A MINI PROJECT REPORT ON

"Regular Expression Testcases for Testing Purpose"

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE IN THE PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE

OF

BACHELOR OF ENGINEERING

(COMPUTER ENGINEERING)

UNDER THE GUIDENCE OF

Mr. Dattatray Modani

DEPARTMENT OF COMPUTER ENGINEERING
P.E.S MODERN COLLEGE OF ENGINEERING
PUNE 411005.

SAVITRIBAI PHULE PUNE UNIVERSITY
[2025 - 26]

Progressive Education Society's

Modern College of Engineering,

Shivaji Nagar, Pune- 411005.

Certificate

This is to certify that the following students of Computer Engineering of PES's. Modern College of Engineering have successfully completed their mini project in **Software Testing And Quality Assurance** and designed the project entitled **"Regular Expression Testcases for Testing Purpose"** under the guidance of the course instructor.

The Group Members are

Sr No.	Group Members	Roll Number
1)	Hemali Bharambe	41204
2)	Vishwajeet Londhe	41244
3)	Sahil Mate	41246
4)	Vaibhavi Mohite	41248

Internal Supervisor
Mr. Dattatray Modani

Head of Department
(Computer Engineering)
Prof. Dr. S. A. Itkar

Abstract

The Dynamic COVID-19 Information Website is an interactive web application designed to provide real-time updates, awareness, and essential information about the COVID-19 pandemic. The website is developed using HTML, CSS, and JavaScript for the front-end interface, ensuring a user-friendly and responsive experience across various devices. The back-end is implemented using PHP, which handles server-side logic, and a MySQL database, which stores critical information such as user accounts, registration details, and user comments. This system allows users to register, log in, and share feedback or discussions related to COVID-19, creating an engaging and informative platform.

The main objective of this project is to present accurate and dynamic COVID-19 statistics and safety guidelines to users while maintaining a secure and efficient data management system. The system includes modules such as user authentication, information display, and comment management, each tested thoroughly to ensure reliability and consistency. Regular expressions are implemented in form validation to check the correctness of user inputs like email format, password strength, and contact numbers, preventing invalid or malicious data entry.

From a software testing and quality assurance perspective, the project focuses on applying black-box testing techniques, functional testing, and validation through regular expressions. The test cases designed for each module ensure that all functionalities, including registration, login, and comment posting, work as intended. Testing also covers input validation, database connectivity, and front-end responsiveness, which are crucial for maintaining both performance and data security.

In conclusion, this mini project demonstrates the integration of web technologies and testing principles to develop a reliable and user-centric information platform. By applying systematic testing methodologies, the project ensures high-quality software that meets usability, performance, and data integrity standards. The combination of dynamic content delivery and rigorous quality assurance practices reflects a practical implementation of software engineering and testing concepts in a real-world scenario.

Contents

Chapter 1

Introduction

1.1 Introduction

The emergence of COVID-19 created a global demand for trusted and accessible sources of information. Many individuals struggled to find accurate data and timely updates from reliable platforms. This project addresses that need by designing an interactive web platform where users can access verified COVID-19 information and engage with others through discussions and comments. The inclusion of registration and login features ensures that each user has a personalized experience, maintaining security and traceability within the system.

From a technical perspective, the project serves as a comprehensive demonstration of full-stack web development integrated with quality assurance techniques. The system implements validation through regular expressions and functional testing to ensure the reliability of each module. By incorporating best practices in software testing, the project ensures that the website performs efficiently under different use cases, providing a secure, responsive, and error-free experience for end users.

1.2 Overview

The project integrates multiple web technologies to create a dynamic, data-driven system. The front end is developed using HTML for structure, CSS for layout and design, and JavaScript for dynamic and interactive elements. The back end utilizes PHP for server-side scripting and MySQL for data storage, ensuring efficient management of user credentials and comment details. This modular design allows users to register, log in, and interact seamlessly while maintaining consistent performance. The website's intuitive design focuses on accessibility, providing all users with a smooth browsing experience regardless of their device type.

In addition to development, this project places significant focus on **software quality assurance**. Different testing techniques are applied to validate form inputs, assess page responsiveness, and evaluate data security. Regular expression-based validation ensures that fields like email, password, and mobile number conform to correct formats. Functional testing is conducted to verify each feature, while usability testing ensures that the interface is simple and user-friendly. The website demonstrates how combining web development with structured testing methods can produce a dependable, high-quality application.

1.3 Problem Statement

During the COVID-19 pandemic, one of the major challenges faced by users was identifying and accessing reliable online information sources. Many websites either contained outdated data, lacked interactivity, or were prone to security issues due to poor validation. In addition, users were often unable to communicate or express concerns through these platforms, limiting community engagement. This project aims to solve these issues by providing a verified, user-interactive website that delivers accurate COVID-19 updates and guidelines while also enabling users to post and share their views securely.

The core problem also includes ensuring **data integrity** and **user input validation** in web forms. Without proper validation, web applications become vulnerable to incorrect data entries and potential security threats. By implementing regular expressions for validation and systematic software testing, the project mitigates such issues effectively. This approach not only enhances the overall user experience but also strengthens the reliability and security of the system, ensuring that the information and interactions on the website remain trustworthy and accurate.

1.4 Scope

The Dynamic COVID-19 Information Website project has a broad and well-defined scope covering both **development** and **testing** aspects of the system. The primary aim is to design, develop, and test a responsive and secure web platform that provides users with accurate information related to COVID-19. The website also enables user interaction through registration, login, and comment features, all of which are tested using various software testing techniques. The project not only focuses on the functional accuracy of the application but also emphasizes quality assurance, user validation, and overall performance of the system under different scenarios.

Chapter 2

System Architecture

2.1 Block Diagram of Web Application

System Architecture Flow

1. **User Interaction Layer**
   - The **user accesses the COVID-19 information website** via a browser.
   - Performs actions such as registration, login, or checking COVID-19 stats.

2. **Request Handling Layer**
   - The **browser (front end)** sends requests to the **PHP server** using HTTP/HTTPS.
   - These requests include form data, login details, or navigation actions.

3. **Server Processing Layer**
   - **PHP scripts** handle input validation, apply business logic, and control the workflow.
   - Example: checking credentials or processing registration.

