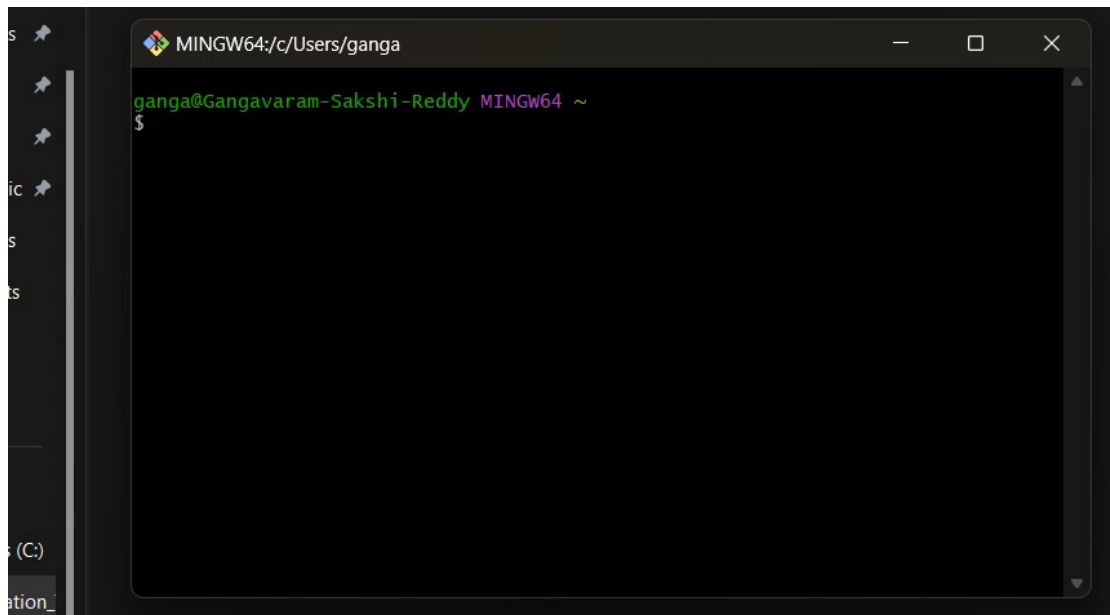


Activity 2.4 :

Go to the desired location where you want to create the folder, such as Desktop.

Folder name like eg...(Automation_Testing_Project)

Open the folder-→ Right click with in the folder and select open gitbash here, you will get pop up like below and clone the repository based on the next steps provided be



Activity 3:- Project Setup

Explained how to set up the project locally, including steps like:

Activity 3.1:- Clone the repository from github.

Activity 3.2:- Clone the actiTIME project repository from [repository URL].

<https://github.com/nandugadiparthi/Acti-Time-Manual-Testing->

Activity 3.3 : Install all dependencies using **requirements.txt**

Requirements.txt containing all software's/libraries required for our projects. We don't need to install separately.

