**CONTINUOUS IMPROVEMENT PROCESS IMPLEMENATION FOR PERSHING SECURITY FRAMEWORK**

A PROJECT REPORT

*Submitted by*

**Saranya.P (111712104089)**

**Vaishni.R (111712104113)**

***In partial fulfilment for the award of the degree***

*Of*

BACHELOR OF ENGINEERING

IN

**COMPUTER SCIENCE AND ENGINEERING**

RMK ENGINEERING COLLEGE, KAVARAIPETTAI

ANNA UNIVERSITY :: CHENNAI 600 025

APRIL 2016

**ANNA UNIVERSITY::CHENNAI – 600 025**

**BONAFIDE CERTIFICATE**

Certified that this project report “**CONTINUOUS IMPROVEMENT PROCESS IMPLEMENTATION FOR PERSHING SECURITY FRAMEWORK**” is the bonafide work of  **SARANYA[111712104089] , VAISHNI[111712104113]** who carried out the project under my supervision.

|  |  |
| --- | --- |
| **SIGNATURE**  **DR. K.L.SHANMUGANATHAN, M.E.,M.S.,Ph.D.,** HEAD OF THE DEPARTMENT  PROFESSOR  Department of Computer Science and  Engineering,  R.M.K Engineering College,  Kavaraipettai - 601 206. | **SIGNATURE**  **Mrs. C.GEETHA, M.E., (Ph. D)**  SUPERVISOR  ASSOCIATE PROFESSOR  Department of Computer Science and Engineering,  R.M.K Engineering College,  Kavaraipettai - 601 206. |

Submitted to Project Viva Voce Examination held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| **EXTERNAL EXAMINER** | **INTERNAL EXAMINER** |

**ABSTRACT**

More and more enterprises and organizations are hosting their data into the cloud, in order to reduce the IT maintenance cost and enhance the data reliability. However, facing the numerous cloud vendors as well as their heterogeneous pricing policies, customers may well be perplexed with which cloud(s) are suitable for storing their data and what hosting strategy is cheaper. The general status quo is that customers usually put their data into a single cloud (which is subject to the vendor lock-in risk) and then simply trust to luck. Based on comprehensive analysis of various state-of-the-art cloud vendors, this paper proposes a novel data hosting scheme (named CHARM) which integrates two key functions desired. The first is selecting several suitable clouds and an appropriate redundancy strategy to store data with minimized monetary cost and guaranteed availability. The second is triggering a transition process to re-distribute data according to the variations of data access pattern and pricing of clouds. We evaluate the performance of CHARM using both trace-driven simulations and prototype experiments. The results show that compared with the major existing schemes, CHARM not only saves around 20% of monetary cost but also exhibits sound adaptability to data and price adjustments.

**ACKNOWLEDGEMENT**

Support on demand, encouragement at the needed moment and guidance in the right direction are in dispensable for the success of any project. We have received these in excess from all corners from various people, we are glad to submit our gratitude to them.

We thank **Shri. R.S.Munirathinam, Chairman,** and **Shri. R.M.Kishore , Vice Chairman** of RMK group of Institutions for extending a generous hand in providing the best of resources to the college. Dr**. Elwin Chandra Monnie, the esteemed Head of our Institution** has been a source of motivation to all the staffs and students of our college. We are so much thankful to him. Our sincere thanks to **Dr. Mohammed Junaid, the Vice Principal** for giving the continuous support for the completion of our project.

Our sincere thanks to **Dr. K.L.Shanmuganathan, the Head of the Department** for his continuous support and motivation throughout our project.

We extend our profound gratitude to **Dr. K.Sethukarasi our Project Coordinator** and **Mrs. C.GEETHA our Guide** for her guidance, who has indeed been a polestar throughout the course of the project, we thank him for giving us full support to complete the project successfully.