4. **Database Operations Layer**
   - The **server communicates with MySQL** to store, retrieve, or validate information.
   - Example: verifying login details or retrieving COVID-19 updates.

5. **Response Generation Layer**
   - The **server sends a response** (success, failure, or content) back to the client browser.
   - This completes the **user request-response cycle**.

6. **Output Display Layer**
   - The **browser dynamically displays** the received data or message to the user (e.g., "Login successful").

7. **Testing Automation Layer (Selenium Integration)**
   - **Selenium WebDriver** simulates user actions (clicks, input, form submissions).
   - It automatically verifies if pages load correctly, forms validate properly, and data is fetched from the database as expected.

8. **Test Execution & Validation Layer**
   - Selenium scripts run test cases through browsers (Chrome/Firefox).
   - Each test is marked **Pass/Fail** based on the output comparison.

9. **Report Generation Layer**
   - Tools such as **TestNG / Extent Reports / Allure** generate visual reports (HTML/PDF).
   - Reports summarize test results (passed, failed, skipped).

10. **Defect Logging & Analysis Layer**
    - Failures are recorded in **JIRA / Excel** for debugging and tracking.
    - Developers fix issues, and the testing cycle continues.

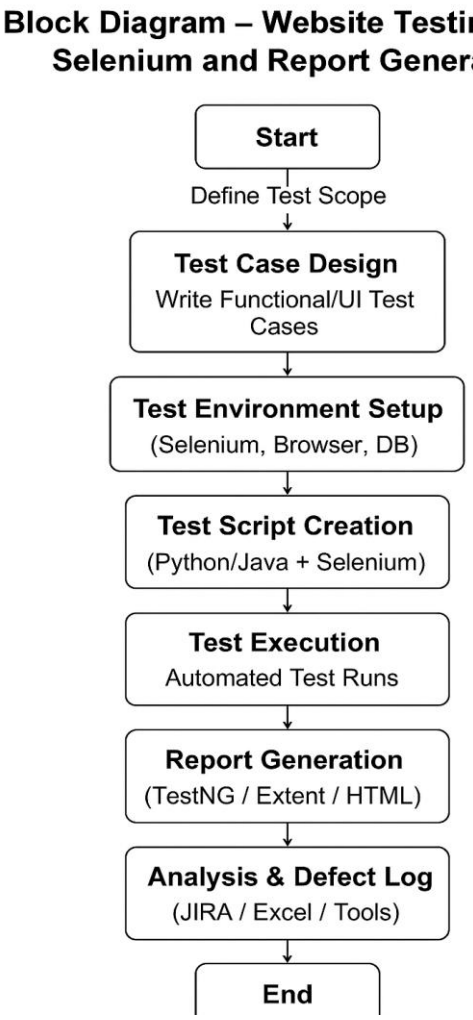
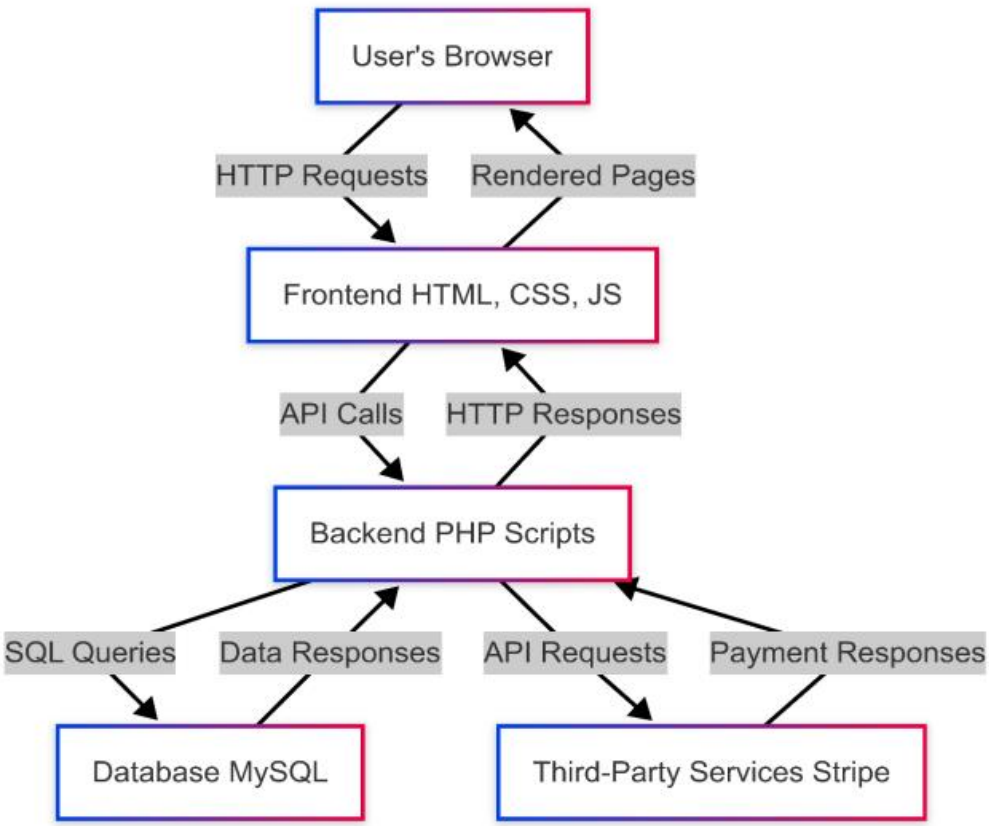


Figure: Website testing System Architecture

System Block Diagram Flow

1. **User Interface Layer (Frontend)** –
   Built using **HTML, CSS, and JavaScript**, this layer includes pages like:
   - Home (COVID-19 updates, prevention tips)
   - Login/Registration form
   - Comment section
2. **Backend Layer (Server-side)** –
   Implemented using **PHP**, it handles:
   - Form validation and data submission
   - Interaction with the database
   - Data retrieval (like user info or comments)
3. **Database Layer (MySQL)** –
   Stores all essential data:
   - User registration details
   - Login credentials
   - User comments or feedback
4. **Testing Layer (Selenium + TestNG)** –
   Used for automated testing of web functionality, such as:
   - Input validation (using regular expressions)
   - Login and form submission tests
   - Dashboard display checks



This project — *Dynamic COVID-19 Information Website* — uses a combination of front-end and back-end technologies for efficient design, development, and testing. The following technologies were selected for their reliability, scalability, and ease of integration.