Activity 3.4: RUN the command from the terminal:

```
Copy code
```

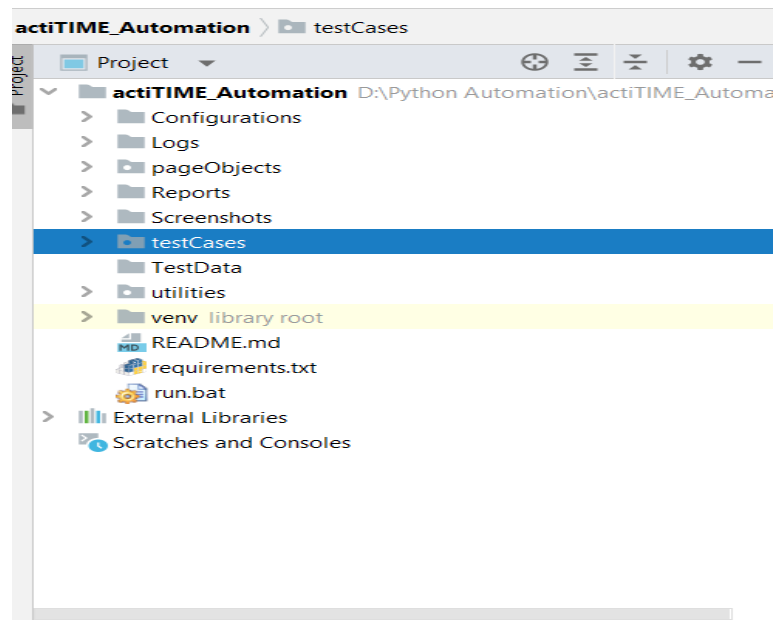
```
pip install -r requirements.txt
```

Activity 3.5 : Configuration settings (if any)

Milestone 2

Project Structure

Activity 1: Explained the structure of the project, including directories and files, and their purposes. For example:



- **testscases/**: Directory containing test scripts
- **pageObjects/**: Directory containing page object models
- **utils/**: Directory containing utility functions
- **reports/**: Directory containing test reports
- **testData/**: Directory containing test data (**ex:excel, csv, text, json files**) related to project
- **Configuration/**: Configuration file for storing settings
- **Screenshots/**: Project testcases screenshots.
- **Logs/**: Directory maintain the project logs.
- **Run.bat/**: Now, you can double-click the **run.bat** file to execute the Python script specified in the command. The script will run in a Command Prompt window.

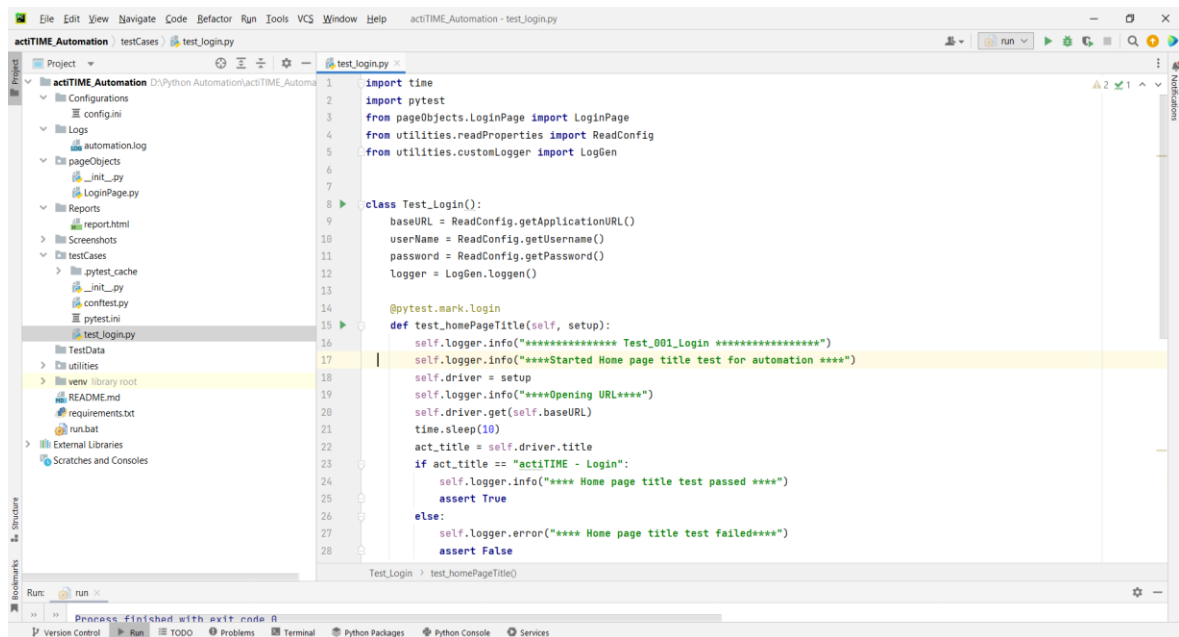
Activity 2 : Writing Test Cases

Explained how to write test cases using Python, Selenium, and pytest.

