



School of Information Technology and Engineering

SOFTWARE TESTING – ITE2004

WINTER SEMESTER 2022-23

**PROJECT REPORT
on**

DIGILOCKER SYSTEM

By

**JAISHREE - 20BIT0297
KALAMEGAM V - 20BIT0302
PRIYA N - 20BIT0305
PRIYADARSHINI.R - 20BIT0307
ABRETHA BEGAM.A - 20BIT0309**

submitted to

**Dr.IYAPPARAJA M
Associate Professor Grade 1
School of Information Technology and Engineering**

ABSTRACT :

One should always have a valid identity proof with them at all times. Having our Aadhaar card or PAN card or any other important document with us is a must. But carrying them around in our wallet or bag could be risky since it could be misplaced. And even worse, if it gets in the hands of a malicious person, our personal information could be at risk. Therefore, in the present world where everything is being digitized, it is time we do the same with all our important documents.

Digi locker is an Indian government-approved website which lets us store digital copies of all our important documents. The documents are issued to us by registered issuers and is therefore considered a legally valid soft copy. That is, it gets the same treatment as the original documents. So, whenever we need to present any document, we can simply use this digital copy. The main advantage of using Digi locker is that we are now able to access a whole range of official certificates/documents (Aadhar card, PAN card, driving license, birth certificate, class X and class XII mark sheets to name a few) from anywhere and at anytime.

Considering all the points we just mentioned, it makes us ponder over one question. Is this website secure? Does this website do what it actually is supposed to do? It becomes an inevitable task for us to test this website in order to get answers to these questions. At the end of the day, all our personal information is contained here and it could be compromised if the website has some defect or is not working accordingly to meet its requirements. **Performing software testing helps us identify if there are any problems or errors** in the system. The main need for software testing in this case is to ensure **security**.

INTRODUCTION:

Currently, in India, almost all of the government issued documents are in physical form across the country. This means every time a resident need to share the document with an agency to avail any service, an attested photo copy either in physical form or on scanned form is shared. Use of physical copies of document creates huge overhead in terms of manual verification, paper storage, manual audits, etc. incurring high cost and inconvenience. This creates problem for various agencies to verify the authenticity of these documents, thus, creating loopholes for usage of fake documents/certificates. Due to the nature of these documents not having a strong identity attached to it, anyone with same name can indeed misuse someone else's document.

Targeted at the idea of paperless governance, Digi Locker is a platform for issuance and verification of documents & certificates in a digital way, thus eliminating the use of physical documents.

The purpose of this document is to present a detailed description of the Digi Locker system. It will explain in purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both the stakeholders and the developers of the system.

The project is intended to create a website where citizens can download their all type of government approved certificates that's connected to our Aadhar number. **Citizen:** A legally recognized subject or national of a state or commonwealth, either native or naturalized.

LITERATURE SURVEY:

S.N o	Author,Year	Title	Methodology	Pros	Cons
1.	Urko Rueda Molina,Fitsum Kifetew, Annibale Panichella. 2018	Java Unit Testing Tool Competition - Sixth Round	Public contest repository,JUnit tools setup,CUTs, Execution frame,Testgeneration, Metrics computation,Combined analyses,Time budgets.	Generate the test case in Less time.	large budgets would likely result in higher overall performance for the blended check suites over the individual device outcome and the human-developed tests.
2.	Jean Petric,Tracy Hall ,David Bowes 2018	How Effectively Is Defective Code Actually Tested? An Analysis of JUnit Tests in Seven Open Source Systems	First extract the defects from seven open source projects the usage of the SZZ algorithm. Suit these defects with JUnit checks to pick out the share of defects that had been covered by JUnit tests. Additionally do the equal for non-defective code. Then usePrincipal Component Analysis and laptop gaining knowledge of to investigate the traits of JUnit assessments that had been profitable in identifying defects.	show that the number of methods touched by a JUnit test is strongly related to that test uncovering a defect	results show that a large number of defective methods are not covered by JUnit tests.

3.	D. Ma'ayan 2018	The quality of junit tests: an empirical study report	Introduce a static evaluation approach for analysing of large corpuses of code. For every corpus, traverse over all projects within it, and in every undertaking traverse over all its Java files. Each Java file is translated into an AST, and at some stage in a traversal over the AST, pattern, are, recognized. Finally, all the facts is saved in a CSV file for a similarly analysis.	These early results might demonstrae the importance of automation tools and the need for refactoring techniques for unit tests.	Here not find correlations between our results to traditional testing metrics such as code coverage by tests.
4.	R. MUKHERJEE AND K. S. PATNAIK 2019	Prioritizing JUnit Test Cases Without Coverage Information: An Optimization Heuristics Based Approach	Threads of validity, Java XML parser.	it appears clearly that application of optimization heuristics produce great benefift for ordering JUnit test cases at test method level.	Re-execution of all scheduled take a look at instances is not possible because of constraint trying out time window. We hope the positive results of our find out about will make utilization of optimization algorithms as a promising choice for prioritizing JUnit take a look at cases at take a look at approach level.

5.	Venkatesan, Praveen Kumar; Gade Rozario, Rikhil; Fiaidhi, Jinan, 2020	Junit frame work for unit testing.	Mongodb.	This improves programmer efficiency and system code reliability, which in effect decreases programmer frustration and debug time. This enhances the developer's efficiency.	The software may be used in as a whole or in components. If a product is to be appropriate for usage, each test must be passed.
6.	Xavier Devroey, Sebastiano Panichella, Alessio Gambi, 2020	Java Unit Testing Tool Competition - Eighth Round	Similar to previous edition Public contest repository , Execution environment, Test generation and time budget, Metrics computation, Combined analysis and comparison with manually written tests, Statistical analysis.	According to this paper most of the bugs mentioned in seventh edition are corrected.	Individual tool running on two different hardware may cause side effects. To reduce the impact resource has to be limited
7.	Maxim L. Gromov, Svetlana A. Prokopenko, Natalia V. Shabaldina, Andrey V. Laputenko Tomsk State University, Tomsk, Russia - 2019	Model Based JUnit Testing	Formal models of programs and tests, Mapping a Java class to a formal model, FSMTTest2JUnit	Based on this paper , the created tool is used to test telecommunication protocols ordinary Java application and students UML Diagram design implementation	Need to improve programmes FSMTTest2JUnit and TFSMTTest2\JUnit to automate the formal model's extraction from the Java class.

8.	Rakshith D C1, Dr. Manjunath A E2 , 2020	A Comprehensive Study on Automation Testing using JUnit	<p>During the configuration step, an object hierarchy for the test case is established, with each item standing in for a different use-case.</p> <p>Programmers or developers implement these subclasses, and each of their methods must be annotated with the phrase "Test" in order to establish the object hierarchy. The "Test" methods are where the domain-and application specific code is written.</p>	<p>Open source framework for java development. It is useful to those who lack knowledge about unit testing by generation of automation test case and execution.</p>	<p>Not applicable for customer-driven testing .Lacking technical knowledge it makes difficult to understand IDE errors.</p>
9.	Kifetew, Fitsum ; Devroey, Xavier; Rueda, Urko 2019	Java Unit Testing Tool Competition - Seventh Round	<p>Public contest repository10 , Execution frame, Test generation, Metrics computation, Combined analyses, Time budgets,</p>	<p>The ease with which a particular tool might be installed and used in the contest environment was demonstrated by the fact that the Contest infrastructure is dockerized.</p>	<p>Docker is not available in the cluster environment where we perform all of the studies, so we are unable to use it to complete the experiment. Due to this, keeping the two versions synchronised required more work</p>

10.	Danielle Gonzalez, Suzanne Prentice Mehdi Mirakhorli, 2018	A Fine-tuned Approach for Automated Conversion of JUnit Assertions English	Identify the assert's parameters, convert the expected and actual value parameters to English phrases, and combine these phrases into a single English sentence based on the assert's condition	English summaries of JUnit tests will become more valuable and accurate as a result of a thorough set of heuristics, improving test maintainability and traceability. The procedure was also put into practice and made available as the AssertConvert tool	need to improve parameter conversion heuristics, add heuristics for assert .
11.	Julian Harty, 2021	Security testing using JUnit and Perl scripts	A rigorous and thorough approach to identify and address potential vulnerabilities in software systems then using Perl scripts to automate the testing and analysis of the results.	Comprehensive testing, Costeffective, Repeatable and scalable, Integration with existing processes	Limited coverage, False positives/negatives, Technical expertise, Tool limitations

12	Jianwei Wu, James Clause,2023	Automated Identification Uniqueness in JUnit Tests	of The methodology has the potential to improve the efficiency of test suite maintenance by identifying redundant test cases and reducing the size of the test suite without sacrificing test coverage.	Improved Efficiency, Increased Test Effectiveness, High Accuracy, Flexibility, Practicality	The proposed methodology may not be scalable for very large test suites with a high number of test cases. The proposed methodology is specifically designed for JUnit test suites and may not be applicable to other types of test suites or testing frameworks.
13.	Christopher Vendome Maurício Aniche Christoph Treude Marco Aurélio Gerosa,2019	UniVerse: largescale JUnit-test analysis in the wild	Provides a comprehensive approach to analyzing JUnit tests and can be used to gain insights into how developers write tests and how testing practices evolve over time.	The methodology collects multiple metrics and data points about the tests, including their purpose, metrics, evolution, coverage, and statistical analysis. This provides a nuanced view of testing practices and trends.	The methodology is limited to analyzing JUnit tests in Java, and may not be applicable to other programming languages or testing frameworks. The methodology only analyzes the test code and does not consider other factors that may affect testing practices, such as developer experience or project constraints.

14.	Elvys Soares; Marcio Ribeiro; Rohit Gheyi; Guilherme Amaral; Andre Medeiros Santos, 2022	Refactoring Test Smells With JUnit 5 Why Should Developers Keep Up-to-Date		<p>Provides a comprehensive approach to identifying and refactoring test smells in JUnit 5 tests, with potential applications for improving the quality and maintainability of test suites.</p>		<p>The refactoring recommendations provided in the study are based on best practices and guidelines for effective test writing, and are supported by empirical evidence. The methodology can be extended to other testing frameworks and programming languages, providing a general approach to identifying and refactoring test smells that can be applied in a range of contexts.</p>
15.	Boni García, Diego Molina Paco, Saavedra, 2019	Selenium-Jupiter: A JUnit 5 extension for Selenium WebDriver		<p>Provides a practical approach to automating web testing using Selenium WebDriver and JUnit 5, with the Selenium-Jupiter extension providing additional functionality and convenience for managing the test environment</p>		<p>The extension provides built-in reporting capabilities, including capturing screenshots and videos of test execution, which can help diagnose failures and issues. The extension can be easily integrated</p>

			and analyzing results.	with existing tools and frameworks, including CI/CD pipelines and other testing frameworks, making it a flexible and versatile tool for web testing automation.w	
16.	An Experiment on the Effects of Modularity on Code Modification and Understanding. 2020	Ewan Tempero, Kelly Blincoe, Danielle Lottridge	Modularity has been referred to for decades with variations on the construct and a gap in evidence on the effects on programmers. they conducted an experiment to investigate the causal impact of modularity on software engineering performance. The results indicate that designs with high modularity code enable participants to achieve successful modification compared to designs with low modularity.	modularity on modifiability	programmes are developed according to international curriculum not developed based on outside of an organization

17.	A Big Data Platform for International Academic Conferences Based on Microservice Framework. 2018	Biao Yang , He Liu , Xuanrui Xiong , Shuaiqi Zhu , Amr Tolba and Xingguo Zhang	<p>In methodology phase, the Java checking out framework Junit used to be used to function unit checking out on a function-by-function basis. We used printouts and breakpoint debugging to take a look at the shape of loops and branches inside functions, calls between functions, and facts interactions between layers to make sure that no mistakes appear in the program. Additionally, the proper jogging surrounding</p>	<p>online tutorial convention provider platform primarily based on the proposed convention suggestion approach can furnish handy and quick convention organisation and participation offerings for convention organisation</p>	<p>scholars can't quickly obtain high-quality conference information and broaden their research fields.</p>
-----	--	--	---	--	---