2.2 Testing

Testing plays a crucial role in ensuring the reliability, functionality, and quality of the COVID-19 Information System. This project uses **manual and automated testing** approaches to identify defects, validate requirements, and verify that all modules perform as expected.

The primary testing framework and tools used include **Selenium WebDriver**, **TestNG**, and **JUnit**, which together automate the testing process, reduce manual effort, and improve accuracy.

1. Selenium WebDriver

Selenium WebDriver is an open-source automation framework used to perform functional testing of web applications across different browsers and platforms. It interacts directly with the web browser, simulating real user actions such as clicking buttons, entering text, and navigating between pages.

Key Features:

- Cross-browser testing support (Chrome, Firefox, Edge, etc.)
- Automates real user actions on web pages.
- Supports multiple languages like Java, Python, and C#.
- Provides integration with testing frameworks like TestNG and JUnit.

Usage in Project:

Selenium WebDriver was used to automate testing for:

- User registration and login.
- Dashboard navigation.
- Report download and search functionality.
- Form validation and regular expression checks.

2. TestNG Framework

TestNG (Testing Next Generation) is a testing framework inspired by JUnit and NUnit. It is widely used for managing and executing automated test cases in Java projects.

Key Features:

- Annotations like @Test, @BeforeClass, and @AfterMethod for test control.
- Supports grouping and prioritization of test cases.

- Parallel test execution.
- Generates detailed HTML and XML reports.

Usage in Project:

TestNG was integrated with Selenium to organize and execute automated test scripts. Each functionality — such as login, registration, and report generation — was tested through separate test classes defined in the testng.xml file.

3. Regular Expression Testing

Regular expressions (Regex) were used to validate input fields such as:

- **Email format**
- **Phone numbers**
- **Date of birth**
- **Government ID (Aadhaar, PAN, etc.)**

Purpose:

To ensure that data entered by the user adheres to the correct format, preventing invalid submissions and ensuring data integrity.

5. Types of Testing Performed

Type of Testing	Description	Tools Used
Functional Testing	Verified that each function of the application operates as expected.	Selenium, TestNG
Form Validation Testing	Checked user input forms using regular expressions for valid data formats.	Regex, Selenium
Integration Testing	Tested data flow between the frontend (HTML/PHP) and backend (MySQL).	Selenium
Regression Testing	Re-tested previously tested modules after adding new functionality.	TestNG
Performance Testing	Ensured that the system loads data efficiently under normal user conditions.	Manual Observation

6. Testing Environment

Component	Details
Operating System	Windows 10 / 11
Testing Tools	Selenium WebDriver, TestNG, JUnit
Browser Used	Google Chrome
Language	Java
IDE	Eclipse IDE for Java Developers
Database	MySQL
Server	XAMPP (Apache + MySQL + PHP)

Chapter 3

System Specification

3.1 System Functional Requirements

Hardware Requirements

Component	Specification
Processor	Intel Core i3 / i5 or higher
RAM	Minimum 4 GB (8 GB recommended)
Hard Disk	Minimum 2 GB of free space
Display	1366 × 768 resolution or higher
Input Devices	Keyboard, Mouse
Network	Internet Connection (for testing APIs and updates)

Software Requirements

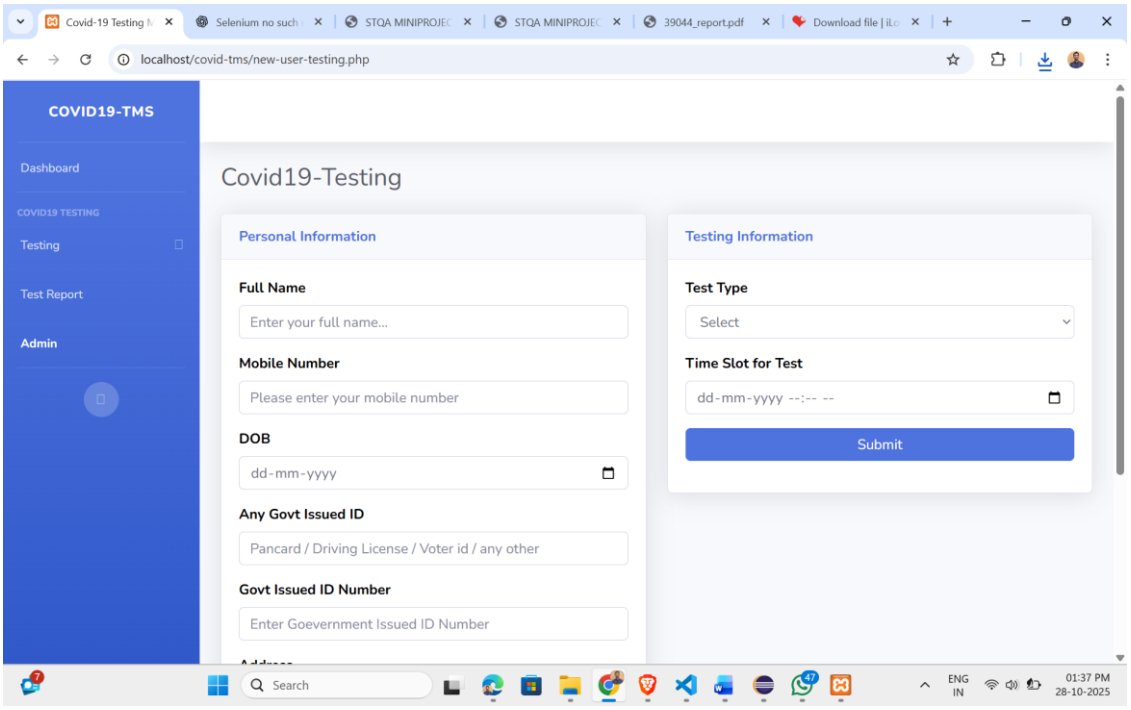
Software Component	Version / Details	Purpose / Description
Operating System	Windows 10 / 11	Platform for development and testing
Frontend Languages	HTML5, CSS3, JavaScript (ES6)	For UI design and interactivity
Backend Language	PHP v8.2.12	Server-side scripting for dynamic content
Database	MySQL v8.0.36	To store user data, comments, and reports
Local Server	XAMPP v8.2.12 (Apache + MySQL + PHP)	Used for local web hosting and database management
IDE / Code Editor	Visual Studio Code v1.94.0	For web development (HTML, CSS, JS, PHP)
Testing IDE	Eclipse IDE for Java Developers 2025-12 M1	For Selenium & TestNG automation testing
Testing Framework	TestNG v7.10.2	Framework for organizing automated tests
Automation Tool	Selenium WebDriver v4.24.0	For browser automation and web testing
Browser	Google Chrome v129.0	Browser used for testing