Last, but not the least, we take this opportunity to thank all the **staff members** of the Department of Computer Science and Engineering. Regards to **our family, classmates and friends** who offered an unflinching moral support for completion of this project.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT** | **iii** |
|  | **LIST OF FIGURES** | **ix** |
|  | **LIST OF ABBREVIATIONS** | **x** |
| **1.** | **INTRODUCTION** |  |
|  | 1.1 Objective of the Project |  |
|  | 1.2 System Study |  |
|  | 1.2.1 Existing System  1.2.2 Proposed System |  |
| **2.** | **SYSTEM ANALYSIS** |  |
|  | 2.1 Scope  2.2 Purpose  2.3 Resources Required  2.4 Feasibility Study |  |
|  | 2.4.1 Technical Feasibility |  |
|  | 2.4.2 Economic Feasibility  2.4.3 Operational Feasibility |  |
|  | 2.5 System Specification |  |
|  | 2.5.1 Hardware Specification |  |
|  | 2.5.2 Software Specification |  |
| **3.** | **SYSTEM DESIGN** |  |
|  | 3.1 System Architecture |  |
|  | 3.2 Use Case Diagram |  |
|  | 3.3 Activity Diagram |  |
|  | 3.4 Sequence Diagram |  |
|  | 3.5 Collaboration Diagram |  |
| **4.** | **TECHNOLOGIES USED** |  |
|  | 4.1 Javascript  4.2 Maven  4.3 Junit  4.4 Java7/J2EE  4.5 Eclipse |  |
| **5.** | **PROJECT DESCRIPTION**  5.1 Modules Description |  |
| **6.** | **TESTING OF THE PROJECT**  6.1 Testing  6.2 Unit Testing  6.3 Running unit test with Maven |  |
| **7.** | **IMPLEMENTATION AND MAINTENANCE**  7.1 Implementation  7.2 Maintenance |  |
| **8.** | **CONCLUSION**  8.1 Appendix(Screenshots)  8.2 References |  |
|  |  |  |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **TITLE** | **PAGE NO.** |
| 3.1 | System Architecture |  |
| 3.2 | Use Case Diagram |  |
| 3.3 | Activity Diagram |  |
| 3.4 | Sequence Diagram |  |
| 3.5 | Collaboration Diagram |  |
| 5.1 | Security authentication |  |

**LIST OF ABBREVIATIONS**

|  |  |  |
| --- | --- | --- |
| 1. | SDLC | Software Development Life Cycle |
| 2. | PSF | Pershing Security Framework |
| 3. | API | Application Program Interface |
| 4. | JDBC | Java Database Connectivity |
| 5. | IDE | Integrated Development Environment |
| 6. | POM | Project Object Model |
| 7. | XML | Extensible Markup Language |
| 8. | JSP | Java server page |
| 9. | EJB | Enterprise JavaBeans |
| 10. | GUI | Graphical User Interface |
|  |  |  |

**CHAPTER 1**

**INTRODUCTION**

**1.1 OBJECTIVE OF THE PROJECT**

Computer security is the protection of information systems from theft or damage to the hardware, the software, and to the information on them, as well as from disruption or misdirection of the services they provide. It includes controlling physical access to the hardware, as well as protecting against harm that may come via network access, data and code injection and due to malpractice by operators, whether intentional, accidental, or due to them being tricked into deviating from secure procedures.

The main objective of the project is to develop a sample application that usespershingsecurity framework by using Maven and developingjunit modules for the Pershing Security Framework which can provide security to a greater extent.

* 1. **SYSTEM STUDY**

The system study phase involves the understanding of the Pershing security framework which provides authentication and authorization to users .The security framework provides different types of access to users based on their roles assigned .

**1.2.1 EXISTING SYSTEM**

Authentication and authorization framework for intelligently controlling [access](http://whatis.techtarget.com/definition/access) to computer resources, enforcing policies, auditing usage, and providing the information necessary for services. These combined processes are considered important for effective network management and security.As the first process, [authentication](http://searchsecurity.techtarget.com/definition/authentication) provides a way of identifying a user, typically by having the user enter a valid user name and valid password before access is granted. The process of authentication is based on each user having a unique set of criteria for gaining access. The server compares a user's authentication credentials with other user credentials stored in a database. If the credentials match, the user is granted access to the network. If the credentials are at variance, authentication fails and network access is denied.

Following authentication, a user must gain [authorization](http://searchsoftwarequality.techtarget.com/definition/authorization) for doing certain tasks. After logging into a system, for instance, the user may try to issue commands. The authorization process determines whether the user has the authority to issue such commands. Simply put, authorization is the process of enforcing policies: determining what types or qualities of activities, resources, or services a user is permitted. Usually, authorization occurs within the context of authentication. Once you have authenticated a user, they may be authorized for different types of access or activity.This can include the amount of system time or the amount of data a user has sent and/or received during a session. Accounting is carried out by logging of session statistics and usage information and is used for authorization control, billing, trend analysis, resource utilization, and capacity planning activities.