Provide examples of different types of test cases, such as:

- Test case 1: Home Page functionality

Screenshot of the code:



```
1 import time
2 import pytest
3 from pageObjects.LoginPage import LoginPage
4 from utilities.readProperties import ReadConfig
5 from utilities.customLogger import LogGen
6
7
8 class Test_Login():
9     baseURL = ReadConfig.getApplicationURL()
10    userName = ReadConfig.getUsername()
11    password = ReadConfig.getPassword()
12    logger = LogGen.logger()
13
14    @pytest.mark.login
15    def test_homepageTitle(self, setup):
16        self.logger.info("***** Test_001_Login *****")
17        self.logger.info("****Started Home page title test for automation ****")
18        self.driver = setup
19        self.logger.info("****Opening URL****")
20        self.driver.get(self.baseURL)
21        time.sleep(10)
22        act_title = self.driver.title
23        if act_title == "actiTIME - Login":
24            self.logger.info("**** Home page title test passed ****")
25            assert True
26        else:
27            self.logger.error("**** Home page title test failed****")
28            assert False
```

- Test case 2: Register new user functionality
- **Add new testcases as per Manual testcases prepared.**

Include best practices for writing maintainable and readable test cases.

Milestone 3

Running Tests

Explained how to run the test cases using pytest. Include information about command-line options and flags that can be used to customize test execution.

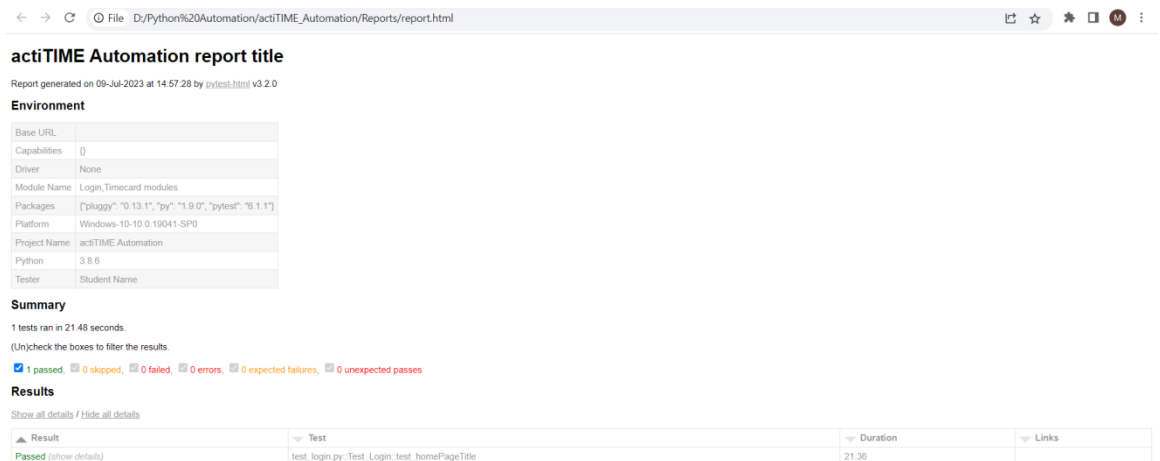
Activity 1:- Run.bat/: Now, you can double-click the **run.bat** file to execute the Python script specified in the command. The script will run in a Command Prompt window.

Milestone 4

Test Reporting

Explained how test reports are generated and where they are stored. Include information about any tools or libraries used for reporting, such as pytest-html.

Activity 1: Reports/: Directory containing test reports



The screenshot shows a web browser window displaying a pytest-html report. The browser's address bar shows the file path: D:/Python%20Automation/actiTIME_Automation/Reports/report.html. The report title is "actiTIME Automation report title". Below the title, it states "Report generated on 09-Jul-2023 at 14:57:28 by pytest-html v3.2.0".

The "Environment" section contains a table with the following details:

Base URL	
Capabilities	{}
Driver	None
Module Name	Login,Timecard modules
Packages	["pluggy", "0.13.1", "py", "1.9.0", "pytest", "6.1.1"]
Platform	Windows-10-10.0.19041-SP0
Project Name	actiTIME_Automation
Python	3.8.6
Tester	Student Name

The "Summary" section indicates "1 tests ran in 21.48 seconds." and provides a legend for the results: 1 passed (blue checkmark), 0 skipped (grey square), 0 failed (red square), 0 errors (red square), 0 expected failures (grey square), and 0 unexpected passes (red square).