Chapter 4

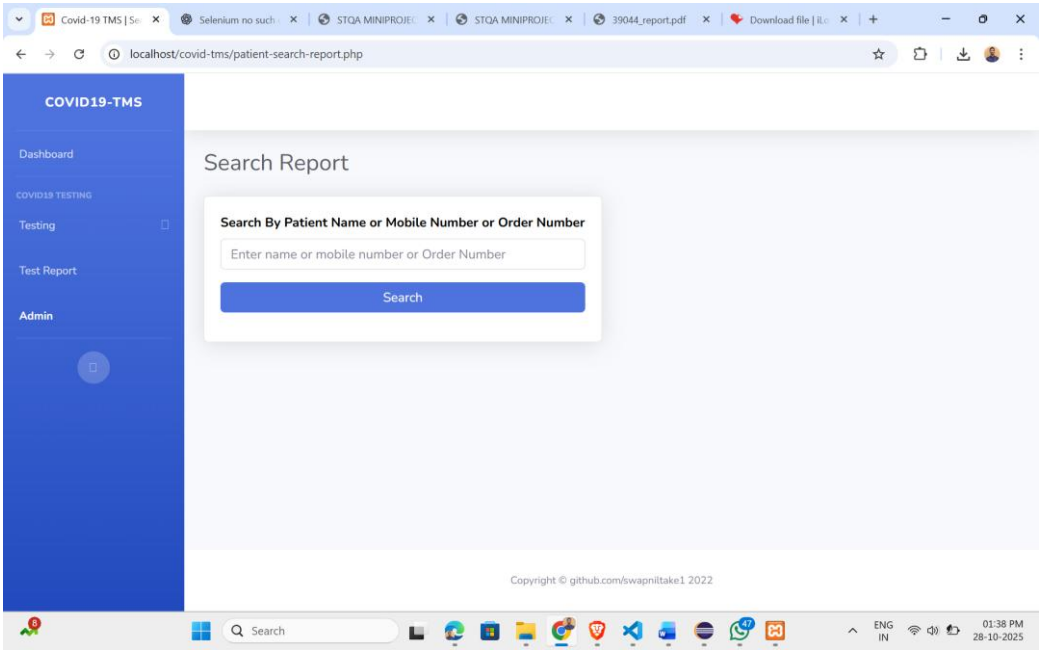
Webpage Screenshots (GUI)

4.1 Registeration and Login Interface

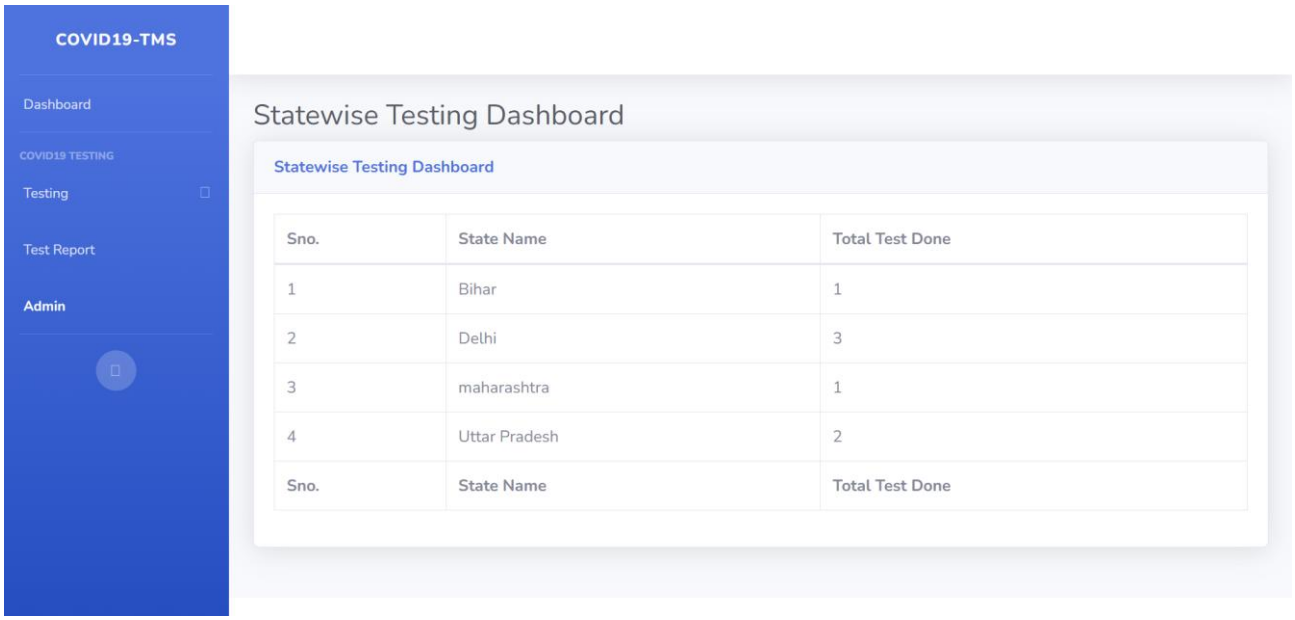
The system registers and login page, which is part of the developed system, is the primary interface for logging into the system. Both pages are the initial interface a user encounters in any computerized system. As a result, by properly designing and managing errors, the user might develop a positive attitude toward the rest of the system. The main register and login interface for for Covid 19.