18. On the Distribution of Test Smells i Open Android Applications: An Exploratory Study. 2022	Anthony Peruma,Khalid Saeed Almalki , Christian D. Newman, ,Mohamed Wiem Mkaouer ,Ali Ouni	<p>They investigated the diagram of unit assessments by way of analyzing the occurrence of check smells in Android apps and their have an impact on on the overall quality of the apps thru a set of quantitative, comparative and empirical experiments. they accrued facts for these experiments . Information about they separated take a look at from production archives is reachable through the mission intern et site .they carried out a two phased strategy that consisted of: data collection and scent detection. In the first phase, they collected datasets from more than one sources, whilst in the 2nd phase, they analyzed the gathered datasets to realize check smells, along side with the project metadata, wanted later for the experiments..</p>	helps developers to build and maintain better quality test cases for Android apps	represents a threat to test file's maintainability
19. Practitioner Perceptions of Ansible Test Smells. 2023	Yue Zhang,Fan Wu,Akond Rahman	<p>they answer RQ by conducting an online surveywith practitioners,who develop IaC manifests. they first asked practitioners about their experience in developing IaC manifests. Next, they asked how frequently practitioners test IaC</p>	Ansible manifests have relevance amongst practitioners	researchers didn't investigate why test smells have varying practitioner perceptions

			manifests using a Likert scale		
20.	A complete automation of unit testing of Java programs. 2019	Yoonsik Cheon, Myoung Yee Kim, Ashaveena Perumandla	JUnit is a open-source unit testing framework for Java and provides a way to organize test data and perform test execution. In JUnit, one has to write Java code, called a test class, that describes testdata, invokes the methods to be tested, and determines test results.	genetic algorithms to objectoriented programs by addressing the issues of genetic encoding, genetic operations, and fitness of objects.	limited to simple data types

TEST CASE REPORT:

Module 1: Signup

TEST CASE ID	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	STATUS
T1		Name is required	Name is required	Accepted
T2	Abretha	Success	Success	Accepted
T3		Name is required	Success	Failed
T4		Selected date, month ,year	Selected date, month ,year	Accepted
T5	Mar-2023	Select date	Select date	Accepted
T6	29--2023	Select month	Select month	Accepted
T7	29-mar-	Select year	Select year	Accepted
T8	--2003	Select date, month	Select date, month	Accepted
T9	-mar-	Select date, year	Select date, year	Accepted
T10	23--	Select month, year	Select month, year	Accepted
T11		Select gender	Select gender	Accepted

T12	Male	Success	Success	Accepted
T13		Mobile number is required	Success	Failed
T14	456789	Invalid mobile number	Invalid mobile number	Accepted
T15		Mobile number is required	Mobile number is required	Accepted
T16	78787878	Invalid number	Success	Failed
T17	678	Pin must be 6 digits	Pin must be 6 digits	Accepted
T18	67890123	Pin must be 6 digits	Pin must be 6 digits	Accepted
T19		Security pin is required	Security pin is required	Accepted
T20	12345	Pin must be 6 digits	success	Failed
T21		Email is required	Email is required	Accepted
T22	Priya.com	Invalid email	Invalid email	Accepted
T23	dharshini@	Invalid email	success	Failed
T24		Aadhar number is required	Aadhar number is required	Accepted

T25	12345678	Invalid Aadhar must be 12 digit only	Invalid Aadhar must be 12 digit only	Accepted
T26	12345678	Invalid Aadhar number	Invalid Aadhar number	Accepted

MODULE 2:MARKSHEET DOWNLOAD

TEST CASE ID	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	STATUS
TC1		Name is required	Name is required	Accepted
TC2	909345	success	success	Accepted
TC3		Roll number is required	Roll number is required	Accepted
TC4	909349	Invalid roll number	Invalid roll number	Accepted
TC5		Year is required	Year is required	Accepted
TC6	2016	success	success	Accepted
TC7	2013	2016-2019 documents only available	2016-2019 documents only available	Accepted
TC8	Regular	Success	Success	Accepted
TC9		Select certificate type	Select certificate type	Accepted

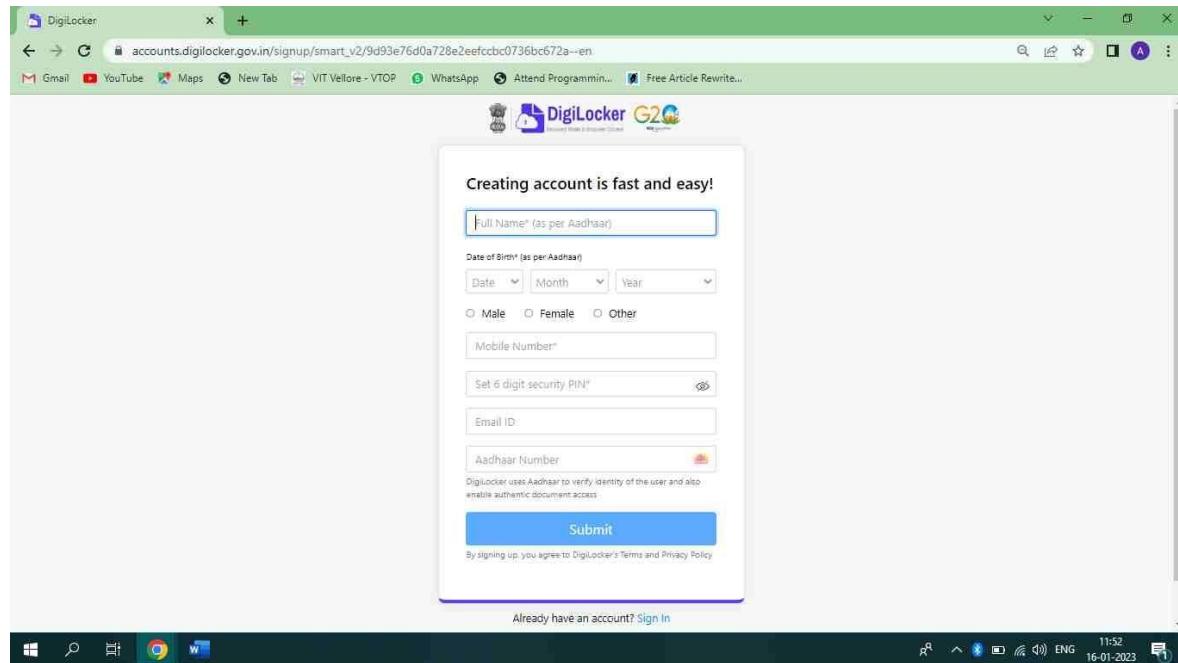
TC10		Select flag	Select flag	Accepted
TC11		Success	Success	Accepted
TC12	JAN	Success	Success	Accepted
TC13	JHK	Invalid month	Invalid month	Accepted
TC14		Month is required	Month is required	Accepted

MODULE:3 PAN CARD DOWNLOAD

TEST CASE ID	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	STATUS
TC1	123	Name is required	Name is required	Accepted
TC2		Gender is required.	Gender is required.	Accepted
TC3		DOB is required	DOB is required	Accepted
TC4	PDA414	Invalid PAN number	Invalid PAN number	Accepted
TC5		PAN number is required	PAN number is required	Accepted
TC6	PDA414KJII	Invalid PAN number	Invalid PAN number	Accepted

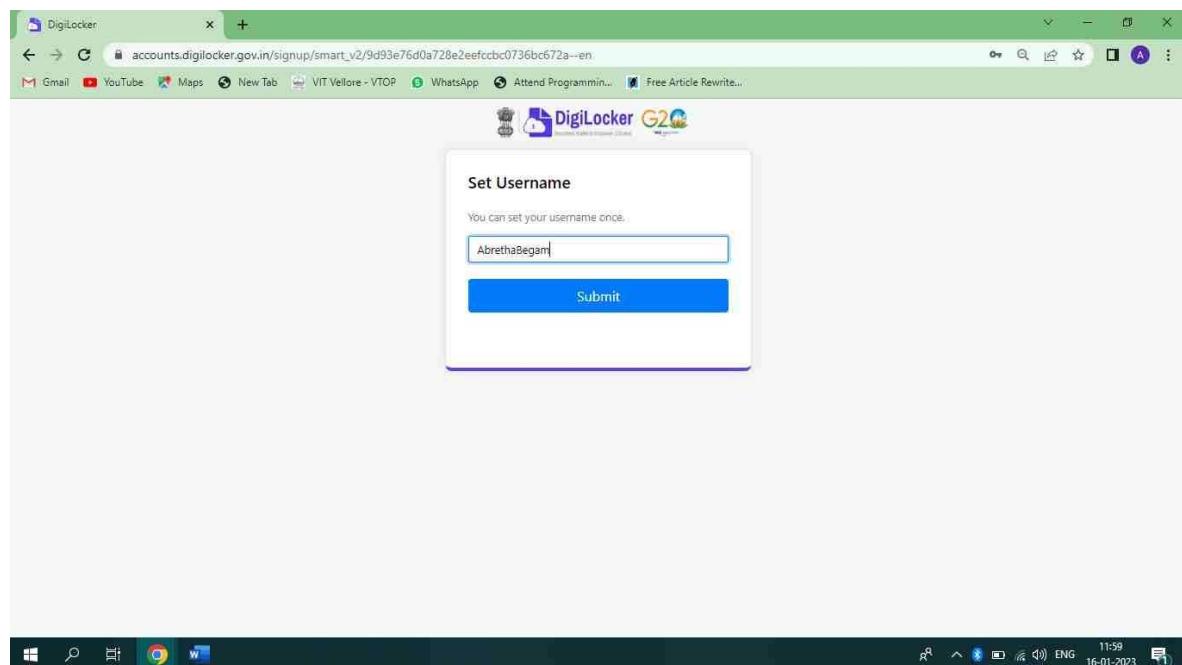
SAMPLE OUTPUT SCREENSHOTS :

By clicking Sign Up :



After Clicking Submit Verifying through OTP :

Setting Username:



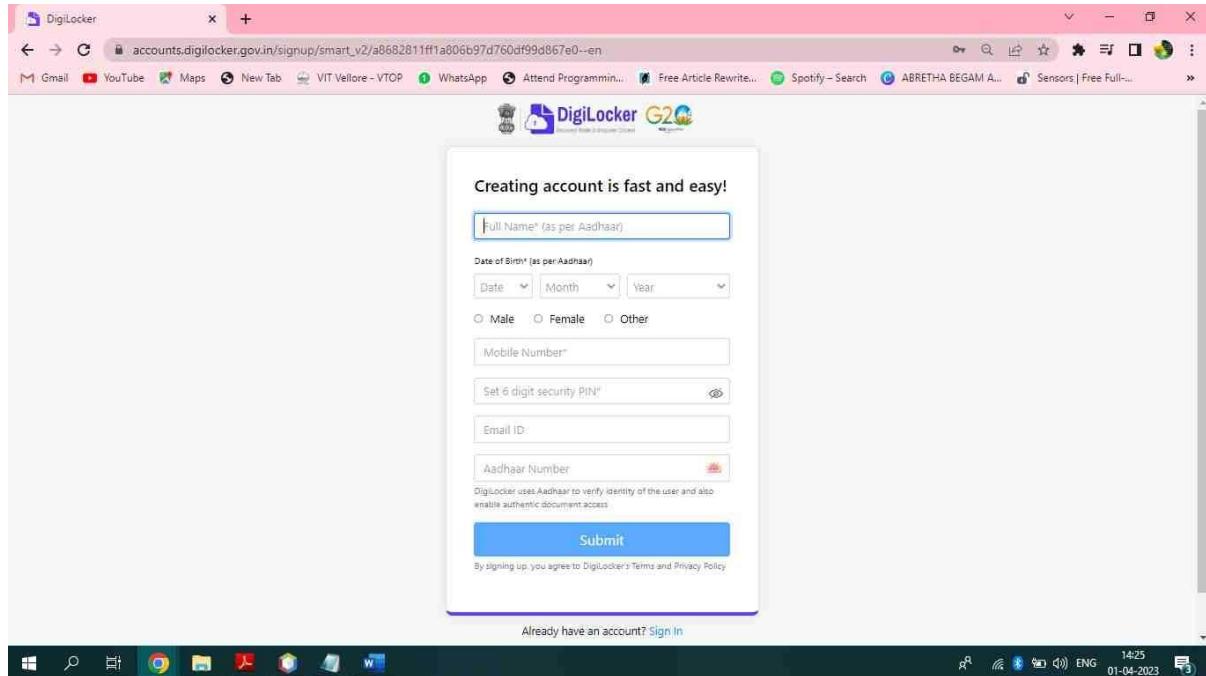
After Sign Up Sign In

Home Page :

The screenshot shows the DigiLocker Home Page in a web browser. The URL in the address bar is digilocker.gov.in/home. The page header displays the DigiLocker logo and the text "Issued Online eGoverment Services". The main content area starts with a welcome message: "Welcome, Abreetha Begam Aris Mohamed !". It states that DigiLocker 'Issued Documents' are at par with original documents as per IT ACT, 2000. Below this, there's a section titled "Inspiration for your first document" featuring four thumbnail images: "AADHAAR CARD" (a woman holding a card), "SSC Mark Sheet" (a girl holding a blue folder), "HSC Mark Sheet" (a man holding a book), and "Driving Licence" (a person on a motorcycle). The left sidebar contains navigation links: Home (selected), Issued Documents, Search Documents, Drive, DigiLocker Services, and About DigiLocker. The bottom section, "New in DigiLocker", highlights four new services: "Punjab and Sind Bank" (Account Statement and TDS Certificate), "Naval Pension Office (NAV PEN)" (Indian Navy Pension Documents), "Power Development Department, Govt. of J&K" (Electricity Bill), and "Single Window System, Jammu & Kashmir" (Various Certificates). The status bar at the bottom shows system icons and the date/time: 12:01 16-01-2023.