The "Results" section has a toggle for "Show all details / Hide all details". Below it is a table with the following data:

Result	Test	Duration	Links
Passed (show details)	test_login.py::Test_Login::test_homePageTitle	21.36	

Advantages :

- Actitime is a web-based project management and time tracking software that helps businesses manage tasks, projects, and track employee work hours. When it comes to manual testing, Actitime offers some advantages that can be beneficial for users.
- User-Friendly Interface: Actitime's manual testing module typically has a user-friendly interface, making it easy for testers to navigate and perform their testing tasks efficiently.
- Easy Test Case Creation: Manual testing in Actitime allows testers to create and manage test cases easily. Testers can define the steps, expected results, and associated test data for each test case.
- Flexible Test Execution: Actitime provides flexibility in test execution. Testers can execute test cases in any order, allowing them to adapt to changing priorities and requirements during the testing process.
- Test Progress Tracking: Manual testing in Actitime offers features for tracking the progress of testing activities. Testers can update the status of each test case, indicating whether it's in progress, passed, failed, or blocked.

Disadvantages:

- While Actitime's manual testing module offers some advantages, it also has certain disadvantages that users should be aware of
- Time-Consuming: Manual testing can be time-consuming, especially for large and complex projects. Testers need to execute each test case step by step, which can be inefficient and slow down the overall testing process.
- Prone to Human Errors: Manual testing heavily relies on human involvement, making it susceptible to human errors and oversight. Testers might miss critical issues, leading to potential defects in the software.
- Limited Test Coverage: Manual testing may have limited test coverage compared to automated testing. Testers may not be able to execute all possible test scenarios

due to time constraints or the repetitive nature of executing the same test cases repeatedly.

- **Resource-Intensive:** Manual testing requires skilled testers who can dedicate significant time and effort to execute test cases. This can lead to higher resource costs, especially for projects that demand extensive testing efforts.
- **Not Suitable for Regression Testing:** Regression testing, which involves testing the application after changes to ensure existing functionality remains intact, can be challenging and time-consuming in manual testing. It's often more efficient to use automated testing for regression testing

Application

- **ActiTIME** is a web-based time tracking and management software, primarily used for project and task management, timesheet management, and workforce analysis. Manual testing plays a crucial role in ensuring the quality and functionality of the **ActiTIME** application. Here are some of the key applications of **ActiTIME** manual testing.
- **Functional Testing:** Ensure that all the functionalities of **ActiTIME** work as expected and meet the specified requirements. This includes testing features like time tracking, task creation, project management, reports generation, user management, etc.
- **User Interface (UI) Testing:** Verify the consistency and usability of the user interface across different browsers and devices. Focus on validating the layout, navigation, and visual elements.
- **Compatibility Testing:** Test **ActiTIME** on various web browsers, operating systems, and devices to ensure its compatibility and proper functionality across different environments.
- **Integration Testing:** Test the integration of **ActiTIME** with other third-party tools or systems, such as project management tools, accounting software, etc.

Troubleshooting

List common issues that may occur during project setup or test execution and provide possible solutions.

- Troubleshooting ActiTIME manual testing involves identifying and resolving issues that arise during the testing process. Here are some common troubleshooting steps you can follow:
- Identify the Problem: First, carefully document and reproduce the issue you encountered during manual testing. Identify the exact steps that led to the problem, including any specific inputs or configurations used.
- Check Test Environment: Ensure that your test environment is set up correctly, including the browser version, operating system, and any required plugins. Verify that ActiTIME is running on the correct server and that all dependencies are properly configured.
- Verify Test Data: Double-check the test data you used during testing. Ensure that it is valid and properly set up to simulate real-world scenarios accurately.

Conclusion

The **actiTIME** automation project provides a framework for automating various tasks within the **actiTIME** web application using Python, Selenium, and Pytest. With this documentation, you should be able to set up the project, execute tests, and extend/customize it according to your needs.