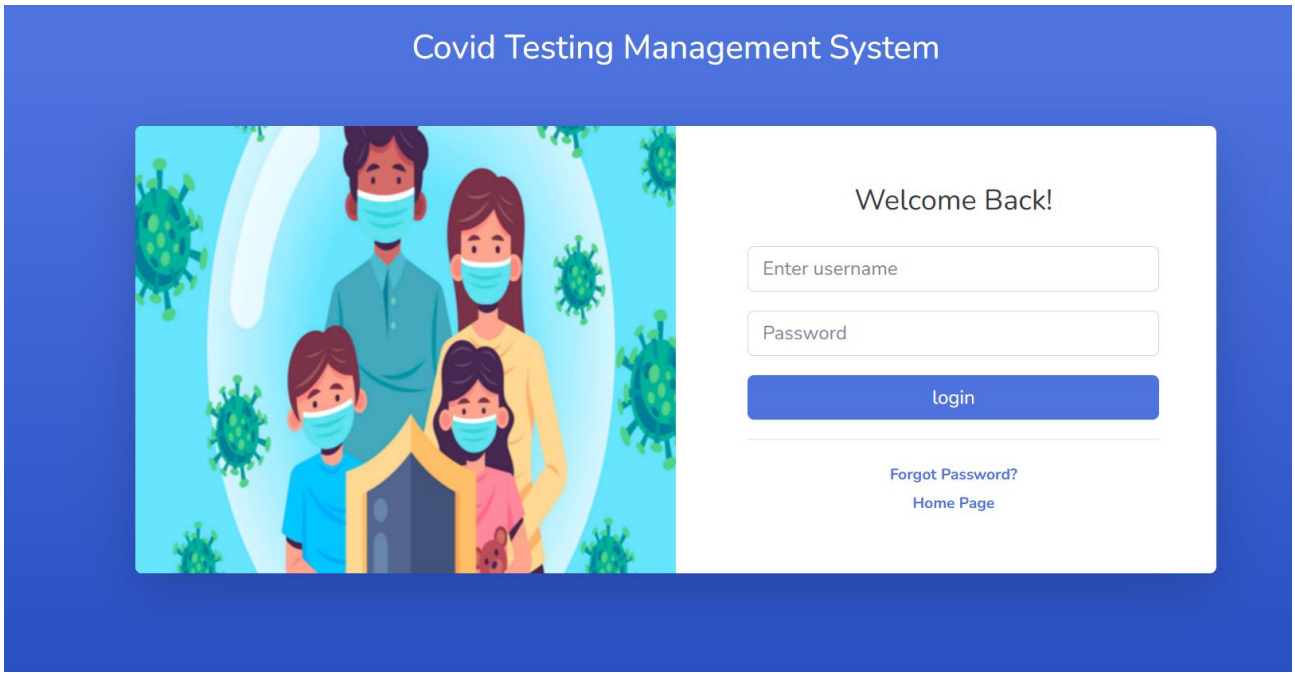
4.2 Search Report



4.3 Dashboard

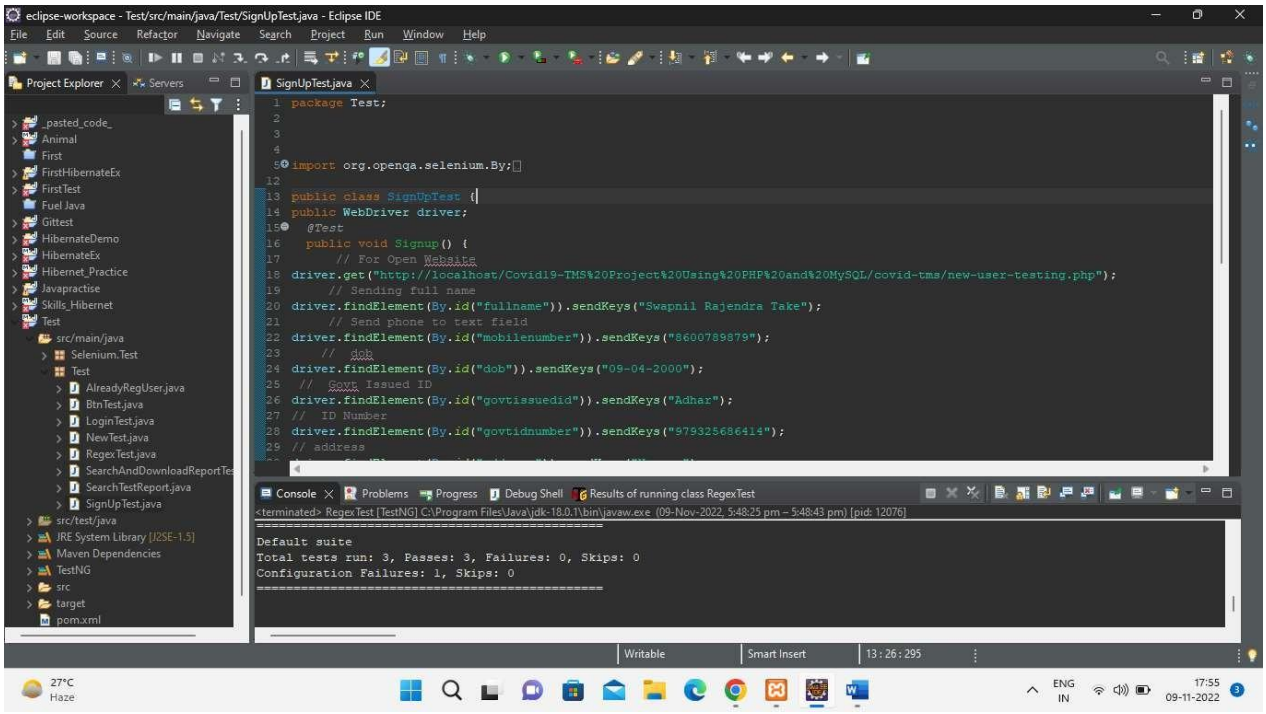


4.4Admin Pages