DIGILOCKER SYSTEM:

Module 1: Signup



Junittesting.java

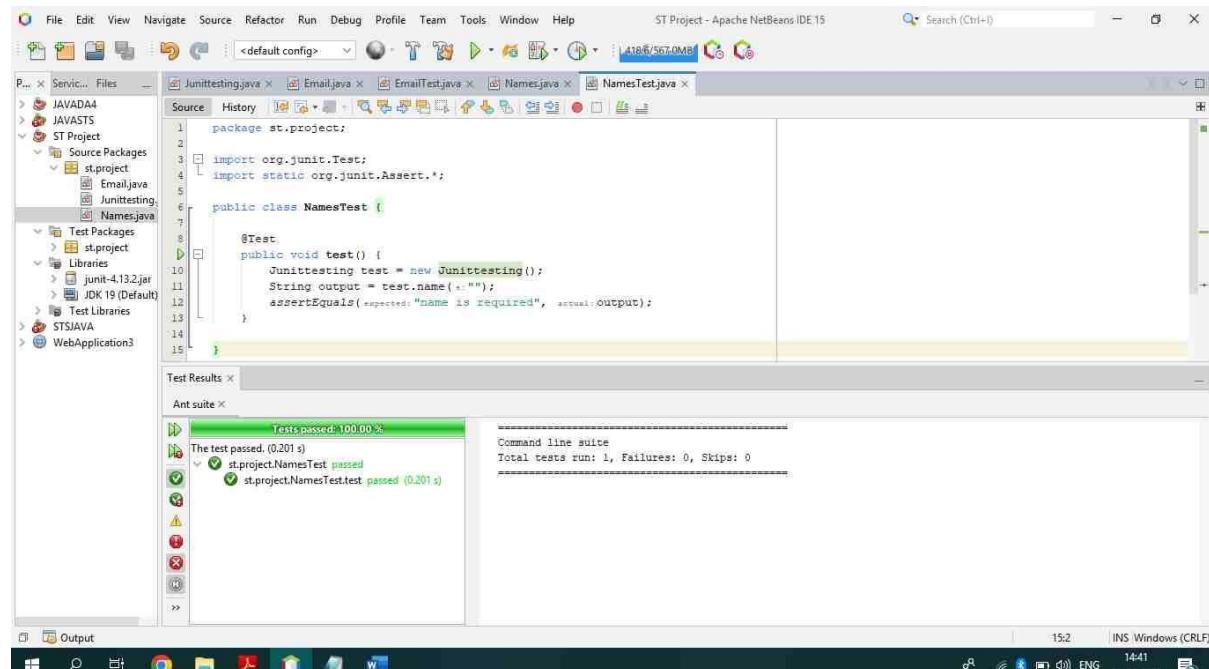
A screenshot of the NetBeans IDE interface. The title bar says "ST Project - Apache NetBeans IDE 15". The left sidebar shows a project structure with "Source Packages" containing "st.project" which has "Email.java", "Junittesting.java", and "EmailTest.java". The main editor area displays the "Junittesting.java" code. The code defines a class "Junittesting" with two methods: "name" and "dob". The "name" method checks if a string starts with a-zA-Z and returns "success" or "name is required". The "dob" method checks combinations of date, month, and year inputs and returns messages like "select date, month, year", "select date, month", "select date, year", "select month, year", "select date", "select month", or "select year". The bottom status bar shows the date as 01-04-2023 and the time as 14:29. The taskbar at the bottom includes icons for File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and Search (Ctrl+I).

Unit 1: Name

Test case ID: T1

Input:

Expected output: "name is required"



```
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help ST Project - Apache NetBeans IDE 15 Search (Ctrl+F) 1486/567.0MB 15:2 INS Windows (CRLF)
```

```
Source History <default config> Junittesting.java Email.java EmailTest.java Names.java NamesTest.java
```

```
1 package st.project;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class NamesTest {
7
8     @Test
9     public void test() {
10         Junittesting test = new Junittesting();
11         String output = test.name("");
12         assertEquals("expected: \"name is required\"", actual:output);
13     }
14 }
```

```
Test Results Ant suite
Tests passed: 100.00 %
The test passed. (0.201 s)
st.project.NamesTest passed
  ✓ st.project.NamesTest.test passed (0.201 s)

Command line suite
Total tests run: 1, Failures: 0, Skips: 0
```

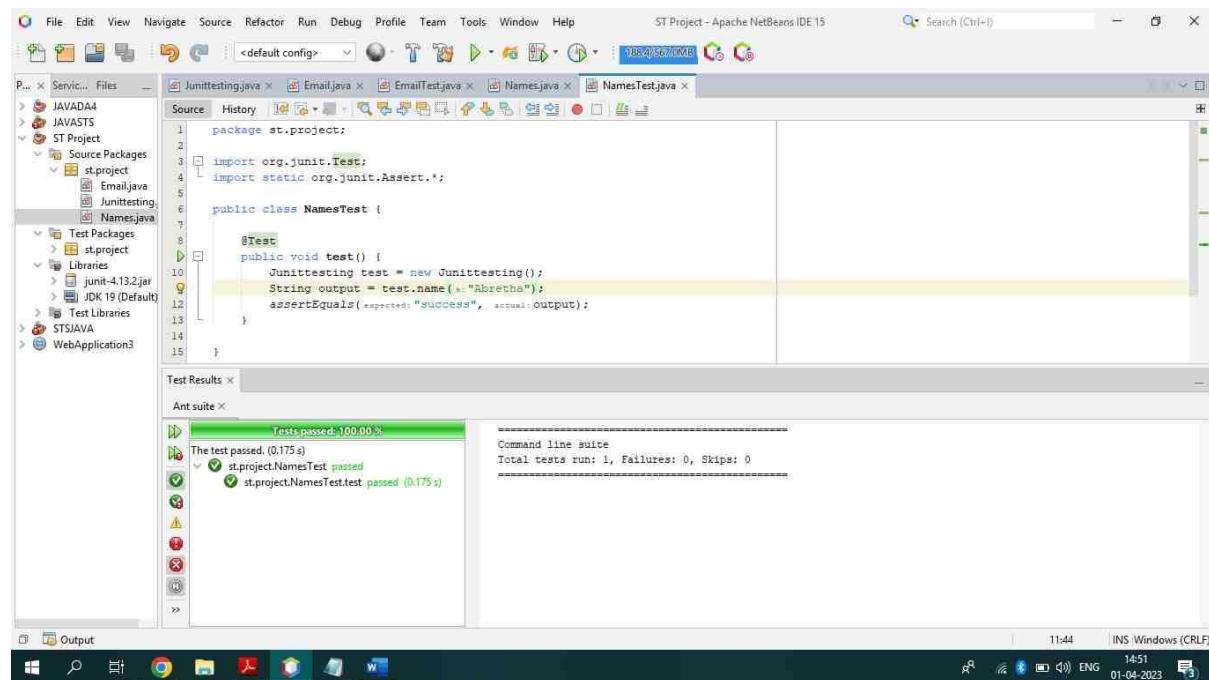
Output 15:2 INS Windows (CRLF)

Windows 14:41 01-04-2023

Test case ID: T2

Input: Abretha

Expected output: "Success"



```
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help ST Project - Apache NetBeans IDE 15 Search (Ctrl+F) 1084/567.0MB 11:44 INS Windows (CRLF)
```

```
Source History <default config> Junittesting.java Email.java EmailTest.java Names.java NamesTest.java
```

```
1 package st.project;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class NamesTest {
7
8     @Test
9     public void test() {
10         Junittesting test = new Junittesting();
11         String output = test.name("Abretha");
12         assertEquals("expected: \"Success\"", actual:output);
13     }
14 }
```

```
Test Results Ant suite
Tests passed: 100.00 %
The test passed. (0.175 s)
st.project.NamesTest passed
  ✓ st.project.NamesTest.test passed (0.175 s)

Command line suite
Total tests run: 1, Failures: 0, Skips: 0
```

Output 11:44 INS Windows (CRLF)

Windows 14:51 01-04-2023

Test case ID: T3 (Failed Test Case)

Input:

Expected output: "name is required"

The screenshot shows the Apache NetBeans IDE 15 interface. The project tree on the left shows a package named 'st.project' containing files like Junittesting.java, Email.java, EmailTest.java, Names.java, and NamesTest.java. The NamesTest.java file is open in the editor, displaying the following code:

```
1 package st.project;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class NamesTest {
7
8     @Test
9     public void test() {
10         Junittesting test = new Junittesting();
11         String output = test.name("");
12         assertEquals("expected:<success>, actual:<name is required>");
13     }
14 }
15
```

The 'Test Results' window at the bottom shows the test results for the 'Ant suite'.

Test Suite	Status	Message
Tests passed: 0.00 %	Green	No test passed, 1 test failed. (0.266 s)
st.project.NamesTest	Failed	> Failed: org.junit.ComparisonFailure: expected:<[success]> but was:<[name is required]>

On the right, the 'Command line suite' window shows the command-line output of the test run.

```
=====
Command line suite
Total tests run: 1, Failures: 1, Skips: 0
=====
```

Unit 2: DOB

Test case id: T4

Input:

Expected output: select date,month,year

The screenshot shows the Apache NetBeans IDE 15 interface. The project tree on the left shows a package named 'st.project' containing files like DOB.java, Email.java, EmailTest.java, Names.java, and NamesTest.java. The DOBTest.java file is open in the editor, displaying the following code:

```
1 package st.project;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class DOBTest {
7
8     @Test
9     public void test() {
10         Junittesting test=new Junittesting();
11         String output=test.dob("","","");
12         assertEquals("select date, month, year", actual.output);
13     }
14 }
15
```

The 'Test Results' window at the bottom shows the test results for the 'Ant suite'.

Test Suite	Status	Message
Tests passed: 100.00 %	Green	The test passed. (0.327 s)
st.project.DOBTest	Passed	st.project.DOBTest.test passed (0.327 s)

On the right, the 'Command line suite' window shows the command-line output of the test run.

```
=====
Command line suite
Total tests run: 1, Failures: 0, Skips: 0
=====
```

Test case id: T5

Input: -mar-2023

Expected output: select date

The screenshot shows the Apache NetBeans IDE 15 interface. The left pane displays the project structure with a file named Junittesting.java selected. The main editor pane contains the following Java code:

```
1 package st.project;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class DOBTest {
7     @Test
8     public void test() {
9         Junittesting test=new Junittesting();
10        String output=test.dob("","mar","");
11        assertEquals("Select date",output);
12    }
13 }
14
15
```

The bottom pane shows the "Test Results" window with the following output:

```
Tests passed: 100.00%
The test passed. (0.211 s)
st-project.DOBTest passed
  ✓ st-project.DOBTest.test passed (0.211 s)
```

On the right, the "Output - ST Project (test)" window shows:

```
Command line suite
Total tests run: 1, Failures: 0, Skips: 0
```

The system tray at the bottom right indicates the date as 01-04-2023 and the time as 15:18.

Test case id: T6

Input: 29- -2023

Expected output: select month

The screenshot shows the Apache NetBeans IDE 15 interface. The left pane displays the project structure with a file named Junittesting.java selected. The main editor pane contains the following Java code:

```
1 package st.project;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class DOBTest {
7     @Test
8     public void test() {
9         Junittesting test=new Junittesting();
10        String output=test.dob("29","","");
11        assertEquals("Select month",output);
12    }
13 }
14
15
```

The bottom pane shows the "Test Results" window with the following output:

```
Tests passed: 100.00%
The test passed. (0.164 s)
st-project.DOBTest passed
  ✓ st-project.DOBTest.test passed (0.164 s)
```

On the right, the "Output - ST Project (test)" window shows:

```
Command line suite
Total tests run: 1, Failures: 0, Skips: 0
```

The system tray at the bottom right indicates the date as 01-04-2023 and the time as 15:19.

