FINDING HOSPITALS

A Hackathon Project Report

By Team – 2

Associate Names	Associate IDs
Nikhil Mishra	2252557
Rubina M Sali	2252434
Kavya Sree Komakula	2252138
Chippa Shyamala	2252008
Duddyala Akshitha	2252251
Pampari Manish	2252141

ABSTRACT

The objective of this project is to use Selenium with Java to automate the process of finding hospitals that are open 24/7, and a rating of more than 3.5 in Bangalore city. The project involves using various automation techniques such as handling alerts, different browser windows, search options, and filling forms on the website.

The project also involves picking all the top cities from the diagnostics page and storing them in a list, displaying the same. Additionally, the project includes filling invalid details on the Corporate Wellness page, scheduling, and capturing the warning message from the alert. This project will help to improve the efficiency of finding hospitals with specific features, saving time and effort for the end-users.

In conclusion, this project provides a solution to the problem of finding hospitals that match specific criteria on a website. The use of Selenium with Java allows for the automation of the tasks, making the process more efficient and reliable. The automation scope covers various aspects, including handling alerts, multiple browser windows, search options, extracting multiple option items, filling forms, and capturing warning messages. Overall, this project showcases the power of automation in web testing and can be extended to other websites and criteria.

CONTENT

Chapter 1: Introduction

1.1 Overview and Motivation

The project aims to automate the process of finding hospitals with specific features such as 24/7 availability, and a rating of more than 3.5 in Bangalore city using Selenium with Java. The automation techniques used in the project include handling alerts, different browser windows, search options, and filling forms on the website. Additionally, the project involves extracting multiple options and storing them in collections, navigating back to the home page, and capturing warning messages.

In today's world, people have a busy lifestyle, and finding the right hospital with specific features can be time-consuming and tiring. This project aims to automate the process of finding hospitals that meet the user's specific requirements, thereby saving time and effort. Moreover, the project provides an opportunity to learn and implement various automation techniques such as handling alerts, different browser windows, search options, and filling forms on the website. The project is beneficial to the end-users, and it also helps in improving the efficiency of the healthcare industry.

1.2 Problem Statement

The project requires identifying hospitals in Bangalore that are open 24/7, and have a rating of at least 3.5. Once the hospitals meeting these criteria are identified, their names should be displayed. Additionally, the project requires picking all the top cities' names from the Diagnostics page and displaying them. Finally, the project involves filling in invalid details for Corporate Wellness, scheduling, and capturing the warning message from the alert.

1.3 Detailed Description

The "Finding Hospitals" Selenium project aims to automate the process of finding hospitals based on certain criteria, such as being open 24/7, and having a rating of more than 3.5. The project will be implemented using Selenium with Java.

The project will be divided into three main parts:

Finding Hospitals in Bangalore: The first part of the project involves identifying hospitals in Bangalore that meet the criteria of being open 24/7, and having a rating of more than 3.5. The Selenium script will navigate to the website, search for hospitals in Bangalore, and filter the results based on the specified criteria. The script will then extract the names of the hospitals that meet the criteria and display them on the screen.

Collecting Top Cities: The second part of the project involves collecting the names of top cities in the Diagnostics page of the website. The script will navigate to the Diagnostics page, identify the section that contains the names of the cities, and extract them. The script will then store the names of the cities in a list and display them on the screen.

Corporate Wellness Form Validation: The third part of the project involves filling out a form on the Corporate Wellness page of the website, entering invalid details, scheduling an appointment, and capturing the warning message from the alert. The script will navigate to the Corporate Wellness page, fill out the form with invalid details, click the Schedule button, and capture the warning message from the alert that appears on the screen.

The key automation scopes of the project include handling alerts, working with different browser windows, searching options on the website, navigating back to the home page, extracting multiple options items and storing them in collections, filling out forms in different objects on the web page, and capturing warning messages.

The project may face some challenges, such as handling dynamic web pages, working with different browsers, bypassing captchas, ensuring data integrity, handling user authentication, complying with legal requirements and ethical standards, and not overloading the website with requests or causing harm to the website or its users. However, with careful planning and implementation, these challenges can be overcome, and the project can be completed successfully.

CHAPTER 2: Proposed Work

The proposed work for the "Finding Hospitals" Selenium project involves automating the process of identifying hospitals in Bangalore that are open 24/7, and have a rating of more than 3.5. Additionally, the project includes collecting the names of top cities from the Diagnostics page and filling out the Corporate Wellness form with invalid details to capture the warning message from the alert.

The following steps outline the proposed work for the project:

Define project requirements: Clearly define the project requirements and objectives to ensure that the automation process meets the stakeholders' needs.

Set up development environment: Set up the development environment by installing the necessary software and tools, such as Selenium, Java, and a suitable integrated development environment (IDE), such as Eclipse.

Develop script for finding hospitals in Bangalore: Write a script that navigates to the website, searches for hospitals in Bangalore, filters the results based on the criteria, extracts the names of hospitals that meet the criteria, and displays the names on the screen.

Develop script for collecting top cities: Create a script that navigates to the Diagnostics page, identifies the section containing the names of top cities, extracts the names of cities, stores the names in a list, and displays the names on the screen.

Develop script for filling out Corporate Wellness form: Write a script that navigates to the Corporate Wellness page, fills out the form with invalid details, clicks the Schedule button, and captures the warning message from the alert.

Test and debug scripts: Test the scripts to ensure that they work as intended and debug any errors or issues that arise.

Optimize scripts for better performance: Optimize the scripts to improve their performance, such as by reducing the time required to extract data or by increasing the accuracy of data extraction.

Document the project: Document the project, including the project requirements, development environment setup, scripts, test results, issues and challenges, and any other relevant information.

Deliver project to stakeholders: Deliver the completed project to stakeholders, including any necessary documentation or training, to ensure that they can use the automation process effectively.

2.1 Work Flow of the Project

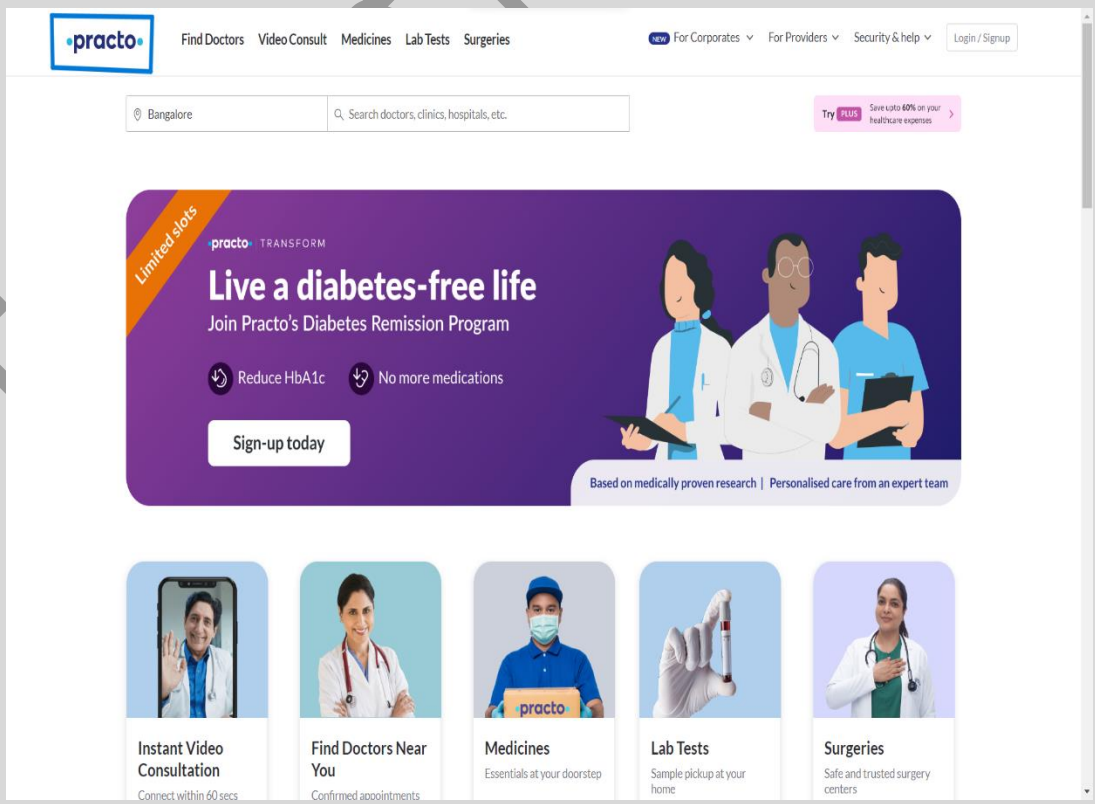
1. Define project requirements
2. Set up development environment
3. Develop script for finding hospitals in Bangalore:
    a. Navigate to the website
    b. Search for hospitals in Bangalore
    c. Filter results based on criteria
    d. Extract hospital names that meet the criteria
    e. Display hospital names on screen
4. Develop script for collecting top cities:
    a. Navigate to Diagnostics page
    b. Identify section containing names of top cities
    c. Extract names of cities
    d. Store names in a list
    e. Display names on screen
5. Develop script for filling out Corporate Wellness form:
    a. Navigate to Corporate Wellness page
    b. Fill out form with invalid details
    c. Click Schedule button

d. Capture warning message from alert

6. Test and debug scripts
7. Optimize scripts for better performance
8. Document the project:
    a. Project requirements
    b. Development environment setup
    c. Scripts
    d. Test results
    e. Issues and challenges
9. Deliver project to stakeholders.