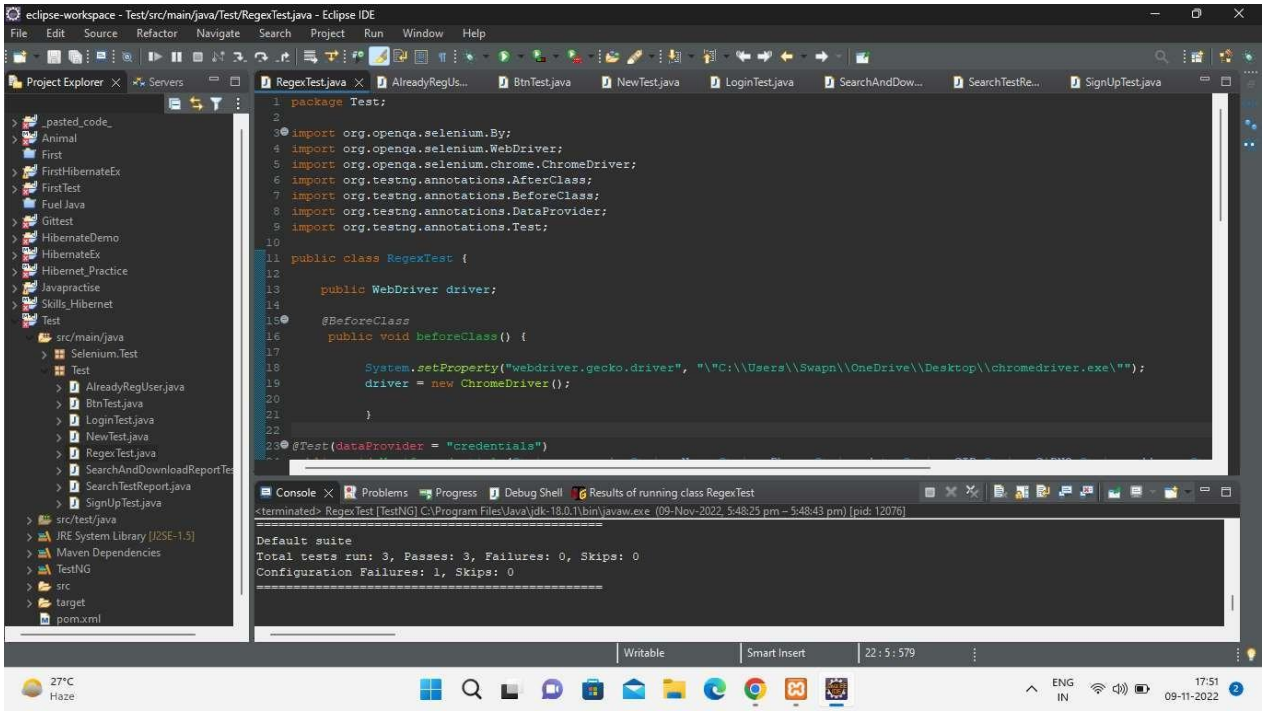


Chapter 5
Testing Screenshot

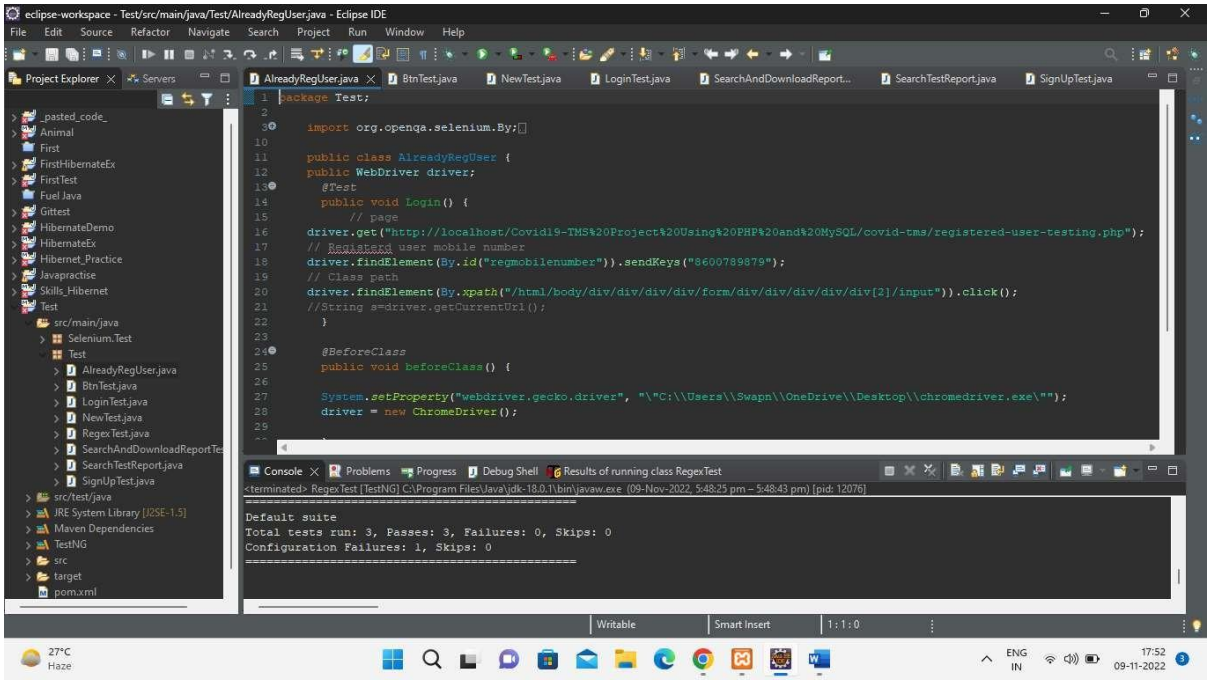
1.SignUp Page