Test case id: T7

Input: 29- mar-

Expected output: select year

The screenshot shows the Apache NetBeans IDE 15 interface. The left sidebar displays a project structure with files like JAVADA4, JAVASTS, and ST Project. The main editor window contains the code for DOBTest.java:

```
1 package st.project;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6
7 public class DOBTest {
8     @Test
9     public void test() {
10         Junittesting test=new Junittesting();
11         String output=test.dob("29", "mar");
12         assertEquals("Select year", output);
13     }
14 }
```

The bottom right corner shows the system tray with the date and time: 01-04-2023 15:22.

Test case id: T8

Input: - -2003

Expected output: select date, month

The screenshot shows the Apache NetBeans IDE 15 interface. The left sidebar displays a project structure with files like JAVADA4, JAVASTS, and ST Project. The main editor window contains the code for DOBTest.java:

```
1 package st.project;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6
7 public class DOBTest {
8     @Test
9     public void test() {
10         Junittesting test=new Junittesting();
11         String output=test.dob("", "", "2023");
12         assertEquals("Select date, month", output);
13     }
14 }
```

The bottom right corner shows the system tray with the date and time: 01-04-2023 15:24.

Test case id: T9

Input: - mar-

Expected output: select date, year

The screenshot shows the Apache NetBeans IDE 15 interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and a search bar. The title bar says "ST Project - Apache NetBeans IDE 15". The left sidebar displays the project structure with packages like JAVADA4, JAVASTS, and ST Project, and source files like DOB.java, Email.java, Junittesting.java, and Names.java. The main editor window contains the following Java code:

```
1 package st.project;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class DOBTest {
7     @Test
8         public void test() {
9             Junittesting test=new Junittesting();
10            String output=test.dob(- "", "+ "" );
11            assertEquals(expected:"Select date, year", actual:output);
12        }
13    }
14
15 }
```

The "Test Results" panel shows the test suite "Ant suite" with a green bar indicating "Tests passed: 100.00 %". It lists two test cases under "st.project.DOBTest": "passed" and "test" which also passed. The command line output shows "Total tests run: 1, Failures: 0, Skips: 0". The bottom status bar shows the time as 15:1 and the date as 01-04-2023.

Test case id: T10

Input: 23- -

Expected output: select month, year

The screenshot shows the Apache NetBeans IDE 15 interface, similar to the previous one. The project structure and code are identical to Test Case T9. The main difference is in the input value used in the test method. The code in the editor is:

```
1 package st.project;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class DOBTest {
7     @Test
8         public void test() {
9             Junittesting test=new Junittesting();
10            String output=test.dob(- "23", "+ "" );
11            assertEquals(expected:"Select month, year", actual:output);
12        }
13    }
14
15 }
```

The "Test Results" panel shows the test suite "Ant suite" with a green bar indicating "Tests passed: 100.00 %". It lists two test cases under "st.project.DOBTest": "passed" and "test" which also passed. The command line output shows "Total tests run: 1, Failures: 0, Skips: 0". The bottom status bar shows the time as 13:10 and the date as 01-04-2023.

Unit 3: Gender

Test case id: T11

Input:

Expected output: select gender

The screenshot shows the Apache NetBeans IDE 15 interface. The left sidebar displays the project structure under 'ST Project' with files like Junittesting.java, Email.java, DOB.java, Gender.java, Names.java, and GenderTest.java. The main editor window shows the code for GenderTest.java:

```
1 package st.project;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class GenderTest {
7     @Test
8     public void test() {
9         Junittesting test=new Junittesting();
10        String output=test.gender(" ");
11        assertEquals("Select gender", output);
12    }
13 }
```

The 'Test Results' window below shows the test results for 'Ant suite':

- The test passed. (0.245 s)
- st-project.GenderTest passed
- st-project.GenderTest.test passed (0.245 s)

On the right, the 'Command line suite' window shows:

```
=====
Total tests run: 1, Failures: 0, Skips: 0
```

Test case id: T12

Input: Male

Expected output: success

The screenshot shows the Apache NetBeans IDE 15 interface. The left sidebar displays the project structure under 'ST Project' with files like Junittesting.java, Email.java, DOB.java, Gender.java, Names.java, and GenderTest.java. The main editor window shows the code for GenderTest.java:

```
1 package st.project;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class GenderTest {
7     @Test
8     public void test() {
9         Junittesting test=new Junittesting();
10        String output=test.gender("Male");
11        assertEquals("Success", output);
12    }
13 }
```

The 'Test Results' window below shows the test results for 'Ant suite':

- The test passed. (0.2 s)
- st-project.GenderTest passed
- st-project.GenderTest.test passed (0.2 s)

On the right, the 'Command line suite' window shows:

```
=====
Total tests run: 1, Failures: 0, Skips: 0
```

Test case id: T13(Failed Test Case)

Input:

Expected output: mobile number is required

The screenshot shows the Apache NetBeans IDE 15 interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and a search bar. The title bar says "ST Project - Apache NetBeans IDE 15". The left sidebar displays the project structure with packages like JAVADA4, JAVASTS, and ST Project, and source files such as DOB.java, Email.java, Gender.java, Junittesting.java, Names.java, and others. The main editor window contains the code for GenderTest.java:

```
1 package st.project;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class GenderTest {
7
8     @Test
9     public void test() {
10         Junittesting test=new Junittesting();
11         String output=test.gender(" ");
12         assertEquals("Success",actual,output);
13     }
14 }
15
16 }
```

The code has several syntax errors, notably at line 11 where there is a missing closing brace. The "Test Results" panel shows the outcome of the test run:

Tests passed: 0.00 %

No test passed, 1 test failed. (0.202 s)

st.project.GenderTest Failed

> st.project.GenderTest.test Failed: org.junit.ComparisonFailure: expected:<Success> but was:<select gender>

The command line output shows:

Command line suite

Total tests run: 1, Failures: 1, Skips: 0

Unit 4: Mobile Number

Test case id: T14

Input: 456789

Expected output: invalid mobile number

The screenshot shows the Apache NetBeans IDE 15 interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and a search bar. The title bar says "ST Project - Apache NetBeans IDE 15". The left sidebar displays the project structure with packages like JAVADA4, JAVASTS, and ST Project, and source files such as DOB.java, Email.java, Gender.java, Junittesting.java, Mobile.java, Names.java, and others. The main editor window contains the code for MobileTest.java:

```
1 package st.project;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class MobileTest {
7
8     @Test
9     public void test() {
10         Junittesting test = new Junittesting();
11         String output = test.mobile("456789");
12         assertEquals("invalid mobile number", actual, output);
13     }
14 }
15
16 }
```

The code has a syntax error at line 11 where there is a missing closing brace. The "Test Results" panel shows the outcome of the test run:

Tests passed: 100.00 %

The test passed. (0.226 s)

st.project.MobileTest passed

> st.project.MobileTest.test passed (0.226 s)

The command line output shows:

Command line suite

Total tests run: 1, Failures: 0, Skips: 0

Test case id: T15

Input:

Expected output: mobile number is required

The screenshot shows the Apache NetBeans IDE 15 interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and ST Project - Apache NetBeans IDE 15. The search bar says "Search (Ctrl+I)". The left sidebar displays a project structure with Source Packages, Test Packages, Libraries, and various Java files like Names.java, DOB.java, Email.java, Gender.java, Junittesting.java, Mobile.java, and NamesTest.java. The main editor window shows the code for MobileTest.java:

```
1 package st.project;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class MobileTest {
7
8     @Test
9     public void test() {
10         Junittesting test = new Junittesting();
11         String output = test.mobile("");
12         assertEquals("mobile number is required", actual.output);
13     }
14 }
15
16
```

The "Test Results" panel shows the outcome of the test run:

- Ant suite:
 - Tests passed: 100.00%
 - The test passed. (0.223 s)
 - st.project.MobileTest passed
 - st.project.MobileTest.test passed (0.223 s)
- Command line suite
 - Total tests run: 1, Failures: 0, Skips: 0

The status bar at the bottom indicates "Output - ST Project (test)" and "8:56 INS Windows (CRLF)".

Test case id: T16 (Failed Test Case)

Input: 78787878

Expected output: Invalid mobile number

The screenshot shows the Apache NetBeans IDE 15 interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and ST Project - Apache NetBeans IDE 15. The search bar says "Search (Ctrl+I)". The left sidebar displays a project structure with Source Packages, Test Packages, Libraries, and various Java files like Names.java, DOB.java, Email.java, Gender.java, Junittesting.java, Mobile.java, and NamesTest.java. The main editor window shows the code for MobileTest.java:

```
1 package st.project;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class MobileTest {
7
8     @Test
9     public void test() {
10         Junittesting test = new Junittesting();
11         String output = test.mobile("78787878");
12         assertEquals("success", actual.output);
13     }
14 }
15
16
```

The "Test Results" panel shows the outcome of the test run:

- Ant suite:
 - Tests passed: 0.00%
 - No test passed, 1 test failed. (0.224 s)
 - st.project.MobileTest Failed
 - st.project.MobileTest.test Failed: org.junit.ComparisonFailure: expected:<success> but was:<[invalid mobile number]>
- Command line suite
 - Total tests run: 1, Failures: 1, Skips: 0

The status bar at the bottom indicates "Output - ST Project (test)" and "12:41 INS Windows (CRLF)".

Unit 5: Security pin

Test case id: T17

Input: 678

Expected output: pin must be 6 digits

The screenshot shows the Apache NetBeans IDE interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and a search bar. The title bar indicates "ST Project - Apache NetBeans IDE 15". The left sidebar displays the project structure under "Source Packages" for the "st.project" package, including files like DOB.java, Email.java, Gender.java, Junittesting.java, Mobile.java, Names.java, and Security.java. The main editor window contains the following Java code:

```
1 package st.project;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class SecurityTest {
7
8     @Test
9     public void test() {
10         Junittesting test=new Junittesting();
11         String output=test.pin("678");
12         assertEquals(expected:"Security pin is required", actual.output);
13     }
14 }
```

The "Test Results" panel shows the test suite status: "Tests passed: 100.00%". It details two test cases: "st.project.SecurityTest passed" and "st.project.SecurityTest.test passed (0.258 s)". The command line output shows "Total tests run: 1, Failures: 0, Skips: 0". The bottom status bar shows the date and time as 01-04-2023.

Test case id: T18

Input: 67890123

Expected output: pin must be 6 digits

The screenshot shows the Apache NetBeans IDE interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and a search bar. The title bar indicates "ST Project - Apache NetBeans IDE 15". The left sidebar displays the project structure under "Source Packages" for the "st.project" package, including files like DOB.java, Email.java, Gender.java, Junittesting.java, Mobile.java, Names.java, and Security.java. The main editor window contains the same Java code as the previous screenshot:

```
1 package st.project;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class SecurityTest {
7
8     @Test
9     public void test() {
10         Junittesting test=new Junittesting();
11         String output=test.pin("67890123");
12         assertEquals(expected:"Security pin is required", actual.output);
13     }
14 }
```

The "Test Results" panel shows the test suite status: "Tests passed: 100.00%". It details two test cases: "st.project.SecurityTest passed" and "st.project.SecurityTest.test passed (0.258 s)". The command line output shows "Total tests run: 1, Failures: 0, Skips: 0". The bottom status bar shows the date and time as 01-04-2023.

Test case id: T19

Input:

Expected output: security pin is required

The screenshot shows the NetBeans IDE interface with the following details:

- Project Structure:** The left pane shows a project named "ST Project" with packages like "st.project" containing classes such as "DOB.java", "Email.java", "Gender.java", "Junittesting.java", "Mobile.java", "Names.java", and "Security.java".
- Source Editor:** The main editor window displays the code for "SecurityTest.java". The code defines a test method that checks if the output is "Security pin is required" when the input is an empty string.
- Test Results:** The "Test Results" panel at the bottom shows a green bar indicating "Tests passed: 100.00%". It lists two test cases under "st.project.SecurityTest": "st.project.SecurityTest passed" and "st.project.SecurityTest.test passed (0.258 s)".
- Output:** The "Output - ST Project (test)" panel shows the command line output of the test run.