2.2 Suggested Website

https://www.practo.com/



CHAPTER 3: Tools and Techniques

3.1 Tools Used

The main tool used in this project is **Selenium WebDriver with Java** programming language. Selenium WebDriver is a popular automation testing tool used for web applications. It allows testing of web applications across various browsers and platforms. Java is a popular programming language used for developing automation scripts for Selenium WebDriver. It is widely used because of its platform independence, object-oriented nature, and rich set of libraries.

In addition to Selenium and Java, the project may also require the use of other tools such as:

TestNG: It is a testing framework for Java that helps in the organization of test cases and provides various features such as test case grouping, parallel execution, and data parameterization.

Maven: It is a build automation tool that helps in managing dependencies and building the project. It simplifies the process of building and deploying the application.

Git: It is a version control system that allows multiple developers to work on the same project simultaneously. It helps in tracking changes made to the code and resolving conflicts.

Integrated Development Environment (IDE): An IDE is a software application that provides a comprehensive environment for writing, testing, and debugging code. Examples of popular IDEs for Java include Eclipse, IntelliJ IDEA, and NetBeans.

Browser Drivers: In order to execute tests on different browsers, browser-specific drivers such as ChromeDriver, FirefoxDriver, etc. need to be installed and configured.

These tools help in automating the testing process, managing dependencies, and ensuring version control. They help in achieving faster and more efficient testing, reducing manual errors, and increasing the overall quality of the application.

3.2 Technologies Used

The technologies used in this project include Selenium with Java, Apache POI, POM, Extent Reports, and GitHub.

Selenium is an open-source and portable software testing framework for web applications that supports multiple languages like Java, C#, Ruby, and Python. Java was chosen for this project due to its popularity and the availability of a good community of developers for documentation and issue resolution. Programs written in Java are also faster than other popular languages like Python, and Java is more widely used in commercial applications, making it easier to integrate Selenium tests.

Apache POI is a popular API that allows programmers to create, modify, and display MS Office files using Java programs. It is an open-source library developed and distributed by Apache Software Foundation to design or modify Microsoft Office files using Java program.

POM, or Project Object Model, is the fundamental unit of work in Maven. It is an XML file that contains information about the project and configuration details used by Maven to build the project. It contains default values for most projects, including the build directory, which is target, the source directory, which is src/main/java, the test source directory, which is src/test/java, and so on.

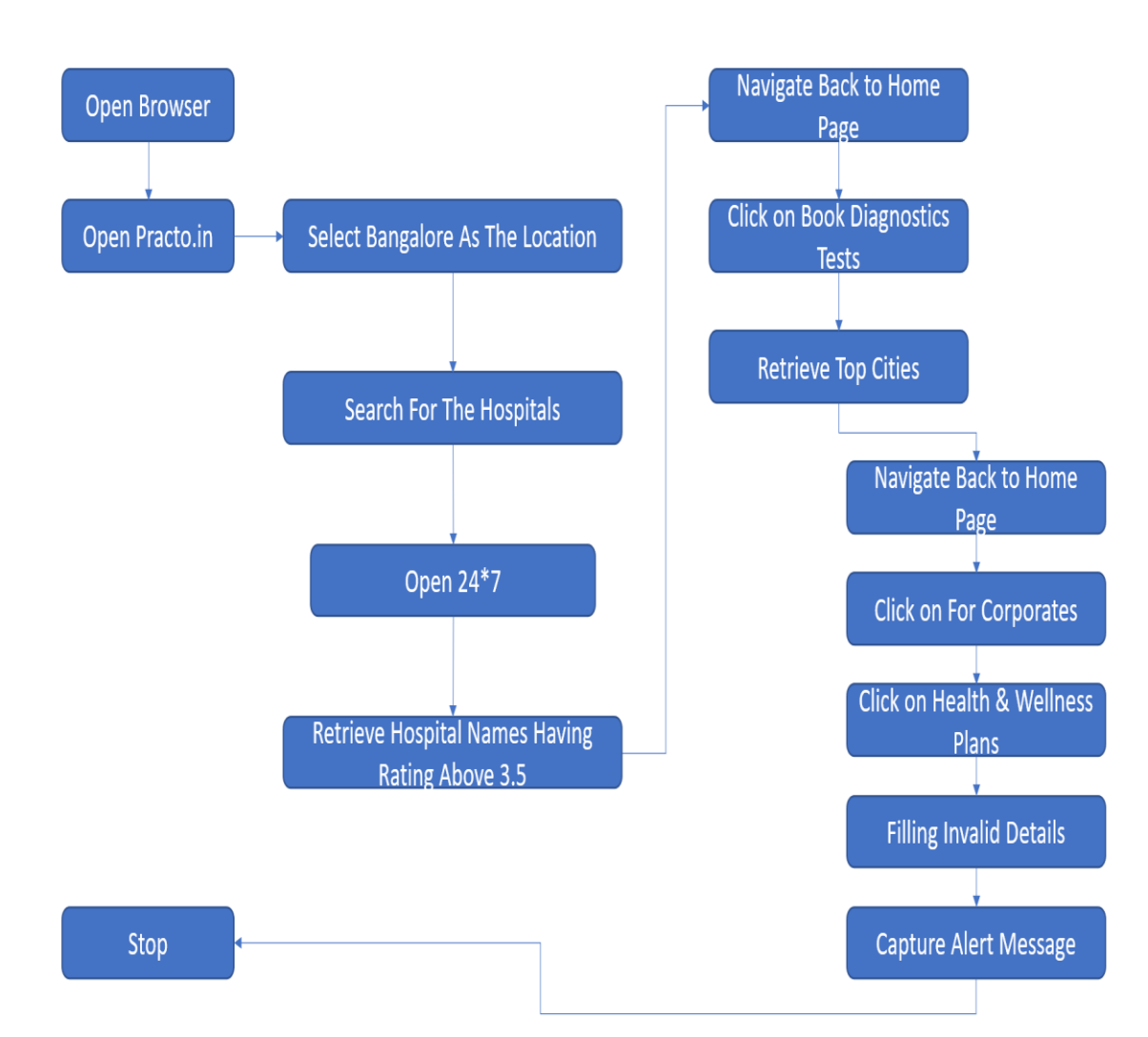
Extent Reports is an open-source reporting library useful for test automation, which can be easily integrated with major testing frameworks like JUnit, NUnit, TestNG, etc. These reports are HTML documents that depict results as pie charts and allow the generation of custom logs, snapshots, and other customized details.

Finally, GitHub is used for source code management and backup. GitHub is a code hosting platform for version control and collaboration, allowing developers to work together on projects from anywhere.