1.2.2**PROPOSED SYSTEM**

By understanding the current authentication & authorization framework, we would need to develop an application that is going to consume this framework. Having that working, we would need to come up with the junit modules for different use cases supported by the pershing security framework. This will then be automated in such a way that these modules will be invoked during build time and will check for any gaps programmatically. Hence avoiding any human intervention for validation.

**CHAPTER 2**

**SYSTEM ANALYSIS**

Systems are created to solve problems. One can think of the systems approach as an organized way of dealing with a problem.The analysis phase is the second phase of the SDLC and to describe if the project should go ahead with the resources available.This also includes looking at any existing system to see what it is doing for the organization and how well that system is doing its job .The feasibility of the project is also considered and the group has to ask questions such as

* Can this system be created with the resources we have?
* Will this system significantly improve the organization?
* Does the old system necessarily need to be enhanced?

**2.1 SCOPE**

Perform continuous improvement process for Pershing Security Framework by developing junit modules for all the uses cases involved. This can be used for providing an automated process.

* 1. **PURPOSE**

This project is to develop an automated process in such a way that these modules will be invoked during build time and will check for any gaps programmatically. Hence avoiding any human intervention for validation.

* 1. **RESOURCES REQUIRED:**

This phase we need to analyze the availability of the resources that are required to design, develop and implement and test the project. The resources need to be analyzed are manpower, time and the system requirements. Teams of individual member are involved in the entire SDLC lifecycle except the testing phase.The testing phase is guided by implementing junit modules which will be invoked during build time for checking any gaps programmatically.

* 1. **FEASIBILTY STUDY**

Feasibility study is an evaluation of system regarding to its workability,impact on organization, ability to meet user needs and effective use of resources. All projects are feasible given unlimited resources and infinite time.The feasibility study is to determine whether the solution is achievable,given the organizations resource and constraints.By performing the feasibility study the scope of the system will be defined completely. It is both necessary and prudent to evaluate the feasibility of the project at the earliest possible time.Feasibility and risk analysis is related in many ways. If project risk is great the feasibility listed below is equally important.

The following feasibility techniques has been used in this project

* Technical feasibility
* Operational feasibility
* Economic feasibility

**2.4.1Technical Feasibilty**

Technical feasibility analysis makes a comparison between the levels of technology available that is needed for the development of the project. The level of technology consists of factors like software tools, machine environment ,platform developed and so on.Technical feasibility centers around the hardware and software and to what extent it can be deployed to run successfully. The tools that are used to develop the application are the best tools available in the technological scenario and hence it requires efficient and versatile programmers and programming skills .Maven ,junit plugin ,oracle database and an IDE are sufficient for the project to be technically feasible.

**2.4.2Economic Feasibilty**

Economic feasibility is used for evaluating the effectiveness of the system.The purpose of the economic feasibility assessment is to determine the positive economic benefits to the organization that the proposed system will provide. It includes quantification and identification of all the benefits expected. This assessment typically involves a cost/ benefits analysis.All the softwares involved in the project are open source. So it can be easily downloaded fom the internet.So no high cost is involved. Regarding the maintenance ,since the source code will be with us , any small and necessary changes can be done with minimum maintenance cost involved in it.Hence the system is economically feasible and the risk of finance doesn’t exist.

**2.4. 3 Operational Feasibilty**

Operational feasibility is necessary as it ensures the success of the project. Certain tests have been carried out to ensure the operational feasibility of the system. The proposed system operates well with all types of connection. All the capabilities work well if proper functioning environment is provided. The developed system is operatableby a range of people and it is consistent .Hence the system is operationally feasible.

**2.5 SYSTEM SPECIFICATION**

**2.5.1 HARDWARE SPECIFICATIONS**

Processor : Any processor above 500 MHz

Ram : 1 GB

Hard Disk : 50 GB

Input device : Standard Keyboard and Mouse.

Output device : Display Screen

**2.5.2 SOFTWARE SPECIFICATIONS:**

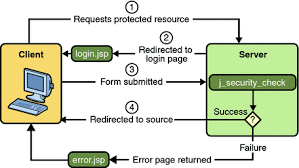
* Eclipse IDE
* Maven
* Junit plugin
* Oracle database

**CHAPTER 3**

**SYSTEM DESIGN**

System design is the process or art of defining the hardware and software architecture, components , modules, interfaces and data for a computer system to satisfy specified requirements. One could see it as an application of system theories to computing. Some overlap with the discipline of system analysis appears inevitable.Design tools such as Unified Modelling Language (UML), now addresses some of the issues of computer systems design and interfacing.