Test case id: T20 (Failed Test case)

Input: 12345

Expected output: success

The screenshot shows the NetBeans IDE interface with the following details:

- Project Structure:** The left pane shows a project named "ST Project" with packages like "st.project" containing classes such as "DOB.java", "Email.java", "Gender.java", "Junittesting.java", "Mobile.java", "Names.java", and "Security.java".
- Source Editor:** The main editor window displays the code for "SecurityTest.java". The code defines a test method that checks if the output is "Success" when the input is "12345".
- Test Results:** The "Test Results" panel at the bottom shows a red bar indicating "Tests passed: 0/00%". It lists one test case under "st.project.SecurityTest": "No test passed, 1 test failed. (0.216 s)".
- Output:** The "Output - ST Project (test)" panel shows the command line output of the test run, including the failure message from JUnit.

Unit 6: Email

Test case id: T21

Input:

Expected output: email is required

The screenshot shows the Apache NetBeans IDE interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and ST Project - Apache NetBeans IDE 15. The toolbar contains various icons for file operations like Open, Save, and Print. The left sidebar displays the project structure under 'Source Packages' for the 'st.project' package, showing files like DOB.java, Emails.java, Gender.java, Junittesting.java, Mobile.java, Names.java, and Security.java. Below this is a 'Test Packages' section with 'st.project'. The central workspace shows a Java code editor with the following code:

```
1 package st.project;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6
7 public class EmailsTest {
8
9     @Test
10    public void test() {
11        Junittesting test=new Junittesting();
12        String output=test.email(" ");
13        assertEquals("email is required", output);
14    }
15
16 }
17
```

The code uses JUnit annotations to run a single test method named 'test' that checks if an empty string is considered invalid. The 'Test Results' panel at the bottom shows a green bar indicating 'Tests passed: 100.00%' and a detailed log:

```
Tests passed: 100.00%
The test passed. (0.18 s)
└─ st.project.EmailsTest passed
    └─ st.project.EmailsTest.test passed (0.18 s)
```

The command line output shows:

```
=====
Command line suite
Total tests run: 1, Failures: 0, Skips: 0
=====
```

The status bar at the bottom right shows the date and time: 01-04-2023, 14:10, and the system language: ENG.

Test case id: T22

Input: priya.com

Expected output: invalid email

This screenshot is identical to the previous one, showing the same project structure, code editor, and test results. The difference is in the test input. In the code editor, the line of code that sets the email value to an empty string ('String output=test.email(" ");') has been changed to set it to 'priya.com' ('String output=test.email("priya.com");'). The 'Test Results' panel shows the test passed successfully with the same output:

```
Tests passed: 100.00%
The test passed. (0.194 s)
└─ st.project.EmailsTest passed
    └─ st.project.EmailsTest.test passed (0.194 s)
```

The command line output remains the same:

```
=====
Command line suite
Total tests run: 1, Failures: 0, Skips: 0
=====
```

The status bar at the bottom right shows the date and time: 01-04-2023, 16:49, and the system language: ENG.

Test case id: T23 (Failed Test case)

Input: dharshini@

Expected output: invalid email

The screenshot shows the Apache NetBeans IDE 15 interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and a search bar. The title bar displays "ST Project - Apache NetBeans IDE 15". The left sidebar shows project structure with packages like JAVA44, JAVASTS, and ST Project, containing classes such as DOB.java, Emails.java, Gender.java, Junittesting.java, Mobile.java, Names.java, and Security.java. A "Test Packages" section contains "st.project". Libraries include junit-4.13.2.jar and JDK 19 (Default). Test Libraries and WebApplication3 are also listed. The main workspace shows the code for "EmailsTest.java":

```
1 package st.project;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class EmailsTest {
7     @Test
8     public void test() {
9         Junittesting test=new Junittesting();
10        String output=test.email(“dharshini@”);
11        assertEquals(expected: “Success”, actual: output);
12    }
13 }
```

The "Test Results" panel shows the outcome of the test:

- Tests passed: 0.00 %
- No test passed, 1 test failed. (0.2 s)
- > ⚠️ st.project.EmailsTest Failed

The "Output - ST Project (test)" panel shows the stack trace:

```
at org.junit.Assert.assertEquals(Assert.java:117)
```

The system tray at the bottom right shows the date and time as 01-04-2023, 16:50.

Unit 7: Aadhaar Number

Test case id: T24

Input:

Expected output: aadhar number is required

The screenshot shows the Apache NetBeans IDE 15 interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and a search bar. The title bar displays "ST Project - Apache NetBeans IDE 15". The left sidebar shows project structure with packages like JAVA44, JAVASTS, and ST Project, containing classes such as Aadhaar.java, DOB.java, Emails.java, Gender.java, Junittesting.java, Mobile.java, Names.java, and Security.java. A "Test Packages" section contains "st.project". Libraries include junit-4.13.2.jar and JDK 19 (Default). Test Libraries and WebApplication3 are also listed. The main workspace shows the code for "AadhaarTest.java":

```
1 package st.project;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class AadhaarTest {
7     @Test
8     public void test() {
9         Junittesting test=new Junittesting();
10        String output=test.aadhar(“”);
11        assertEquals(expected: “aadhar number is required”, actual: output);
12    }
13 }
```

The "Test Results" panel shows the outcome of the test:

- Tests passed: 100.00 %
- The test passed. (0.213 s)
- > ✓ st.project.AadhaarTest passed
- > ✓ st.project.AadhaarTest.test passed (0.213 s)

The "Output - ST Project (test)" panel shows the stack trace:

```
at org.junit.Assert.assertEquals(Assert.java:117)
```

The system tray at the bottom right shows the date and time as 01-04-2023, 16:58.

Test case id: T25

Input: 12345678

Expected output: invalid aadhar must be 12 digit only

The screenshot shows the NetBeans IDE interface with the following details:

- Project Structure:** Shows a project named "st.project" containing source files like Aadhaar.java, DOB.java, Emails.java, Gender.java, Junittesting.java, Mobile.java, Names.java, and Security.java.
- Code Editor:** Displays the content of AadhaarTest.java:

```
1 package st.project;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class AadhaarTest {
7     @Test
8         public void test() {
9             Junittesting test=new Junittesting();
10            String output=test.aadhar("12345678");
11            assertEquals("invalid aadhar must be 12 digit only", output);
12        }
13    }
14}
15}
16}
```
- Test Results:** The "Test Results" panel shows "Tests passed: 100.00%" and a summary of the test run:

```
The test passed. (0.208 s)
st.project.AadhaarTest passed
  st.project.AadhaarTest.test passed (0.208 s)
```

Command line suite
Total tests run: 1, Failures: 0, Skips: 0
- System Tray:** Shows the date and time as 01-04-2023 17:04.

Test case id: T26(Failed Test case)

Input: 12345678

Expected output: invalid aadhar number

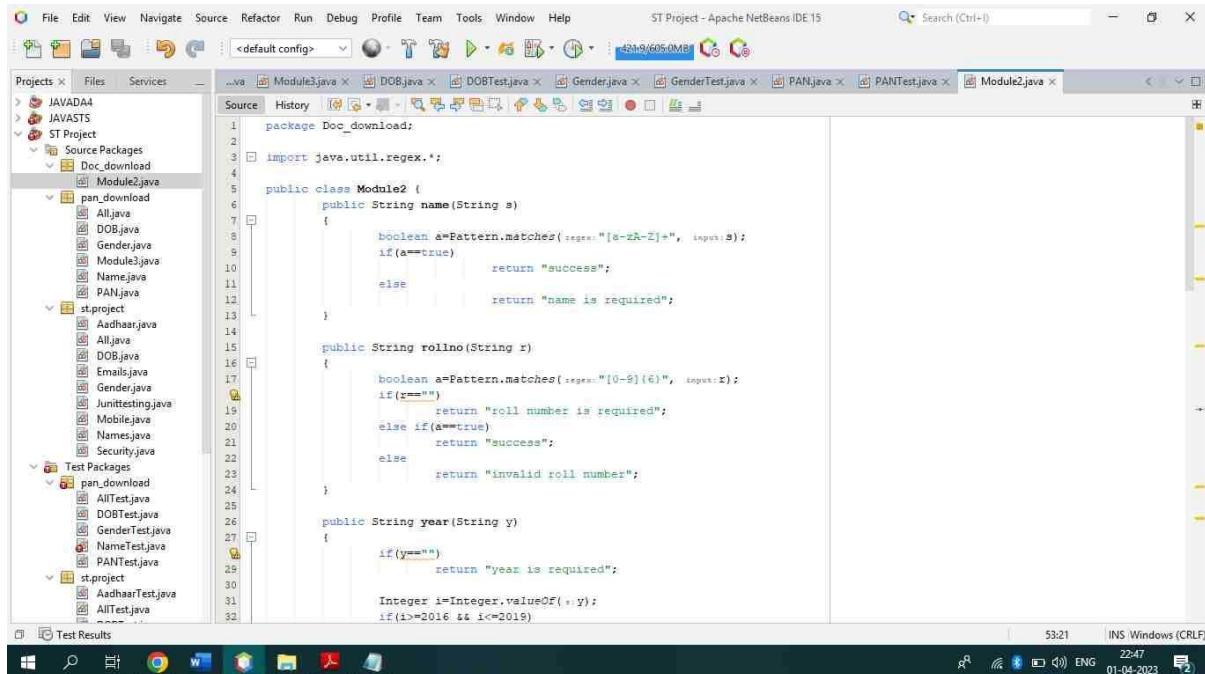
The screenshot shows the NetBeans IDE interface with the following details:

- Project Structure:** Same as the previous screenshot, showing the "st.project" directory.
- Code Editor:** Displays the same content of AadhaarTest.java as in the previous screenshot.
- Test Results:** The "Test Results" panel shows "Tests passed: 0.00%" and a summary of the test run:

```
No test passed, 1 test failed. (0.205 s)
st.project.AadhaarTest Failed
  st.project.AadhaarTest.test failed: org.junit.ComparisonFailure: expected:<[success]> but was:<[invalid aadhar must be 12 digit only]>
```

Command line suite
Total tests run: 1, Failures: 1, Skips: 0
- System Tray:** Shows the date and time as 01-04-2023 17:06.

MODULE 2:Marksheet Download



The screenshot shows the NetBeans IDE interface with the title bar "ST Project - Apache NetBeans IDE 15". The left sidebar displays the project structure under "Source Packages". The main editor window shows the code for "Module2.java". The code defines a class "Module2" with methods "name", "rollno", and "year". The "name" method checks if the input string is empty and returns "success" or "name is required". The "rollno" method checks if the input string is empty and returns "roll number is required" or "success". The "year" method checks if the input string is empty and returns "year is required". It also checks if the input is a valid integer between 2016 and 2019.