CHAPTER 4: Website Work Flow & Snippets

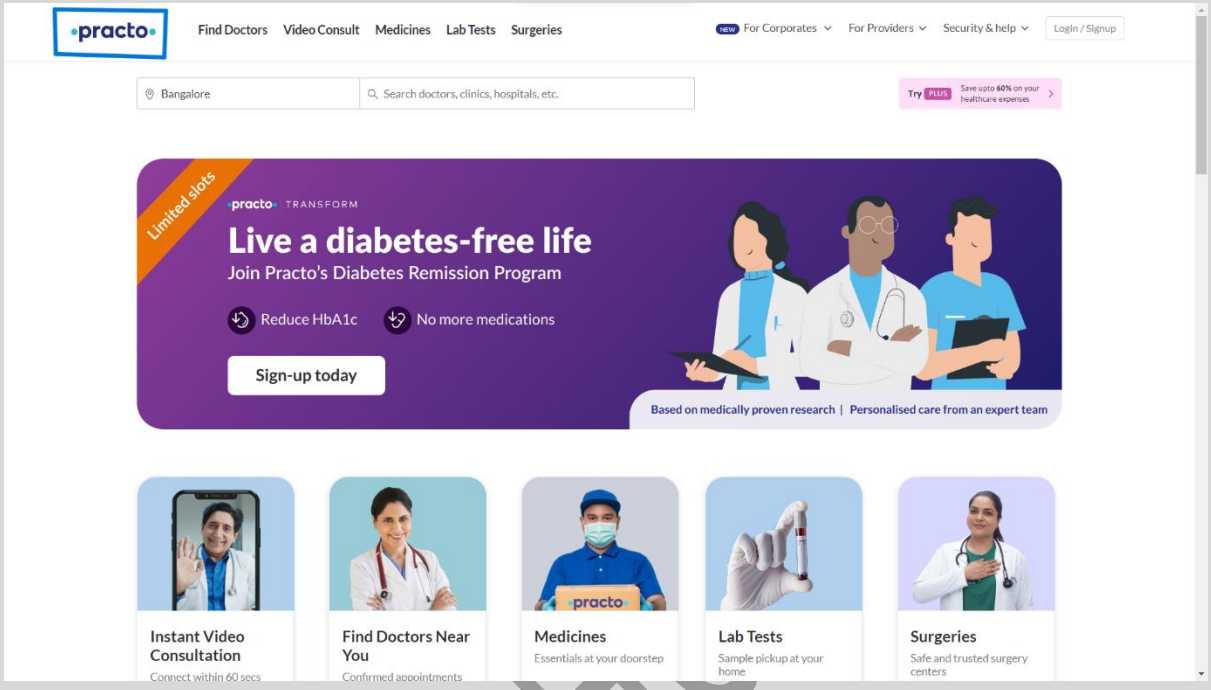
4.1 Website Work Flow Diagram

https://www.practo.com/

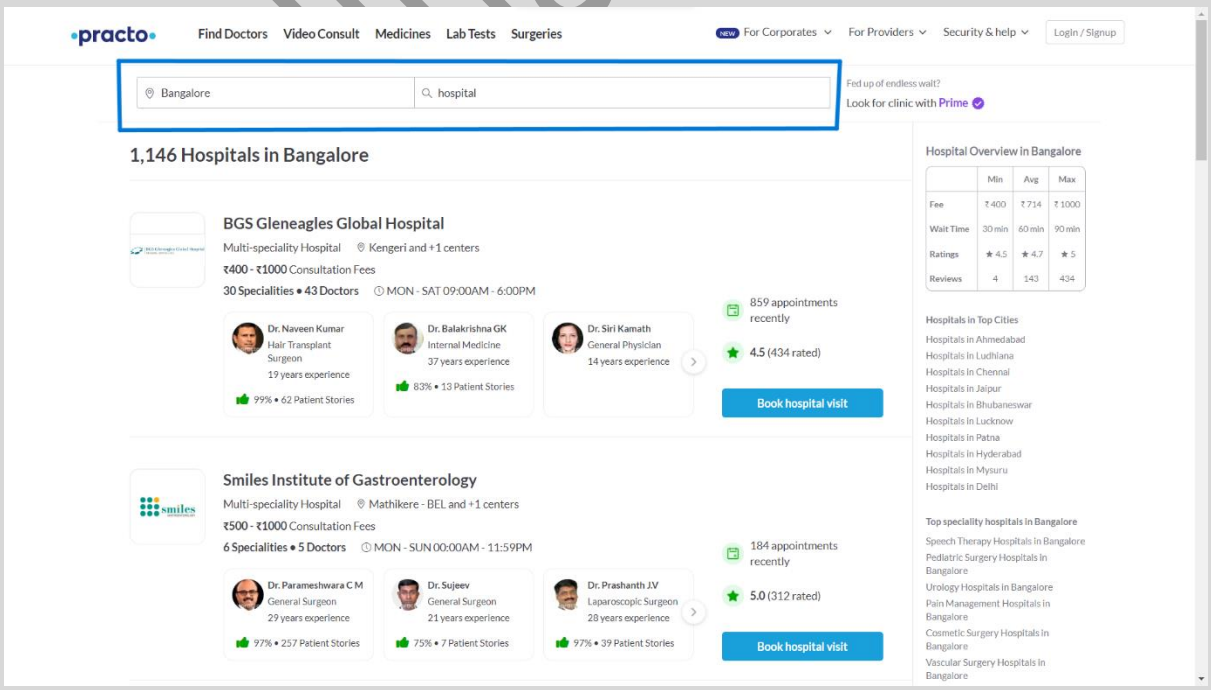


4.2 Website Snippets

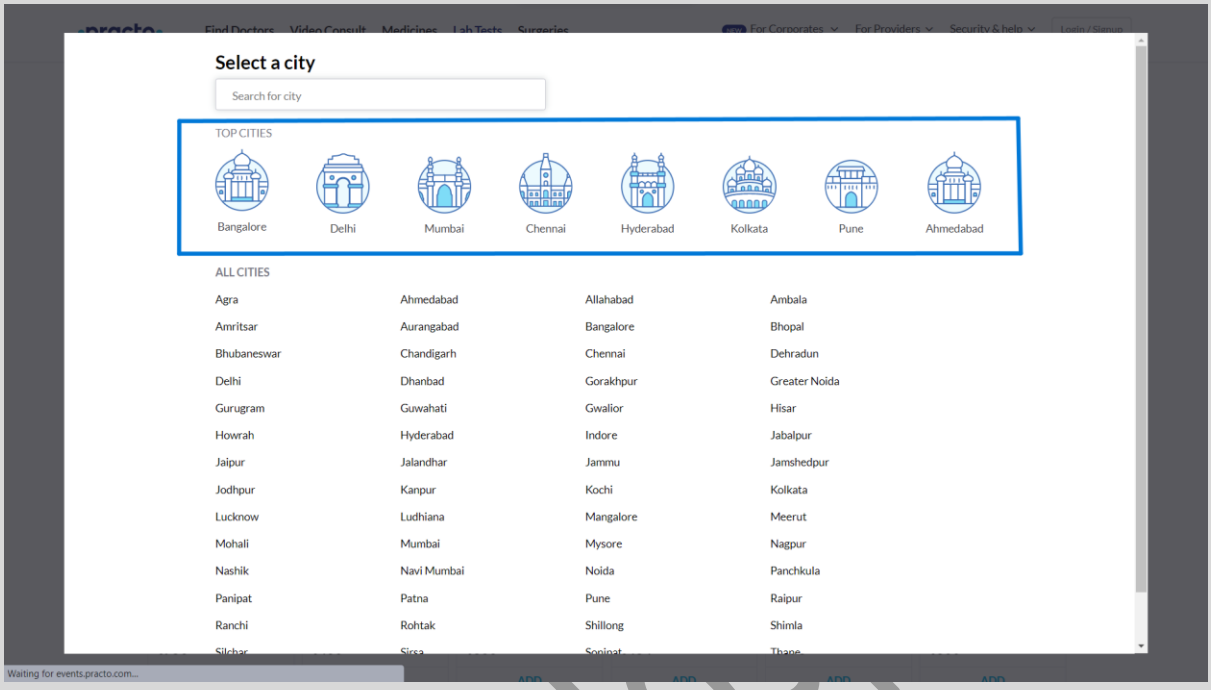
Home Page:



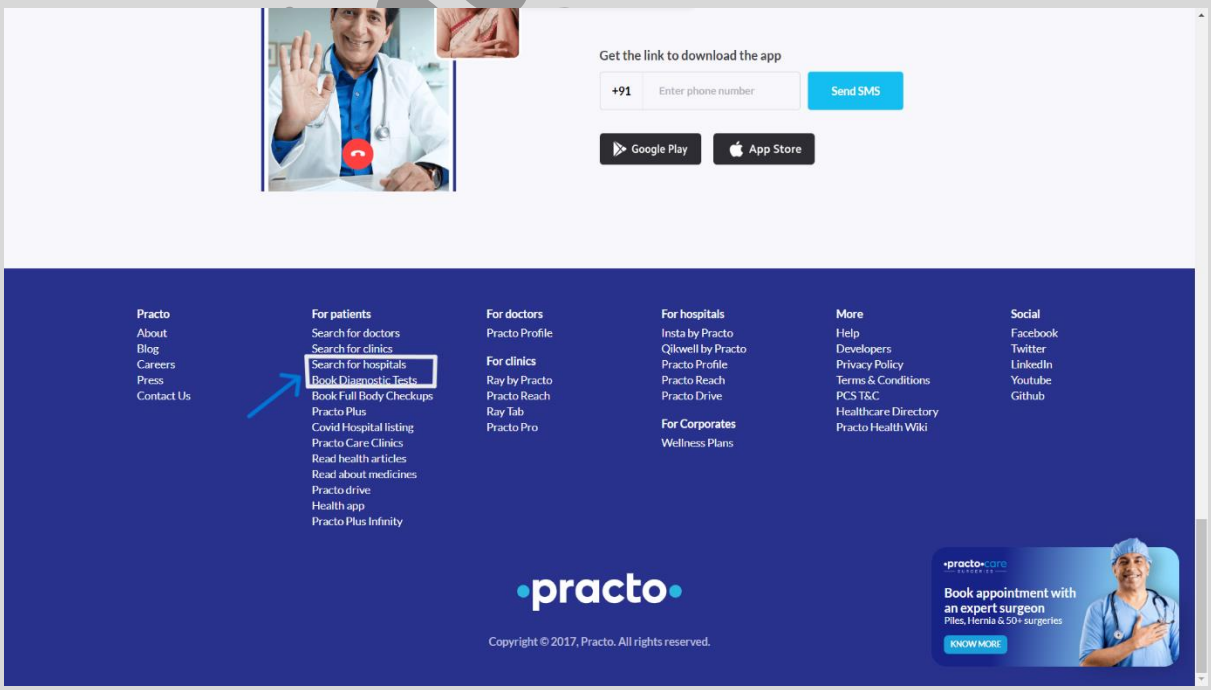
Hospitals in Bangalore:



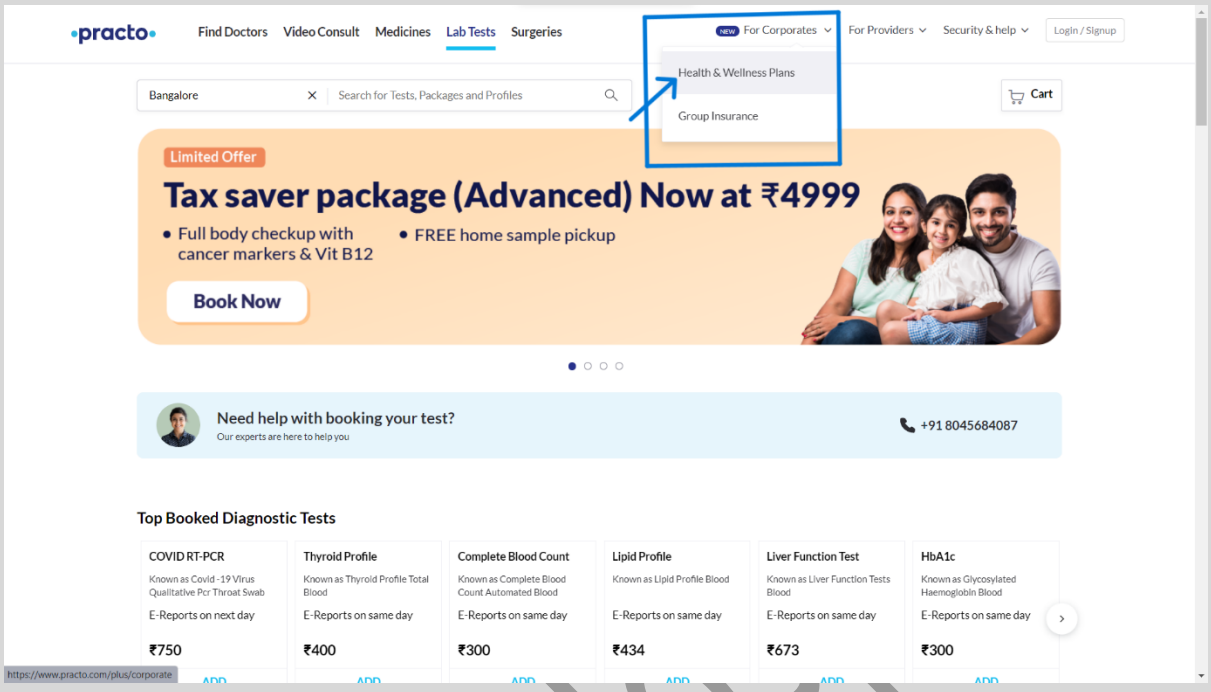
Top Cities:



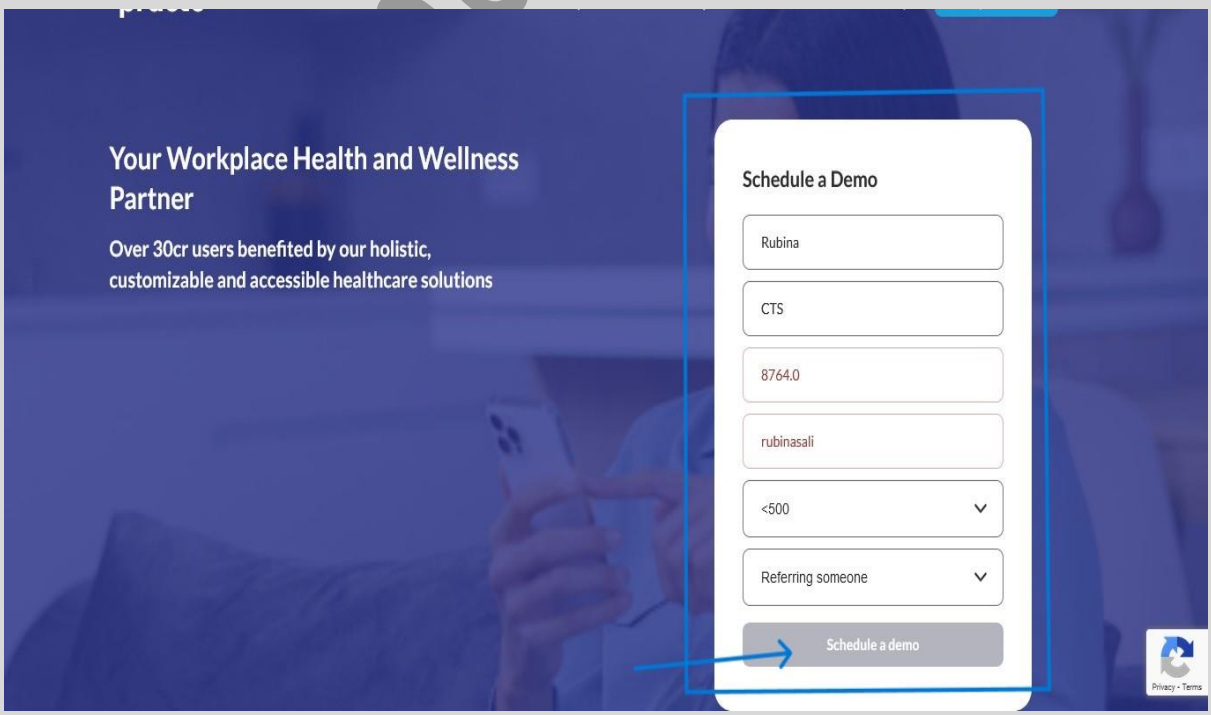
Book Diagnostic Tests:



For Corporates:

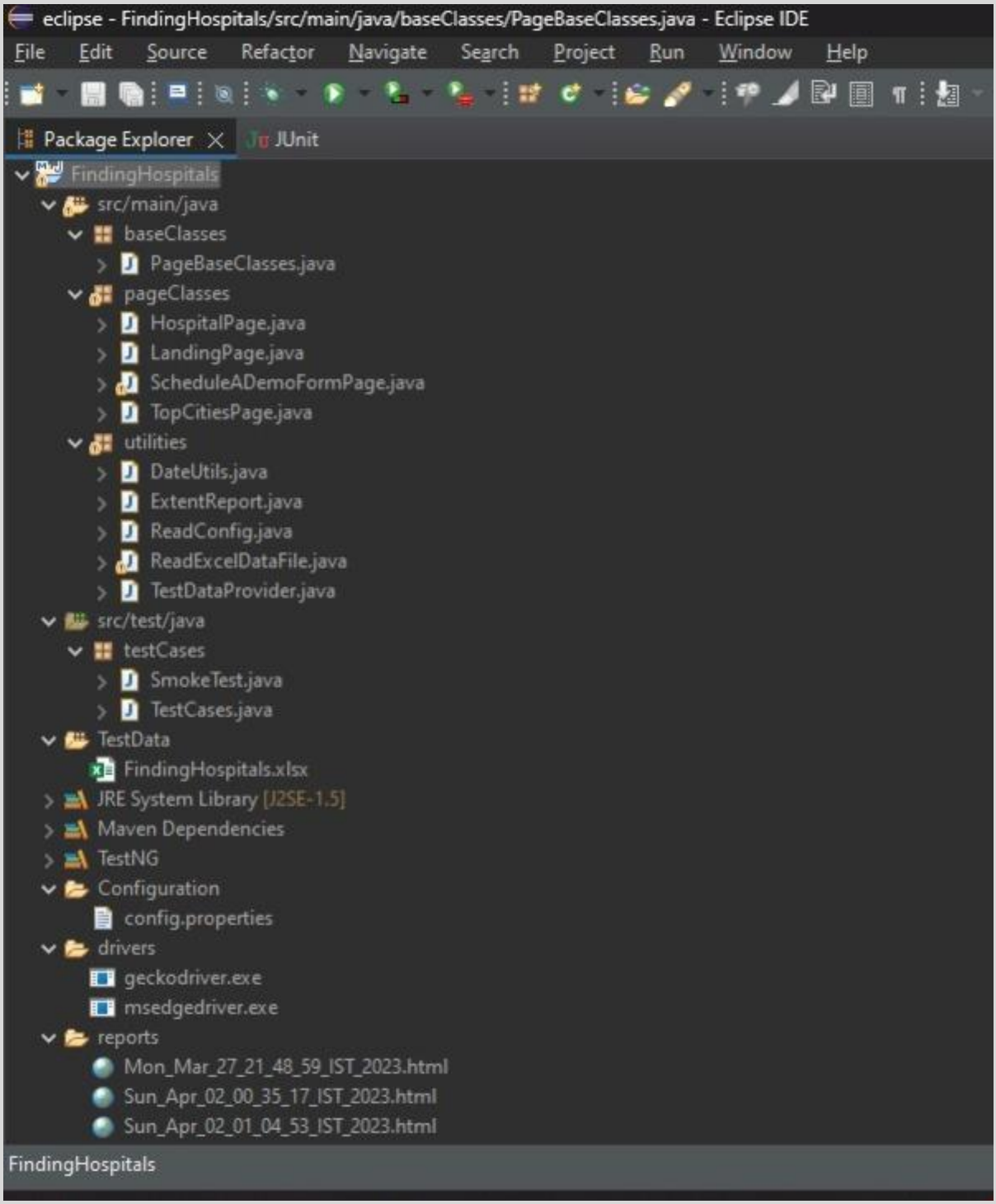


Invalid Demo Form:



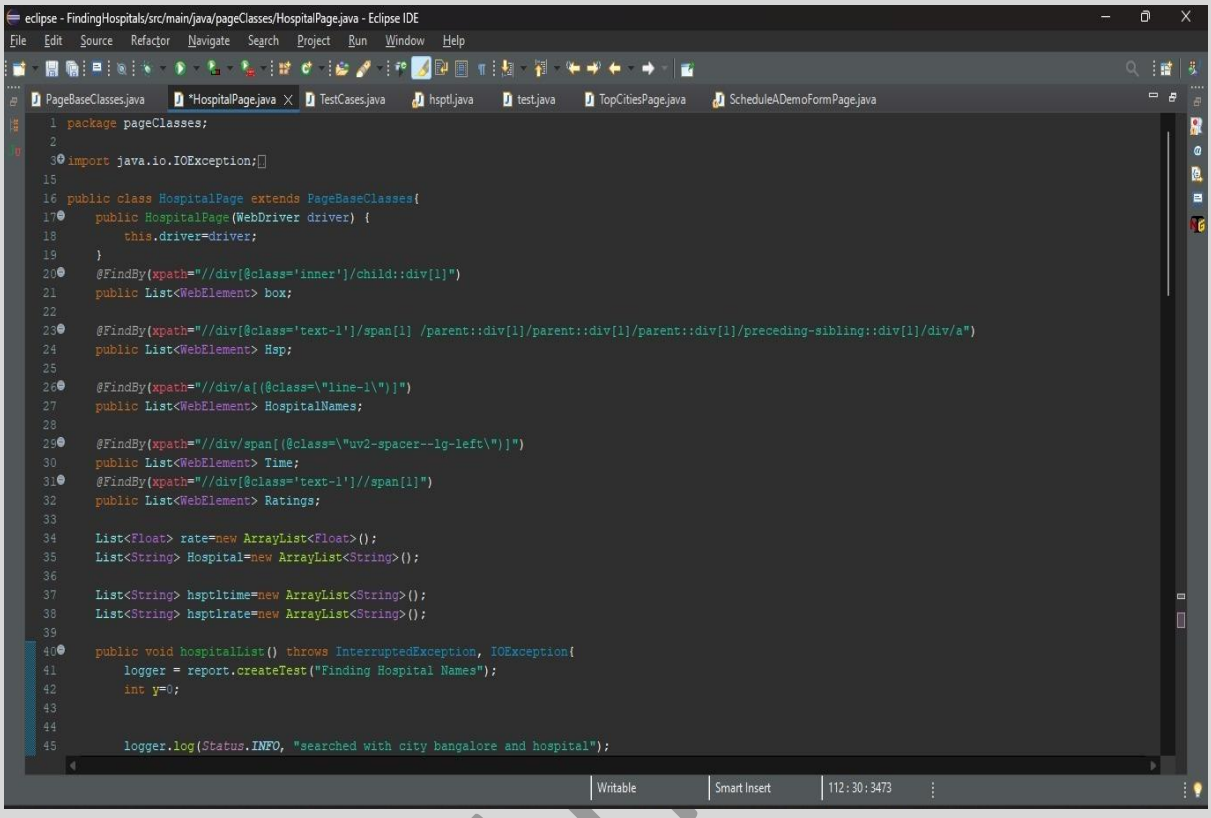
CHAPTER 5: Source Code, Output, & Test Report Snippets

5.1 Source File Structure



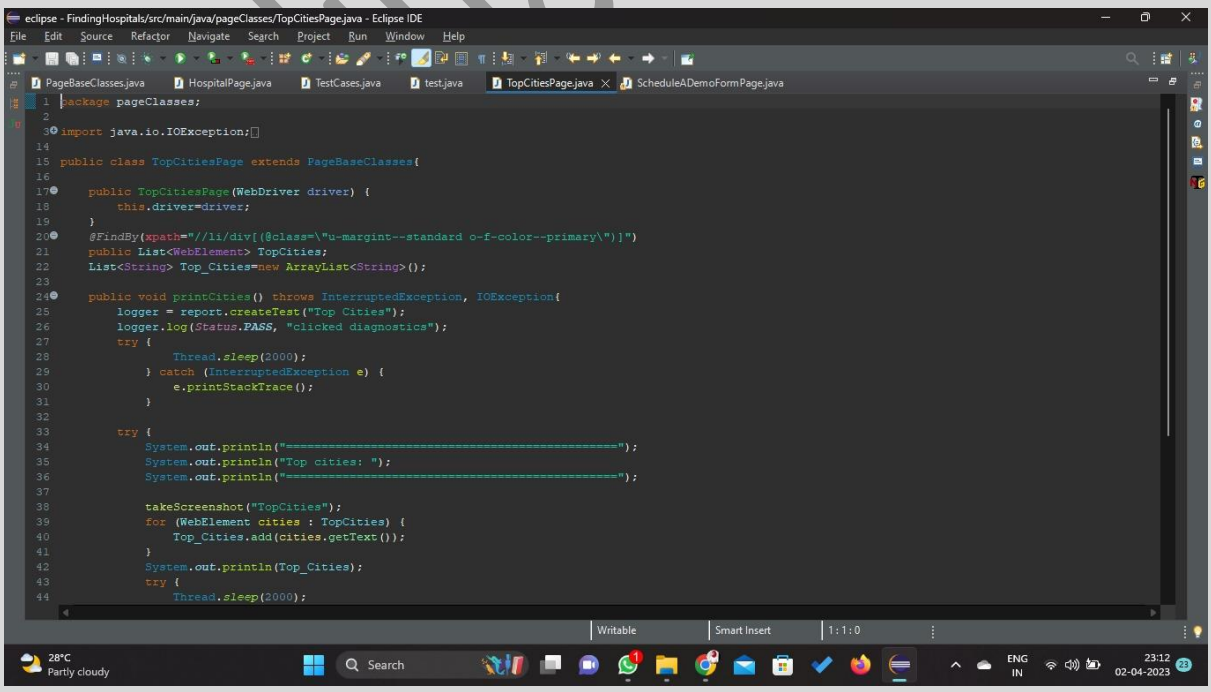
5.2 Source Code Snippets

HospitalPage.java:



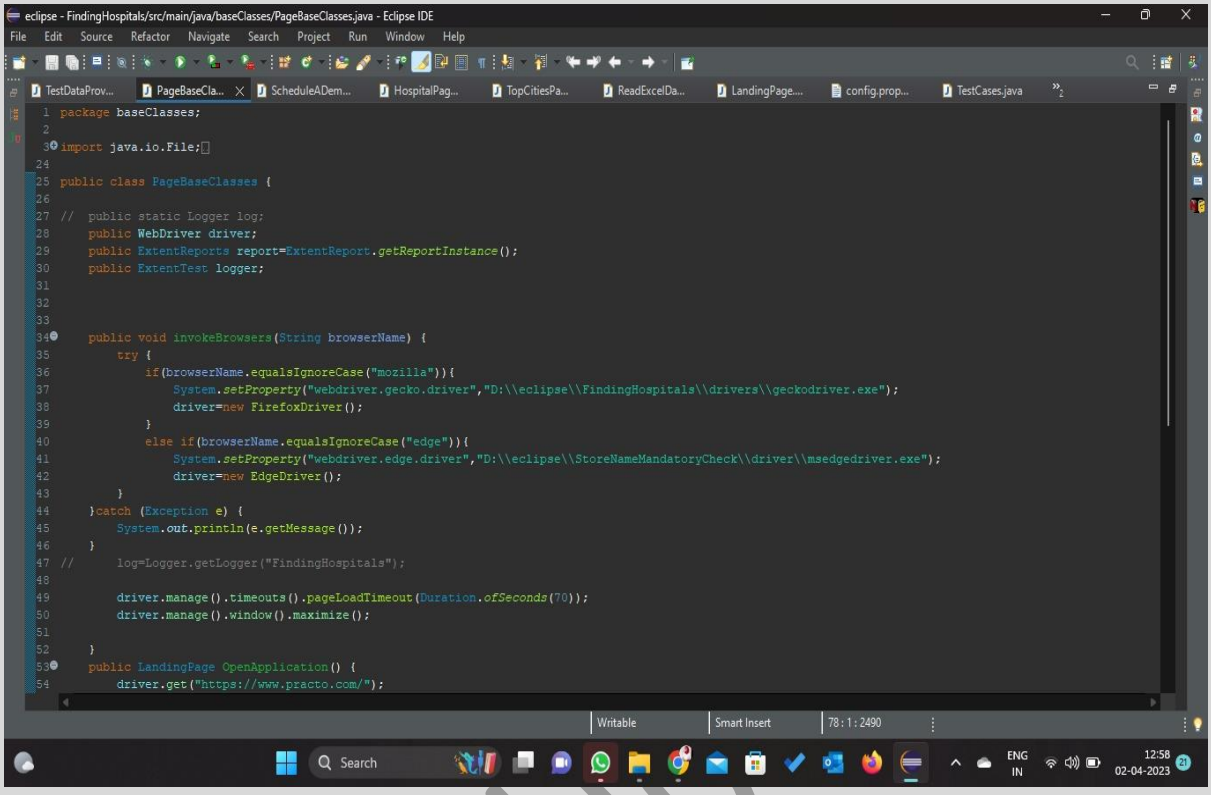
```
package pageClasses;

import java.io.IOException;

public class HospitalPage extends PageBaseClasses{
    public HospitalPage(WebDriver driver) {
        this.driver=driver;
    }
    @FindBy(xpath="//div[@class='inner']/child::div[1]")
    public List<WebElement> box;

    @FindBy(xpath="//div[@class='text-1']/span[1] /parent::div[1]/parent::div[1]/parent::div[1]/preceding-sibling::div[1]/div/a")
    public List<WebElement> Hsp;

    @FindBy(xpath="//div/a[(@class=\"line-1\")]")
    public List<WebElement> HospitalNames;

    @FindBy(xpath="//div/span[(@class=\"uv2-spacer--lg-left\")]")
    public List<WebElement> Time;
    @FindBy(xpath="//div[@class='text-1']//span[1]")
    public List<WebElement> Ratings;

    List<Float> rate=new ArrayList<Float>();
    List<String> Hospital=new ArrayList<String>();

    List<String> hsptltime=new ArrayList<String>();
    List<String> hsptlrate=new ArrayList<String>();

    public void hospitalList() throws InterruptedException, IOException{
        logger = report.createTest("Finding Hospital Names");
        int y=0;


        logger.log(Status.INFO, "searched with city bangalore and hospital");
```

TopCitiesPage.java:

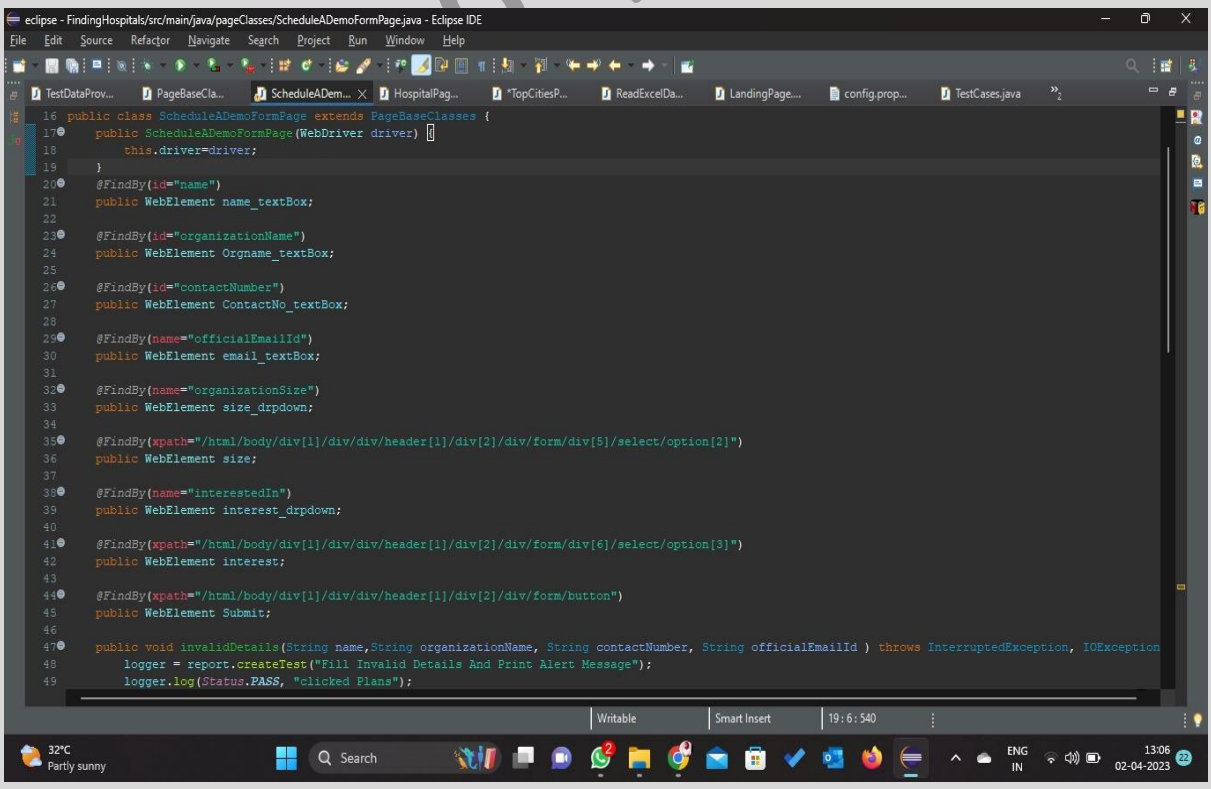


```
package pageClasses;

import java.io.IOException;

public class TopCitiesPage extends PageBaseClasses{

    public TopCitiesPage(WebDriver driver) {
        this.driver=driver;
    }
    @FindBy(xpath="//li/div[(@class=\"u-margint--standard o-f-color--primary\")]")
    public List<WebElement> TopCities;
    List<String> Top_Cities=new ArrayList<String>();

    public void printCities() throws InterruptedException, IOException{
        logger = report.createTest("Top Cities");
        logger.log(Status.PASS, "clicked diagnostics");
        try {
                Thread.sleep(2000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

        try {
            System.out.println("==========================================");
            System.out.println("Top cities: ");
            System.out.println("==========================================");

            takeScreenshot("TopCities");
            for (WebElement cities : TopCities) {
                Top_Cities.add(cities.getText());
            }
            System.out.println(Top_Cities);
            try {
                Thread.sleep(2000);
```

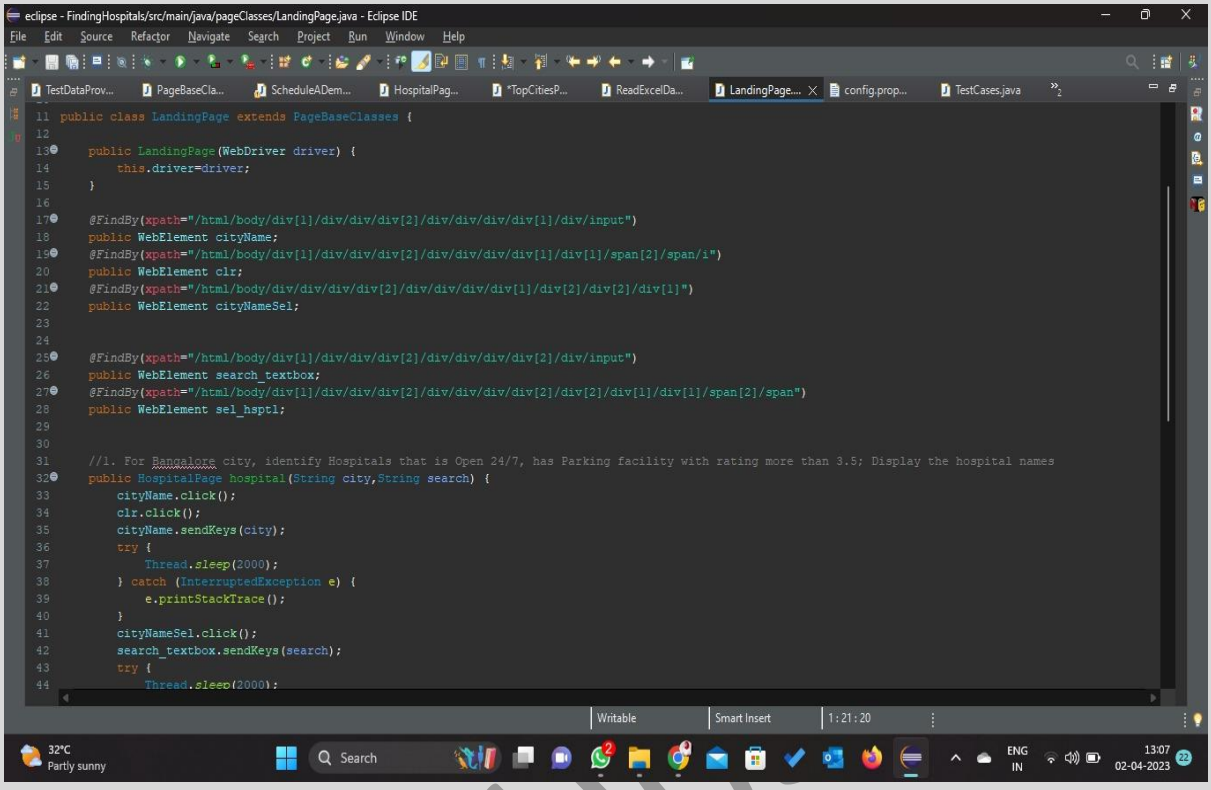
PageBaseClass.java:



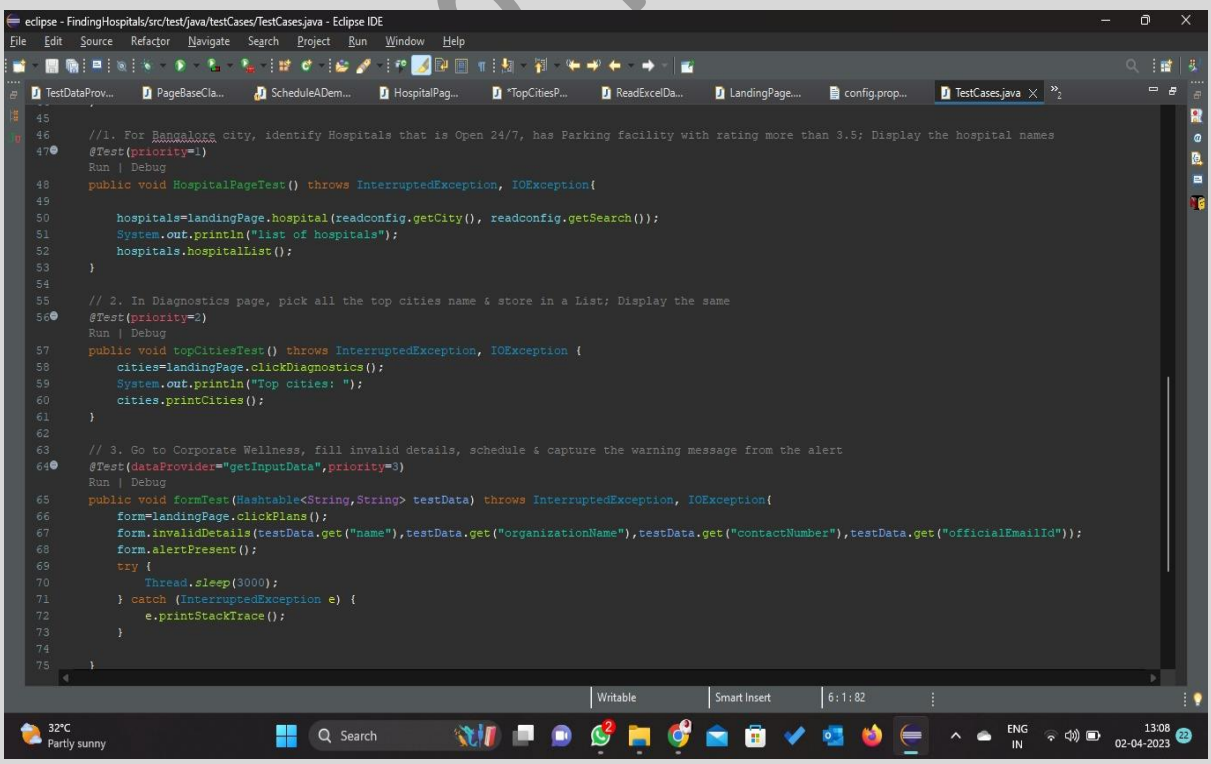
ScheduleADemoPage.java:



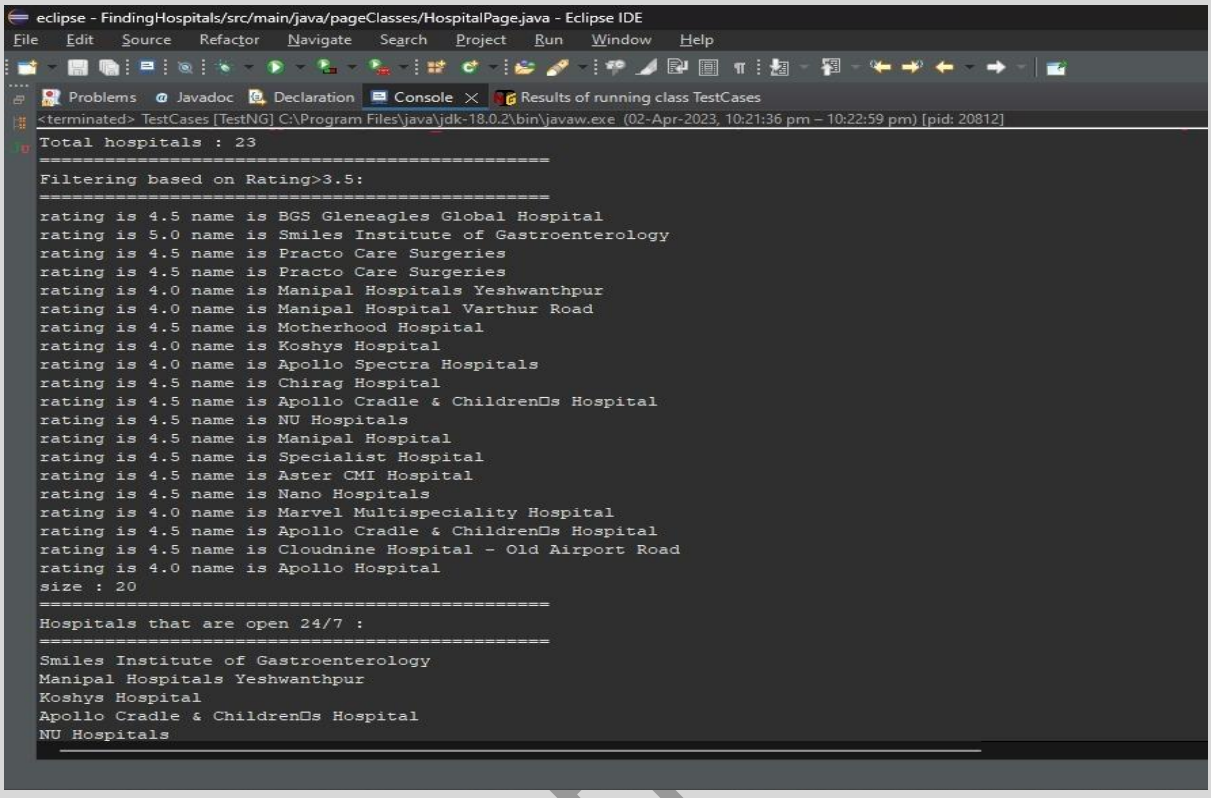
LandingPage.java:

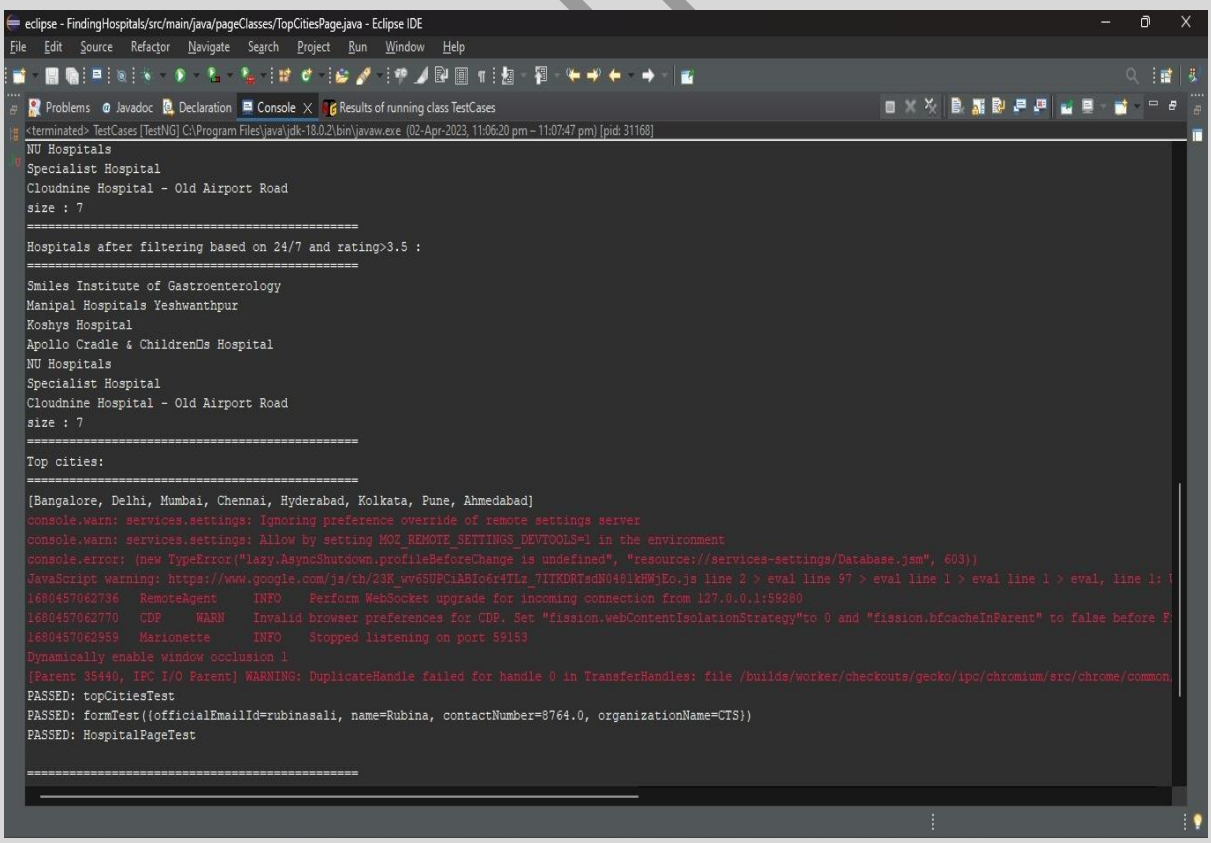


TestCases.java:



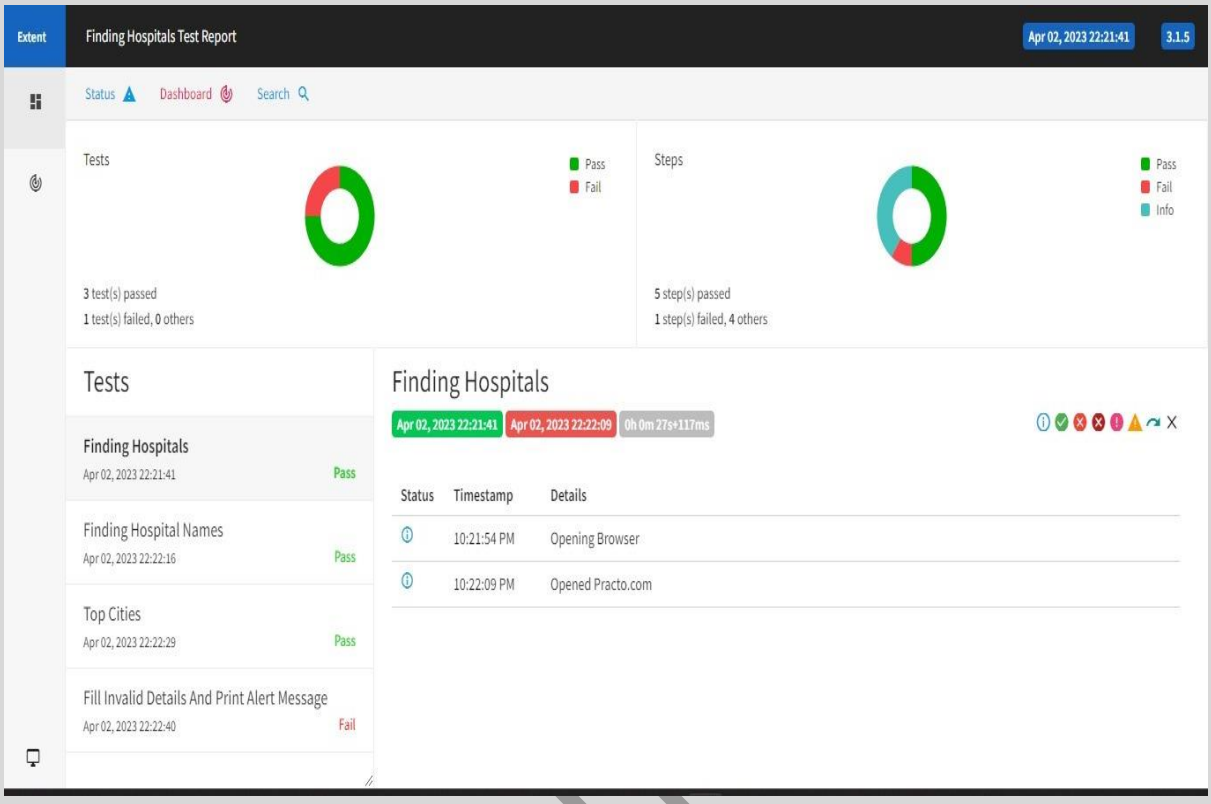
5.3 Output Snippets



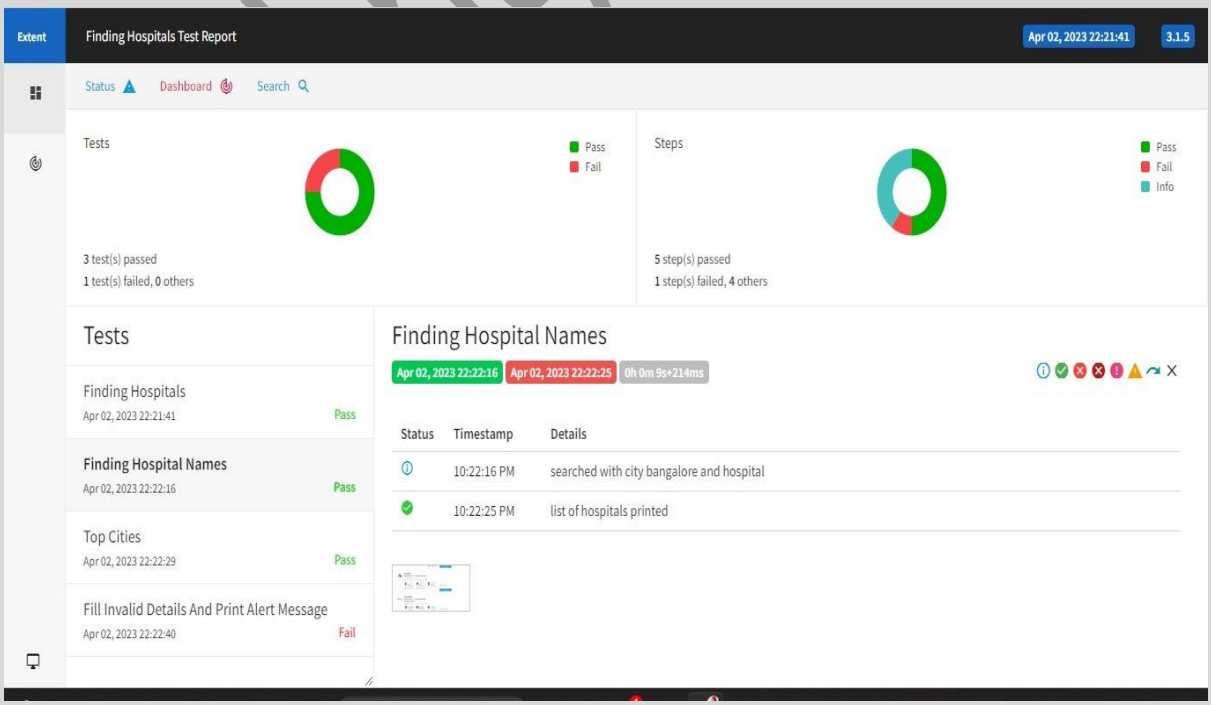


5.4 Test Report Snippets

Finding Hospitals Test Report:



Finding Hospital Names Test Report:



CHAPTER 6: Conclusion

In conclusion, the project focused on automating the testing process of a hospital finding website, with a specific focus on finding hospitals that are open 24/7, and have a rating of more than 3.5. The project was executed using Selenium with Java, Maven, TestNG, Apache POI, POM, and Extent Reports.

Selenium with Java was used as the primary tool for automating the web testing process. Java was chosen as the programming language due to its popularity, active community, and the fact that it is widely used in commercial applications. Maven was used to simplify the build process, provide a uniform build system, and encourage better development practices.

TestNG was used as the testing framework, which offers advanced and useful features that make it a more robust framework compared to its peers. Apache POI was used to create, modify, and display MS Office files using Java programs, while POM provided a fundamental unit of work in Maven that contains information about the project and configuration details used by Maven to build the project.

Finally, Extent Reports was used as an open-source reporting library that allowed for the easy integration of major testing frameworks like JUnit, NUnit, and TestNG, among others. GitHub was used as the platform for source code management and backup.

Overall, the project demonstrated the effectiveness of using automated testing tools to simplify the testing process, reduce manual errors, and increase testing efficiency. The project also highlights the importance of choosing the right combination of tools and technologies to ensure the success of an automation project.

THANK YOU!!