* 1. **System Architecture**



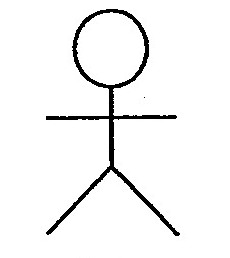
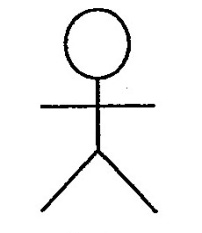
* 1. **Use Case Diagram**

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved.

Enter login credentials

Check Details for authenticatiom

Provide access to resources

 ****

Create and manage sessions

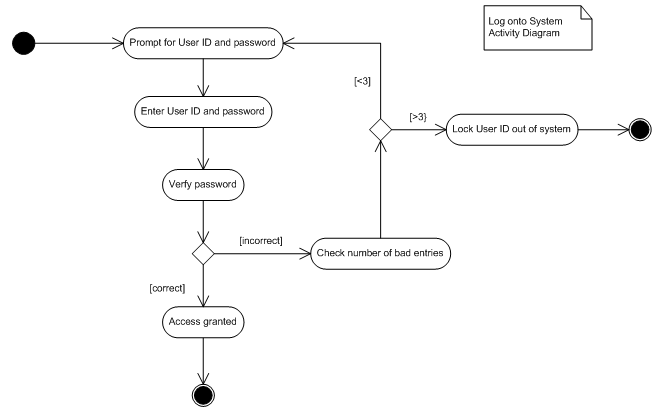
**Administrator End user**

Adding or removing user

Logout

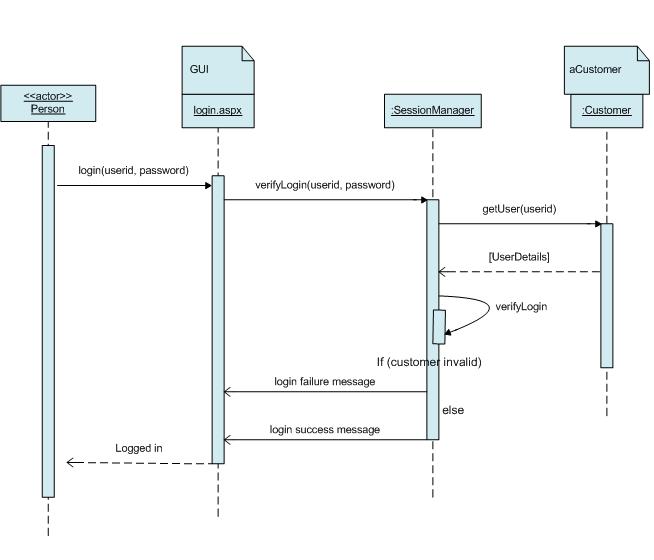
* 1. **Activity Diagram**

Activity diagram is used to describe the dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another activity.



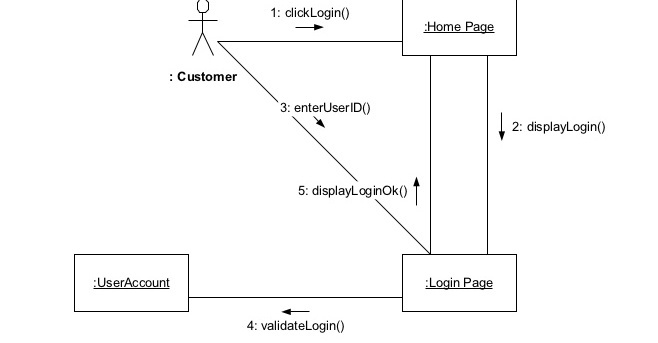
* 1. **Sequence Diagram**

A Sequence diagram is an interaction diagram that shows how processes operate with one another and in what order.A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.



* 1. **Collaboration Diagram**

A collaboration diagram resembles a flowchart that portrays the roles, functionality and behavior of individual objects as well as the overall operation of the system in real time. Objects are shown as rectangles with naming labels inside. These labels are preceded by colons and may be underlined. The relationships between the objects are shown as lines connecting the rectangles.



**CHAPTER 4**

**TECHNOLOGIES USED**

**4.1Javascript**

JavaScript is a lightweight, interpreted programming language. It is designed for creating network-centric applications. It is complimentary to and integrated with Java. JavaScript is very easy to implement because it is integrated with HTML. It is open and cross-platform.JavaScript ("JS" for short) is a full-fledged dynamic programming language that, when applied to an HTML document, can provide dynamic interactivity on website

**4.2MAVEN:**

Apache maven is a software management and comprehension tool. Based on the concept of Project Object Model (POM) model. Maven can manage a project’s build ,reporting and documentation from a central piece of information.