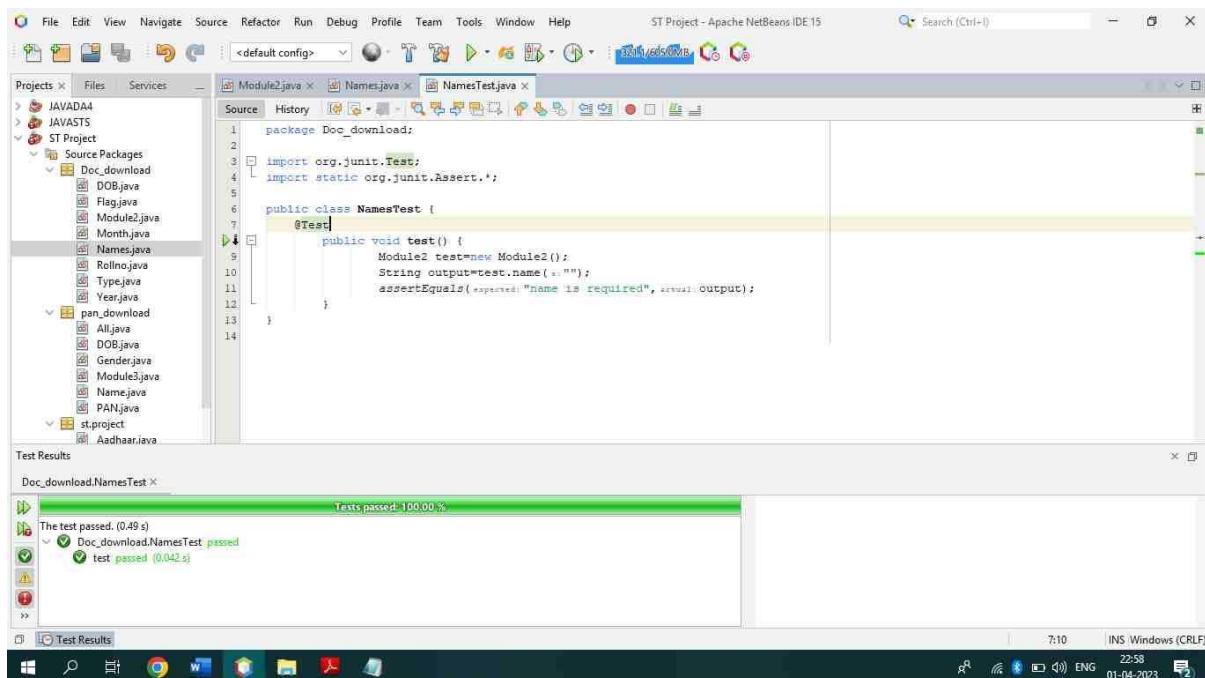
```
1 package Doc_download;
2
3 import java.util.regex.*;
4
5 public class Module2 {
6     public String name(String s)
7     {
8         boolean a=Pattern.matches("[a-zA-Z]+", s);
9         if(a==true)
10             return "success";
11         else
12             return "name is required";
13     }
14
15     public String rollno(String r)
16     {
17         boolean a=Pattern.matches("[0-9]{4}", r);
18         if(r=="")
19             return "roll number is required";
20         else if(a==true)
21             return "success";
22         else
23             return "invalid roll number";
24     }
25
26     public String year(String y)
27     {
28         if(y=="")
29             return "year is required";
30
31         Integer i=Integer.valueOf(+ y);
32         if(i>=2016 && i<=2019)
33             return "valid year";
34         else
35             return "invalid year";
36     }
37 }
```

Unit 1: Name

Test case id: TC1

Input:

Expected output: name is required



The screenshot shows the NetBeans IDE interface with the title bar "ST Project - Apache NetBeans IDE 15". The left sidebar displays the project structure under "Source Packages". The main editor window shows the code for "NamesTest.java", which contains a single test method "test" that creates an instance of "Module2", calls its "name" method with an empty string, and asserts that the result is "name is required". Below the editor, the "Test Results" window shows the test results for "Doc_download.NamesTest". It indicates 100.00% tests passed, with one test named "test" passing in 0.042 seconds.

```
1 package Doc_download;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class NamesTest {
7     @Test
8     public void test() {
9         Module2 test=new Module2();
10        String output=test.name("");
11        assertEquals("name is required", output);
12    }
13 }
```

Test Results

Doc_download.NamesTest

Tests passed: 100.00 %

The test passed. (0.49 s)

Doc_download.NamesTest: passed

test passed (0.042 s)

Unit 2: Roll No

Test case id: TC2

Input: 909345

Expected output: success

The screenshot shows the Apache NetBeans IDE interface. The top menu bar includes File, Edt, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and ST Project - Apache NetBeans IDE 15. The search bar at the top right contains the placeholder "Search (Ctrl+I)". The left sidebar displays the "Projects" view with several Java packages and files listed under "Source Packages" and "st.project". The "Test Results" section at the bottom shows a single test result for "Doc_download.RollnoTest" with a status of "passed" and a duration of "0.037 s". The main workspace shows the code editor for "RollnoTest.java" with the following content:

```
1 package Doc_download;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class RollnoTest {
7
8     @Test
9     public void test() {
10         Module2 test = new Module2();
11         String output = test.rollno("585345");
12         assertEquals("SUCCESS", output);
13     }
14 }
15
16 }
```

Test case id: TC3

Input:

Expected output: roll number is required

The screenshot shows the Apache NetBeans IDE interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and ST Project - Apache NetBeans IDE 15. The toolbar contains icons for file operations like New, Open, Save, Cut, Copy, Paste, Find, and Run. The left sidebar displays the Projects view with several source packages: JAVADA4, JAVASTS, ST Project, Doc_download, pan_download, and st.project. The Doc_download package is expanded, showing files like DOB.java, Flag.java, Module2.java, Month.java, Names.java, Rollno.java, Type.java, Year.java, All.java, DOB.java, Gender.java, Module3.java, Name.java, PAN.java, Aadhaar.java. The pan_download package is also expanded. The bottom-left corner shows the Test Results window for Doc_download.RollnoTest, which passed 100.00% of its tests. The main editor area shows the RollnoTest.java code:

```
1 package Doc_download;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class RollnoTest {
7
8     @Test
9     public void test() {
10         Module2 test = new Module2();
11         String output = test.rollno("");
12         assertEquals("roll number is required", output);
13     }
14 }
15
16 }
```

Test case id: TC4

Input: 9093459

Expected output: invalid roll number

The screenshot shows the Apache NetBeans IDE 15 interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and ST Project - Apache NetBeans IDE 15. The search bar says "Search (Ctrl+I)". The left sidebar displays the project structure under "ST Project" with "Source Packages" expanded, showing files like Doc_download, Module2, and pan_download. The main editor window shows the code for RollnoTest.java:

```
1 package Doc_download;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class RollnoTest {
7
8     @Test
9     public void test() {
10         Module2 test = new Module2();
11         String output = test.rollno("9093459");
12         assertEquals("invalid roll number", output);
13     }
14 }
15
16 }
```

The "Test Results" panel at the bottom shows the output of the test:

```
Doc_download.RollnoTest
The test passed. (0.593 s)
  ✓ Doc_download.RollnoTest passed
    ✓ test passed (0.02 s)
```

The status bar at the bottom right shows the date and time: 01-04-2023, 23:10, ENG, and INS Windows (CRLF).

Unit 3: Year

Test case id: TC5

Input:

Expected output: year is required

The screenshot shows the Apache NetBeans IDE 15 interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and ST Project - Apache NetBeans IDE 15. The search bar says "Search (Ctrl+I)". The left sidebar displays the project structure under "ST Project" with "Source Packages" expanded, showing files like Doc_download, Module2, and pan_download. The main editor window shows the code for YearTest.java:

```
1 package Doc_download;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class YearTest {
7
8     @Test
9     public void test() {
10         Module2 test=new Module2();
11         String output=test.year("");
12         assertEquals("year is required", output);
13     }
14 }
15
16 }
```

The "Test Results" panel at the bottom shows the output of the test:

```
Doc_download.YearTest
The test passed. (0.574 s)
  ✓ Doc_download.YearTest passed
    ✓ test passed (0.014 s)
```

The status bar at the bottom right shows the date and time: 01-04-2023, 23:16, ENG, and INS Windows (CRLF).

Test case id: TC6

Input: 2016

Expected output: success

The screenshot shows the NetBeans IDE interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and a search bar. Below the menu is a toolbar with various icons. The left sidebar displays the project structure under 'ST Project' with packages like 'Doc_download' and 'pan_download'. The main editor window shows the code for 'YearTest.java':

```
1 package Doc_download;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class YearTest {
7     @Test
8     public void test() {
9         Module2 test=new Module2();
10        String output=test.year("2016");
11        assertEquals("success", actual:output);
12    }
13 }
14
15
16
17
```

Below the editor is a 'Test Results' window for 'Doc_download.YearTest'. It shows a green bar indicating 'Tests passed: 100.00 %'. Inside the bar, it says 'The test passed. (0.483 s)' and lists a single test named 'test' with status 'passed (0.024 s)'. The bottom right corner of the screen shows system status icons.

Test case id: TC7

Input: 2013

Expected output: 2016-2019 documents only available

The screenshot shows the NetBeans IDE interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and a search bar. Below the menu is a toolbar with various icons. The left sidebar displays the project structure under 'ST Project' with packages like 'Doc_download' and 'pan_download'. The main editor window shows the code for 'YearTest.java':

```
1 package Doc_download;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class YearTest {
7     @Test
8     public void test() {
9         Module2 test=new Module2();
10        String output=test.year("2013");
11        assertEquals("2016-2019 documents only available", actual:output);
12    }
13 }
14
15
16
17
```

Below the editor is a 'Test Results' window for 'Doc_download.YearTest'. It shows a green bar indicating 'Tests passed: 100.00 %'. Inside the bar, it says 'The test passed. (0.498 s)' and lists a single test named 'test' with status 'passed (0.027 s)'. A note 'No output.' is displayed below the test results. The bottom right corner of the screen shows system status icons.

Unit 4: DOB

Test case id: TC8

Input:

Expected output: DOB is required

The screenshot shows the Apache NetBeans IDE 15 interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and ST Project - Apache NetBeans IDE 15. The search bar at the top right contains the text "Search (Ctrl+I)". The left sidebar displays the project structure under "Source Packages". The "Doc_download" package contains several files: DateofBirth.java, Flag.java, Module2.java, Month.java, Names.java, Rollno.java, Type.java, and Year.java. The "pan_download" package contains All.java, DOB.java, Gender.java, Module3.java, Name.java, PAN.java, and Aadhaar.java. The "st.project" package contains Aadhaar.java. The main editor window shows the code for DateofBirthTest.java:

```
1 package Doc_download;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class DateofBirthTest {
7     @Test
8     public void test() {
9         Module2 test=new Module2();
10        String output=test.dob("");
11        assertEquals("expected","DOB is required",actual,output);
12    }
13}
```

The "Test Results" panel below the editor shows the execution of the test. It indicates "Tests passed: 100.00 %". The test result for "Doc_download.DateofBirthTest" is listed as "passed" with a duration of "0.023 s". A note says "No output." The status bar at the bottom right shows the time as 01-04-2023 23:22.

Unit 5: Certificate Type

Test case id: TC8

Input: regular

Expected output: success

The screenshot shows the Apache NetBeans IDE 15 interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and ST Project - Apache NetBeans IDE 15. The search bar at the top right contains the text "Search (Ctrl+I)". The left sidebar displays the project structure under "Source Packages". The "Doc_download" package contains several files: DateofBirth.java, Flag.java, Module2.java, Month.java, Names.java, Rollno.java, Type.java, and Year.java. The "pan_download" package contains All.java, DOB.java, Gender.java, Module3.java, Name.java, PAN.java, and Aadhaar.java. The "st.project" package contains Aadhaar.java. The main editor window shows the code for TypeTest.java:

```
1 package Doc_download;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class TypeTest {
7     @Test
8     public void test() {
9         Module2 test = new Module2();
10        String output = test.type("REGULAR");
11        assertEquals("expected","Success", actual, output);
12    }
13}
```

The "Test Results" panel below the editor shows the execution of the test. It indicates "Tests passed: 100.00 %". The test result for "Doc_download.TypeTest" is listed as "passed" with a duration of "0.043 s". The status bar at the bottom right shows the time as 01-04-2023 23:24.

Test case id: TC9

Input:

Expected output: select certificate type

The screenshot shows the Apache NetBeans IDE interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and a search bar. The title bar indicates "ST Project - Apache NetBeans IDE 15". The left sidebar displays the project structure under "Source Packages" for the "Doc_download" module, listing files like DateBirth.java, Flag.java, Module2.java, Month.java, Names.java, Rollno.java, Type.java, Year.java, and others. The main editor window shows the code for "TypeTest.java":

```
1 package Doc_download;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class TypeTest {
7
8     @Test
9     public void test() {
10         Module2 test = new Module2();
11         String output = test.type("");
12         assertEquals("Select certificate type", actual.output);
13     }
14
15 }
```

The code contains a JUnit test method "test" that creates an instance of "Module2", calls its "type" method with an empty string, and then uses an assertEquals assertion to check if the output is "Select certificate type".

Below the editor is the "Test Results" window for "Doc_download.TypeTest". It shows a green progress bar at 100.00% and a list of test results:

- The test passed. (0.591 s)
- Doc_download.TypeTest passed
- test passed (0.021 s)

A tooltip message "Tests (1) finished successfully for project: ST Project" is displayed above the status bar. The status bar also shows the date and time: 01-04-2023, 12:46, and the system language: ENG.

Unit 6: Flag

Test case id: TC10

Input:

Expected output: select flag

The screenshot shows the Apache NetBeans IDE interface, identical to the previous one but with a different test result. The title bar indicates "ST Project - Apache NetBeans IDE 15". The left sidebar displays the project structure under "Source Packages" for the "Doc_download" module, listing files like DateBirth.java, Flag.java, Module2.java, Month.java, Names.java, Rollno.java, Type.java, Year.java, and others. The main editor window shows the code for "FlagTest.java":

```
1 package Doc_download;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class FlagTest {
7
8     @Test
9     public void test() {
10         Module2 test = new Module2();
11         String output = test.flag("");
12         assertEquals("Select flag", actual.output);
13     }
14
15 }
```

The code contains a JUnit test method "test" that creates an instance of "Module2", calls its "flag" method with an empty string, and then uses an assertEquals assertion to check if the output is "Select flag".