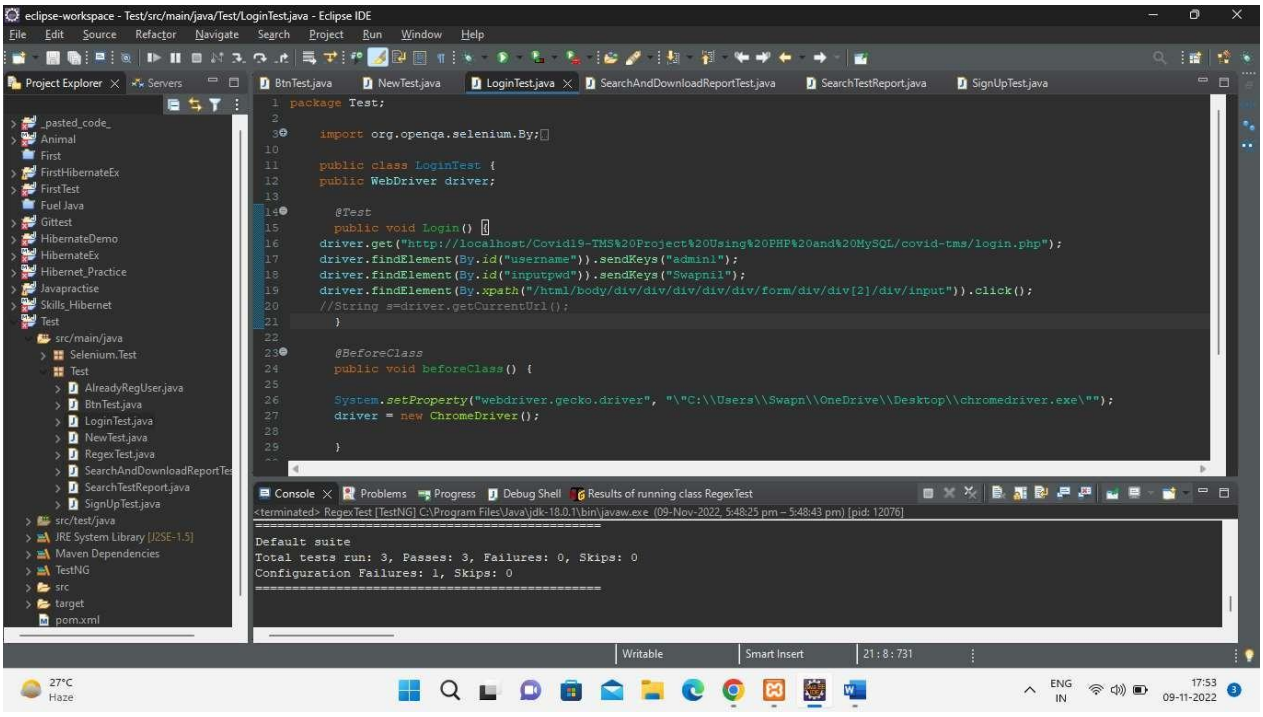
2. Regex test



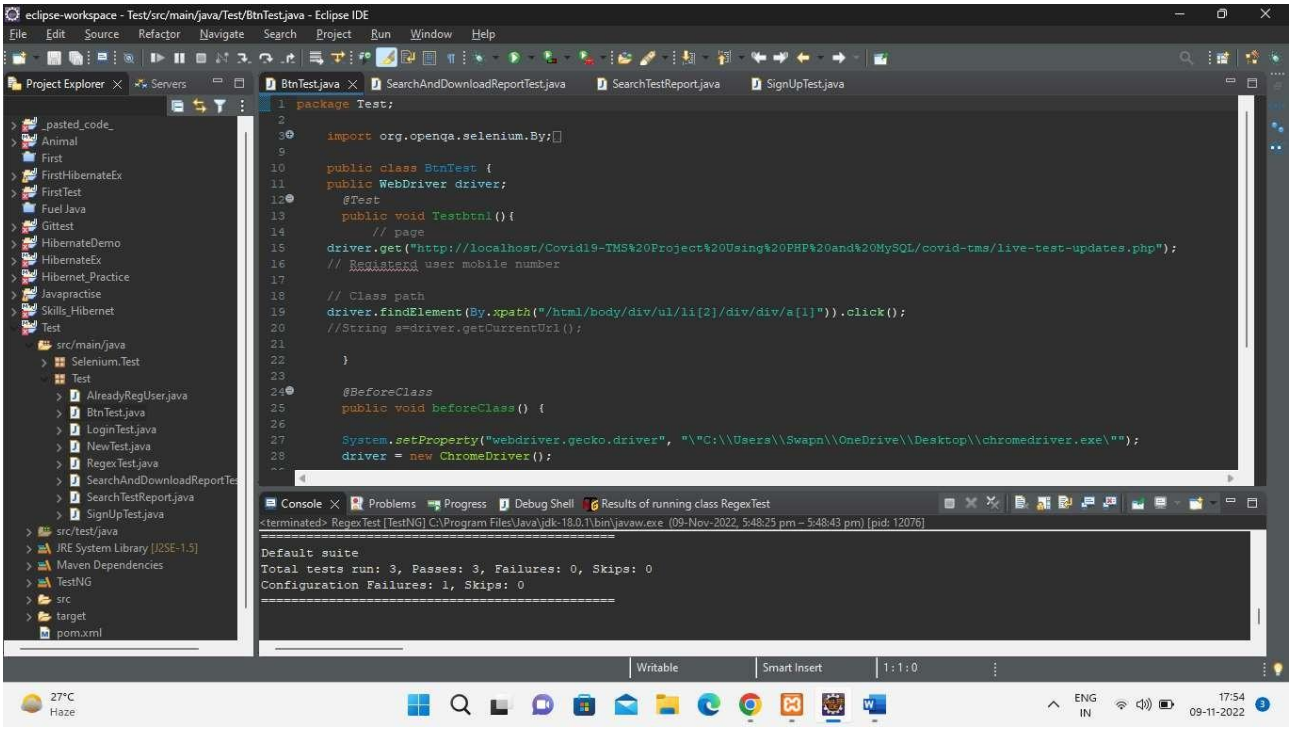
3. Registred user login :