Below the editor is the "Test Results" window for "Doc_download.FlagTest". It shows a green progress bar at 100.00% and a list of test results:

- The test passed. (0.881 s)
- Doc_download.FlagTest passed
- test passed (0.03 s)

A tooltip message "Tests (1) finished successfully for project: ST Project" is displayed above the status bar. The status bar also shows the date and time: 01-04-2023, 13:06, and the system language: ENG.

Test case id: TC11

Input:

Expected output: success

The screenshot shows the Apache NetBeans IDE 15 interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and a search bar. The title bar says "ST Project - Apache NetBeans IDE 15". The left sidebar displays the project structure under "Source Packages", showing files like DateofBirth.java, Module2.java, Month.java, Names.java, Rollno.java, Type.java, Year.java, and others. The main editor window contains the code for FlagTest.java:

```
1 package Doc_download;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class FlagTest {
7
8     @Test
9     public void test() {
10         Module2 test = new Module2();
11         String output = test.flag("X");
12         assertEquals("Success", output);
13     }
14
15 }
```

Below the editor is the "Test Results" panel, which shows the output for Doc_download.FlagTest:

```
The test passed. (0.536 s)
  ✓ Doc_download.FlagTest passed
    ✓ test passed (0.022 s)
```

The status bar at the bottom right indicates the date as 01-04-2023 and the time as 12:30.

Unit 7: Month

Test case id: TC12

Input: JAN

Expected output: success

The screenshot shows the Apache NetBeans IDE 15 interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and a search bar. The title bar says "ST Project - Apache NetBeans IDE 15". The left sidebar displays the project structure under "Source Packages", showing files like DateofBirth.java, Module2.java, Month.java, Names.java, Rollno.java, Type.java, Year.java, and others. The main editor window contains the code for MonthTest.java:

```
1 package Doc_download;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class MonthTest {
7
8     @Test
9     public void test() {
10         Module2 test = new Module2();
11         String output = test.month("JAN");
12         assertEquals("Success", output);
13     }
14
15 }
```

Below the editor is the "Test Results" panel, which shows the output for Doc_download.MonthTest:

```
The test passed. (0.553 s)
  ✓ Doc_download.MonthTest passed
    ✓ test passed (0.021 s)
```

The status bar at the bottom right indicates the date as 01-04-2023 and the time as 16:1.

Test case id: TC13

Input:

Expected output: month is required

The screenshot shows the NetBeans IDE interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and a search bar. The title bar indicates "ST Project - Apache NetBeans IDE 15". The left sidebar displays the project structure under "Source Packages", showing files like DateofBirth.java, Flag.java, Module2.java, Month.java, Names.java, Rollno.java, Type.java, Year.java, and others. The main editor window shows the code for MonthTest.java:

```
1 package Doc_download;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class MonthTest {
7
8     @Test
9     public void test() {
10         Module2 test = new Module2();
11         String output = test.month("");
12         assertEquals("month is required", output);
13     }
14 }
15
16 }
```

The "Test Results" window below the editor shows a green bar indicating "Tests passed: 100.00 %". It lists a single test: "Doc_download.MonthTest passed" with a sub-item "test passed (0.003 s)". The status bar at the bottom right shows the date and time as 01-04-2023 23:34.

Test case id: TC14

Input: JHK

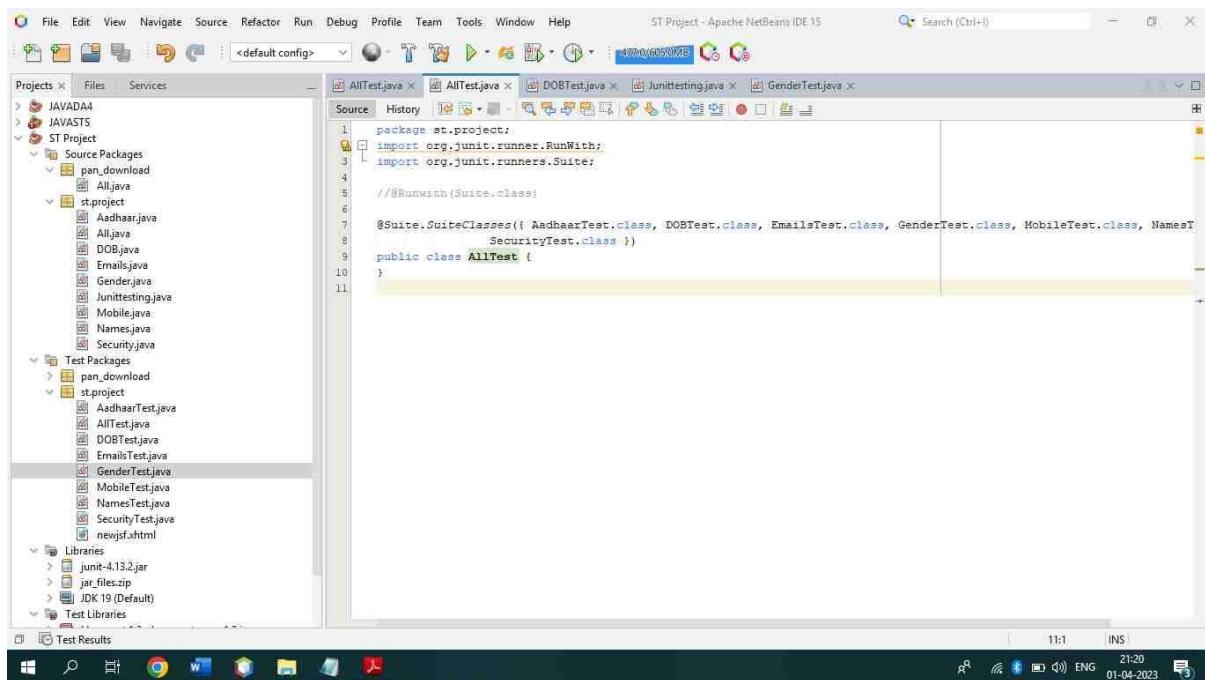
Expected output: invalid month

The screenshot shows the NetBeans IDE interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and a search bar. The title bar indicates "ST Project - Apache NetBeans IDE 15". The left sidebar displays the project structure under "Source Packages", showing files like DateofBirth.java, Flag.java, Module2.java, Month.java, Names.java, Rollno.java, Type.java, Year.java, and others. The main editor window shows the code for MonthTest.java:

```
1 package Doc_download;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class MonthTest {
7
8     @Test
9     public void test() {
10         Module2 test = new Module2();
11         String output = test.month("JHK");
12         assertEquals("invalid month", output);
13     }
14 }
15
16 }
```

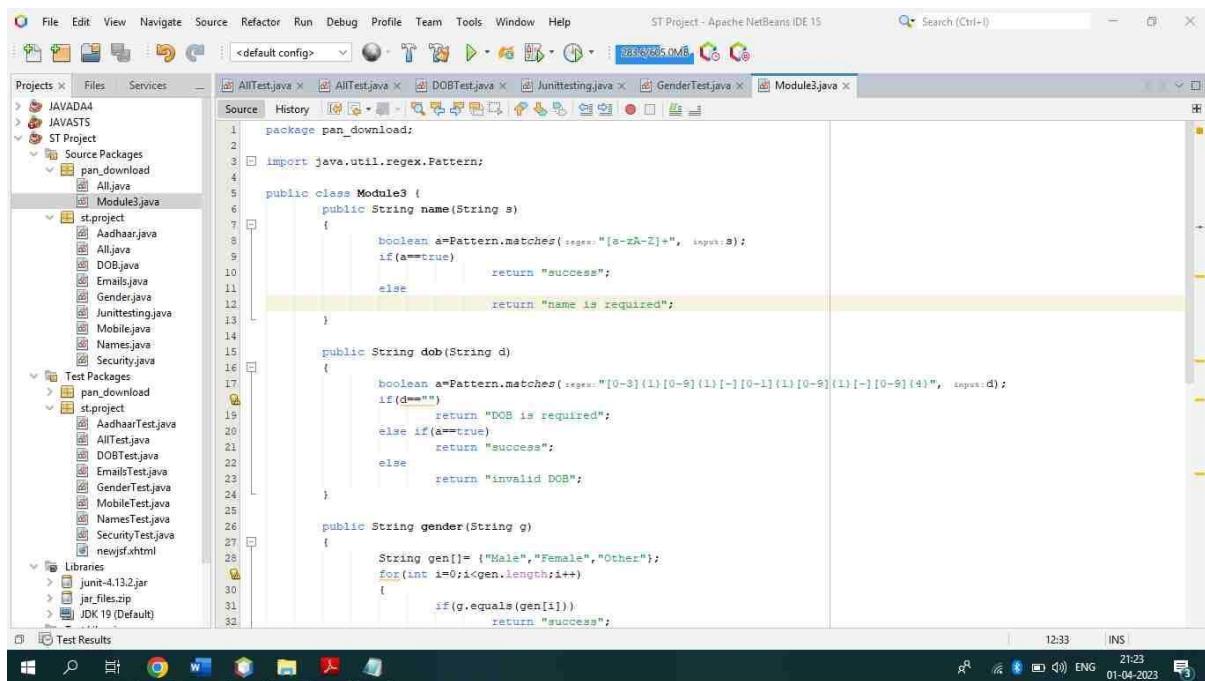
The "Test Results" window below the editor shows a green bar indicating "Tests passed: 100.00 %". It lists a single test: "Doc_download.MonthTest passed" with a sub-item "test passed (0.03 s)". A tooltip message "Tests finished successfully for project: ST Project" appears above the status bar. The status bar at the bottom right shows the date and time as 01-04-2023 23:37.

MODULE 3:PAN card DOWNLOAD



The screenshot shows the Apache NetBeans IDE 15 interface. The left sidebar displays the project structure under 'ST Project'. The 'Source Packages' section contains a package named 'pan_download' which includes files like All.java, Aadhaar.java, All.java, DOB.java, Emails.java, Gender.java, Junittesting.java, Mobile.java, Names.java, and Security.java. Below it is a 'Test Packages' section with 'pan_download' and 'st.project' containing various test classes. The right panel shows the code editor for 'AllTest.java'. The code defines a class 'AllTest' with a constructor annotated with '@RunWith(Suite.class)' and '@Suite.SuiteClasses({ AadhaarTest.class, DOBTest.class, EmailsTest.class, GenderTest.class, MobileTest.class, NamesTest.class, SecurityTest.class })'. The code editor has a status bar at the bottom showing '11:1 INS'.

```
1 package st.projects;
2 import org.junit.runner.RunWith;
3 import org.junit.runners.Suite;
4
5 //RunWith(Suite.class)
6
7 @Suite.SuiteClasses({ AadhaarTest.class, DOBTest.class, EmailsTest.class, GenderTest.class, MobileTest.class, NamesTest.class, SecurityTest.class })
8 public class AllTest {
9 }
10
11
```



The screenshot shows the Apache NetBeans IDE 15 interface. The left sidebar displays the project structure under 'ST Project'. The 'Source Packages' section contains a package named 'pan_download' which includes files like All.java, Module3.java, Aadhaar.java, All.java, DOB.java, Emails.java, Gender.java, Junittesting.java, Mobile.java, Names.java, Security.java, and newjsf.xhtml. Below it is a 'Test Packages' section with 'pan_download' and 'st.project' containing various test classes. The right panel shows the code editor for 'Module3.java'. The code defines a class 'Module3' with three methods: 'name', 'dob', and 'gender'. The 'name' method checks if the input string starts with a-z or A-Z. The 'dob' method checks if the input string matches the pattern [0-3](1)[0-9](1)[-][0-1](1)[0-9](1)[-][0-9](4). The 'gender' method compares the input string with an array of gender values. The code editor has a status bar at the bottom showing '12:33 INS'.

```
1 package pan_download;
2
3 import java.util.regex.Pattern;
4
5 public class Module3 {
6     public String name(String s)
7     {
8         boolean a=Pattern.matches("[a-zA-Z]+", s);
9         if(a==true)
10             return "success";
11         else
12             return "name is required";
13     }
14
15     public String dob(String d)
16     {
17         boolean a=Pattern.matches("[0-3](1)[0-9](1)[-][0-1](1)[0-9](1)[-][0-9](4)", d);
18         if(d=="")
19             return "DOB is required";
20         else if(a==true)
21             return "success";
22         else
23             return "invalid DOB";
24     }
25
26     public String gender(String g)
27     {
28         String gen[]={"Male","Female","Other"};
29         for(int i=0;i<gen.length;i++)
30         {
31             if(g.equals(gen[i]))
32                 return "success";
33         }
34     }
35 }
```