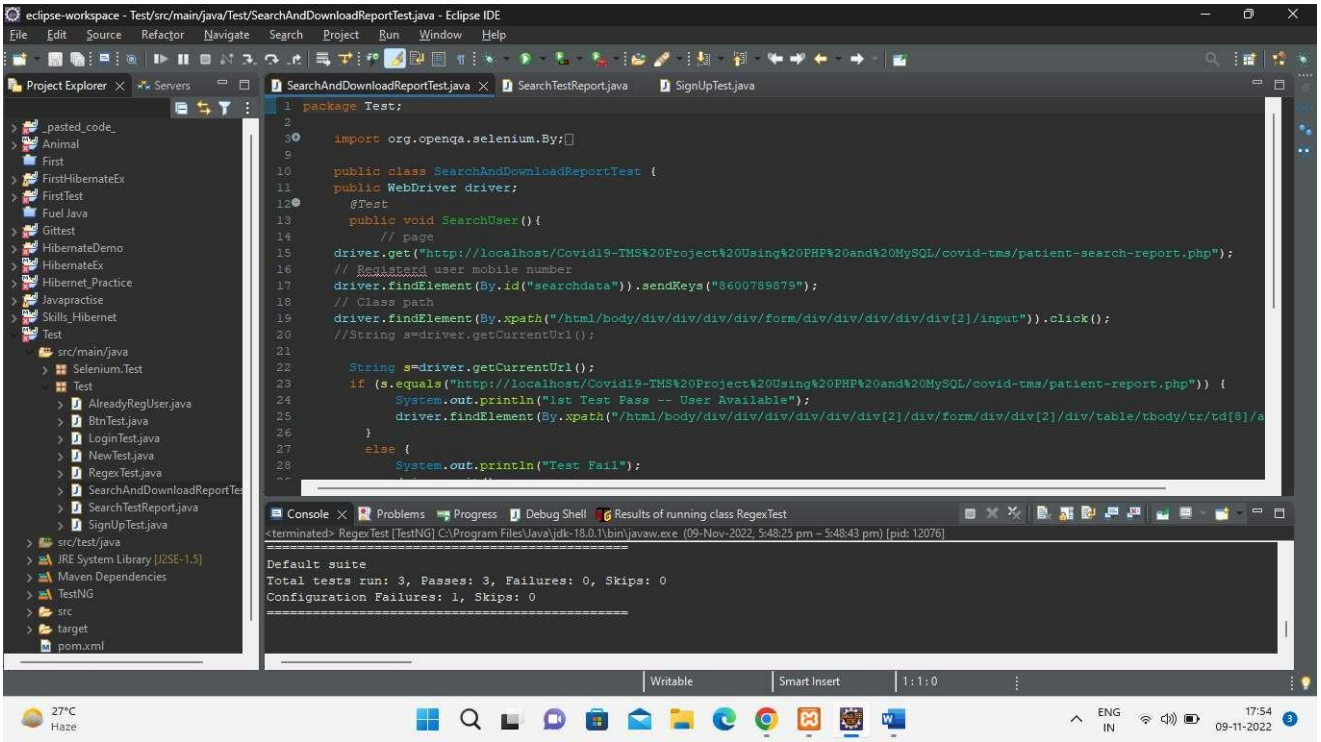
4. Login :



5. Button Test :



6. See Report :



Test Result Report

TABLE OF CONTENTS

1.0 INTRODUCTION

MODULES/ SCENARIOS	DESCRIPTION	% TCs EXECUTED	% TCs PASSED	TCs PENDING	PRIORITY	REMARKS

Project Dynamic webside of covid-19 information using HTML, CSS, JAVASCRIPT And PHP, MySQL database used to store user account, comment, and registration form details.

2.1 Selenium Testing Report

EXECUTION STATUS	COMPLETED/CANCELLED/PENDING
PASSED TESTCASES	7
FAILED TESTCASES	2
PENDING TERSTCASES	0
TEST CASES PLANNED	7

2.2 Regular Expression Testing

EXECUTION STATUS	% TCs EXECUTED	% TCs PASSED	TCs PENDING	PRIORITY	REMARKS
PASSED TESTCASES	3				
FAILED TESTCASES	1				
PENDING TERSTCASES	0				
TEST CASES PLANNED	2				

Email Validation	Email Id must have @ symbol.	100	100	00	HIGH	-
Phone No Validation	Phone no. must have 10 numbers.	100	100	00	HIGH	-
Patients Slot date validation	Patient's date must have next date from test apply.	100	100	00	HIGH	-

Chapter 5

Conclusion

The COVID-19 Information System project successfully demonstrates the use of modern web technologies to build a dynamic and informative website that provides users with accurate and up-to-date details about the COVID-19 pandemic. By integrating HTML5, CSS3, and PHP 8.2, the system offers a responsive interface and smooth server-side functionality, ensuring that users can easily access essential information such as testing updates, safety measures, and reports.

This project not only focuses on design and functionality but also emphasizes software quality and testing. The implementation of Selenium WebDriver with TestNG in the Eclipse IDE ensured that all modules were tested effectively for reliability, accuracy, and user interaction. Automated test cases were developed to verify login features, data handling, and dashboard operations — ensuring the system performs efficiently under various conditions.

The combination of XAMPP as a local development environment and MySQL 8.0 for data management allowed the system to handle database operations securely and seamlessly. Testing was further enhanced using Google Chrome with the ChromeDriver extension, ensuring real-world browser compatibility. These tools together provided a strong foundation for both development and quality assurance.

In conclusion, this mini project provided practical exposure to the complete Software Testing Life Cycle (STLC) and web development process. It helped in understanding how to integrate frontend and backend technologies with automation testing tools to build and validate a fully functional web application. The project fulfills its objective of demonstrating the importance of quality assurance in software projects while delivering a reliable and user-friendly COVID-19 information platform.

Chapter 6

References

1. W3Schools – *HTML Tutorial*. Available at: https://www.w3schools.com/html

2. W3Schools – *CSS Tutorial*. Available at: https://www.w3schools.com/css

3. Mozilla Developer Network (MDN) – *JavaScript Documentation*. Available at: https://developer.mozilla.org/

4. PHP Official Documentation – *PHP Manual*. Available at: https://www.php.net/manual/en/

5. MySQL Documentation – *MySQL Reference Manual*. Available at: https://dev.mysql.com/doc/

6. Apache Friends – *XAMPP Documentation*. Available at: https://www.apachefriends.org/

7. Microsoft – *Visual Studio Code Documentation*. Available at: https://code.visualstudio.com/docs

8. Selenium Official Site – *Selenium WebDriver Documentation*. Available at: https://www.selenium.dev/documentation/

9. TestNG Framework – *TestNG Documentation*. Available at: https://testng.org/doc/

10. World Health Organization (WHO) – *Coronavirus (COVID-19) Dashboard*. Available at: https://covid19.who.int/