The screenshot shows the Apache NetBeans IDE interface with the following details:

- File Menu:** File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help.
- Project Explorer:** Shows a project named "ST Project" containing packages like "Source Packages" (with "pan_download" and "st.project"), "Text Packages" (with "pan_download" and "st.project"), and "Libraries" (with "junit-4.13.2.jar", "jar_files.zip", and "JDK 15 (Default)").
- Code Editor:** The main window displays Java code for validating PAN and DOB numbers. The code includes methods for gender validation and PAN validation, returning success or failure messages based on input parameters.
- Status Bar:** Shows "ST Project - Apache NetBeans (IDE 15)" and the current time "21:24 01-04-2023".

```
15     return "DOB is required";
16   else if(a==true)
17     return "success";
18   else
19     return "invalid DOB";
20
21
22
23
24 }
25
26
27 public String gender(String g)
28 {
29   String gen[] = {"Male","Female","Other"};
30   for(int i=0;i<gen.length;++)
31   {
32     if(g.equals(gen[i]))
33       return "success";
34   }
35   if(g=="")
36     return "gender is required";
37   else
38     return "";
39
40
41 public String pan(String p)
42 {
43   boolean a=Pattern.matches(regex:"[A-Z]{5}[0-9]{4}[A-Z]{1}", input:p);
44   if(p=="")
45     return "PAN number is required";
46   else if(a==true)
47     return "success";
48   else
49     return "invalid PAN number";
50 }
```

Unit 1: Name

Test case id: T01

Input: 123

Expected output: “name is required”

The screenshot shows the Apache NetBeans IDE interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and ST Project - Apache NetBeans IDE 15. The toolbar contains icons for file operations like Open, Save, and Print. The left sidebar displays the project structure under 'Projects X Files Services'. It includes Source Packages (pan_download, Alljava, Module3.java, Name.java), Test Packages (st.project, Aadhaar.java, Alljava, DDB.java, Emails.java, Gender.java, Junittesting.java, Mobile.java, Names.java, Security.java), and Test Results (pan_download.NameTest, st.project.AllTest). The main editor window shows Java code for 'NameTest.java':

```
package pan_download;
import org.junit.Test;
import static org.junit.Assert.*;
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;

public class NameTest {

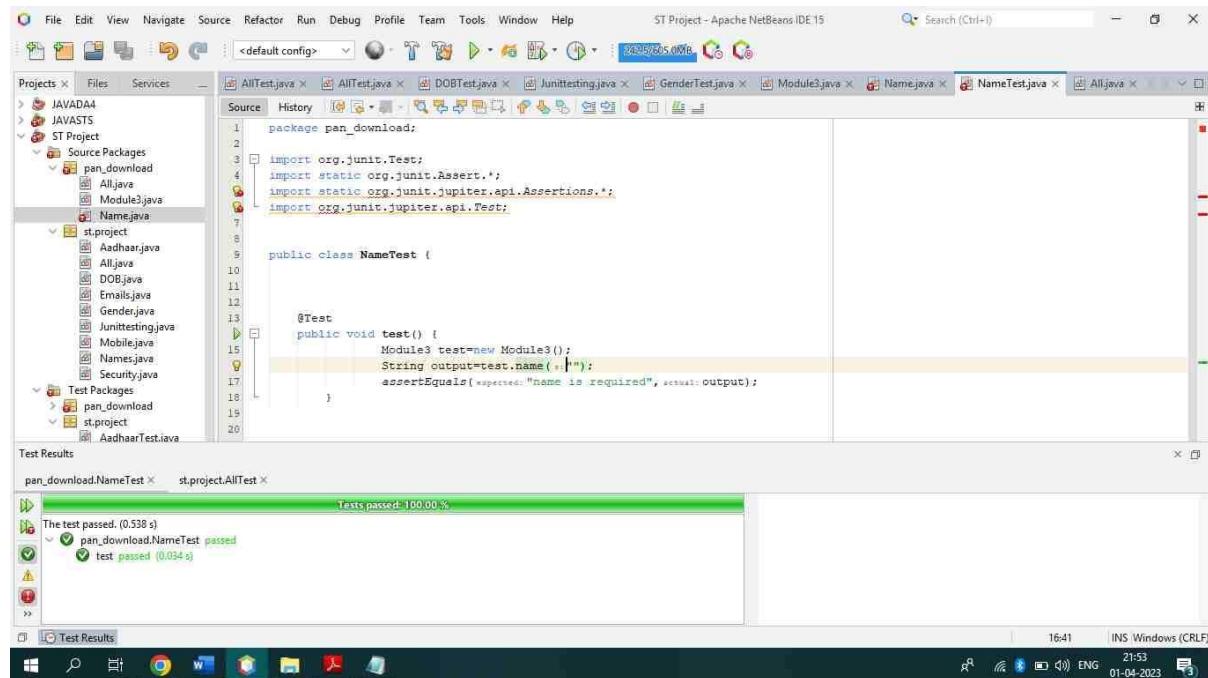
    @Test
    public void test() {
        Module3 test=new Module3();
        String output=test.name("1101");
        assertEquals("name is required", output);
    }
}
```

The bottom right corner shows the system tray with icons for network, battery, volume, and date/time (21:2, ENG 01-04-2023, INS Windows (CRLF)).

Test case id: T02

Input:

Expected output: name is required.



```
1 package pan_download;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5 import static org.junit.jupiter.api.Assertions.*;
6 import org.junit.jupiter.api.Test;
7
8
9 public class NameTest {
10
11     @Test
12     public void test() {
13         Module3 test=new Module3();
14         String output=test.name+"";
15         assertEquals("name is required", actual:output);
16     }
17 }
18
19
20
```

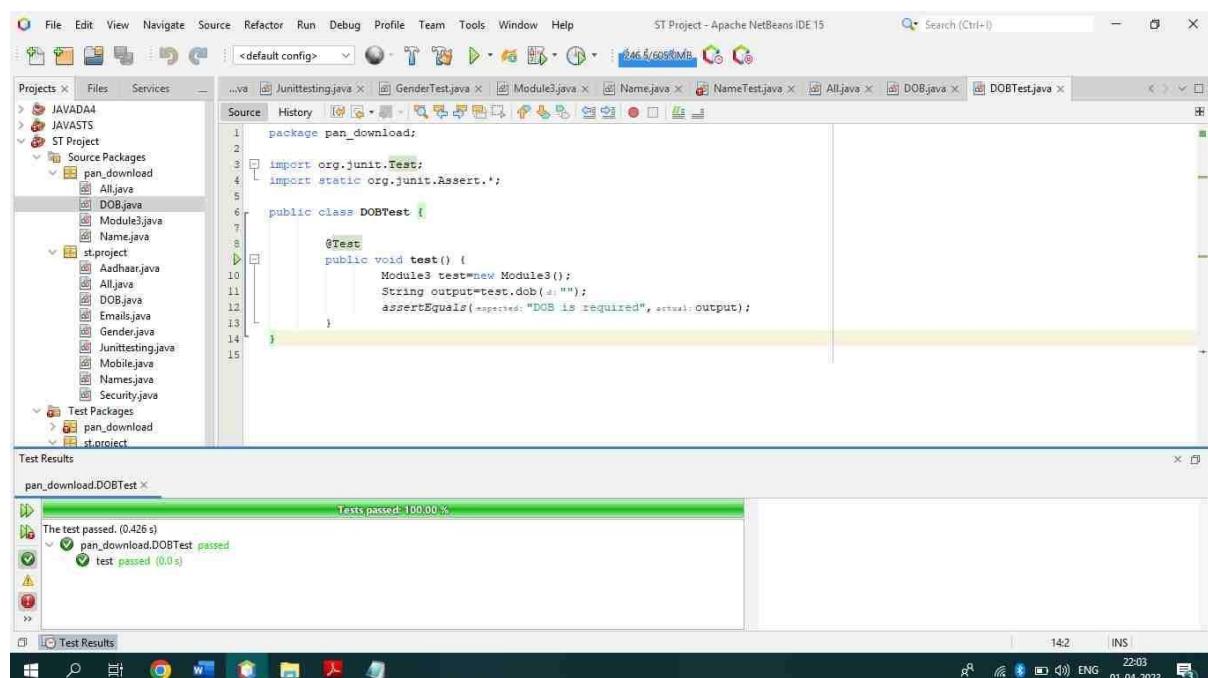
The test passed. (0.538 s)
pan_download.NameTest passed
test passed (0.034 s)

Unit 2: DOB

Test case id: T03

Input:

Expected output: DOB is required



```
1 package pan_download;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class DOBTest {
7
8     @Test
9     public void test() {
10         Module3 test=new Module3();
11         String output=test.dob+"";
12         assertEquals("DOB is required", actual:output);
13     }
14 }
15
```

The test passed. (0.426 s)
pan_download.DOBTest passed
test passed (0.0 s)

Unit 3: Gender

Test case id: T03

Input:

Expected output: gender is required.

The screenshot shows the NetBeans IDE interface with the following details:

- Project Structure:** Shows a project named "ST Project" with Source Packages containing "pan_download" and "Gender.java".
- Code Editor:** Displays the content of `GenderTest.java`:

```
1 package pan_download;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class GenderTest {
7     @Test
8     public void test() {
9         Module3 test = new Module3();
10        String output=test.gender("");
11        assertEquals("gender is required", output);
12    }
13}
```
- Test Results:** Shows the results for `pan_download.GenderTest`. It indicates 100.00% tests passed, with one test named "test" passing in 0.03 seconds.

Unit 4: PAN number

Test case id: T04

Input: PDA414

Expected output: invalid PAN number

The screenshot shows the NetBeans IDE interface with the following details:

- Project Structure:** Shows a project named "ST Project" with Source Packages containing "pan_download" and "PAN.java".
- Code Editor:** Displays the content of `PANTest.java`:

```
1 package pan_download;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class PANTest {
7     @Test
8     public void test() {
9         Module3 test = new Module3();
10        String output=test.pan("PDA414");
11        assertEquals("invalid PAN number", output);
12    }
13}
```
- Test Results:** Shows the results for `pan_download.PANTest`. It indicates 100.00% tests passed, with one test named "test" passing in 0.016 seconds.

Test case id: T05

Input:

Expected output: PAN number is required

The screenshot shows the Apache NetBeans IDE 15 interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and a search bar. Below the menu is a toolbar with various icons. The left sidebar displays a project tree under 'ST Project' with packages like 'pan_download' and 'st.project'. The main editor window shows Java code for a 'PANTest' class. The code defines a single test method 'test' that creates an instance of 'Module3', calls its 'pan' method with an empty string, and then uses 'assertEquals' to check if the output matches the expected value 'PAN number is required'. The bottom panel shows a 'Test Results' window for 'pan_download.PANTest'. It displays a green progress bar indicating 'Texts passed: 100.00 %'. Below the bar, it shows a single test named 'pan_download.PANTest passed' with a sub-test 'test passed (0.028 s)' marked with a green checkmark.

```
1 package pan_download;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class PANTest {
7
8     @Test
9     public void test() {
10         Module3 test = new Module3();
11         String output=test.pan("");
12         assertEquals(expected:"PAN number is required",actual:output);
13     }
14 }
15
16 }
```

Test case id: T06

Input: PDA414LKJII

Expected output: invalid PAN number

This screenshot is identical to the one above, showing the same version of the Java code for the 'PANTest' class. The code is identical, except for the input parameter in the 'pan' method call, which is now set to "PDA414LKJII". The 'Test Results' window at the bottom shows the same successful test run with a 100.00% pass rate and the same test name and duration.

```
1 package pan_download;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class PANTest {
7
8     @Test
9     public void test() {
10         Module3 test = new Module3();
11         String output=test.pan("PDA414LKJII");
12         assertEquals(expected:"invalid PAN number",actual:output);
13     }
14 }
15
16 }
```

Conclusion :

We have created Test Case report and Here we successfully tested Digilocker website with the help of Junit testing tool and netbeans. We successfully added some bugs to code and did testing with junit tool.