**Mavens Objectives**:

Maven’s primary goal is to allow a developer to comprehend to comprehend the complete state of the development effort in the shortest period of time.In order to attain this goal there are several areas of concern that Maven attempts to deal with:

* Making the building process easy.
* Provide a uniform build system.
* Providing a quality project information.
* Providing guideline for best practices in development.
* Allowing transparent migration for new features.

**Features of Maven:**

* Simple project setup that follows best practices-get a new project or module started in seconds.
* Consistent usage across all project means that ramp up time for new developers coming onto a project.
* Super dependency management including automatic updating,dependency closures(also knowns as transitive dependencies).
* Able to work with multiple projects at the same time.
* A large and growing repository of libraries and metadata to use out of box, and arrangements in place with large open source projects for real time availability of their latest releases.

**4.3JUNIT**:

JUnit promotes the idea of "first testing then coding", which emphasis on setting up the test data for a piece of code which can be tested first and then can be implemented . This approach is like "test a little, code a little, test a little, code a little..." which increases programmer productivity and stability of program code that reduces programmer stress and the time spent on debugging.

**Features**

* JUnit is an open source framework which is used for writing & running tests.
* Provides Annotation to identify the test methods.
* Provides Assertions for testing expected results.
* Provides Test runners for running tests.
* JUnit tests allow you to write code faster which increasing quality
* JUnit is elegantly simple. It is less complex & takes less time.
* JUnit tests can be run automatically and they check their own results and provide immediate feedback. There's no need to manually comb through a report of test results.
* JUnit tests can be organized into test suites containing test cases and even other test suites.
* Junit shows test progress in a bar that is green if test is going fine and it turns red when a test fails.

The packages in junit are

* **org.hamcrest**-The stable API defining matcher and its associated interfaces and classes.
* **org.hamcrest.core**-Fundamental matchers of objects and values and composite matchers
* **org.junit**-Provides junit core classes and annotations.
* **org.junit.runner**: Provides classes used to describe ,collect,run and analyse multiple tests.
* **org.junit.runner.manipulation**:Provide classes to filter or sort tests.
* **org.junit.runner.notification**:Provides information about a test run.
* **org.junit.matchers**:Provides useful matchers for use with the *Assert.assertThat(Object,org.hamcrest.Matcher)* statement.
* **org.junit.runners**: Provide standard runner implementations.

**Assertions**:

Junit provides overloaded assertion methods for all primitive types and Objects and arrays (of primitives or Objects). The parameter order is expected value followed by actual value. Optionally the first parameter can be a String message that is output on failure. There is a slightly different assertion,  assertThat  that has parameters of the optional failure message, the actual value, and aMatcher  object. Note that expected and actual are reversed compared to the other assert methods.

Some of the methods in junit are

* assertEquals(expected, actual)
* assertEquals(message, expected, actual)
* assertEquals(expected, actual, delta)
* assertEquals(message, expected, actual, tolerance ) -Test that float or double values match. The tolerance is the number of decimals which must be the same.
* assertFalse(condition)
* assertFalse(message, condition)
* Assert(Not)Null(object)
* Assert(Not)Null(message, object)
* Assert(Not)Same(expected, actual)
* Assert(Not)Same(message, expected, actual)
* assertTrue(condition)
* assertTrue(message, condition)
* assertArrayEquals(expectedArray, resultArray)

**JUNIT ANNOTATIONS:**

**@Test**: The Test annotation tells JUnit that the public void method to which it is attached can be run as a test case. To run the method, JUnit first constructs a fresh instance of the class then invokes the annotated method. Any exceptions thrown by the test will be reported by JUnit as a failure. If no exceptions are thrown, the test is assumed to have succeeded

**@Test(timeout=100):** Somethimes we need to mesure the performance interms of time. The @Test annotations provides an optional parameter called 'timeout', which causes a test to fail if it takes longer than a specified amount of clock time (measured in milliseconds).

**@Before:** When writing tests, it is common to find that several tests need similar objects created before they can run. Annotating a public void method with @Before causes that method to be run before the Test method. The @Before methods of super classes will be run before those of the current class.

**@After:** If you allocate external resources in a Before method you need to release them after the test runs. Annotating a public void method with @After causes that method to be run after the Test method. All @After methods are guaranteed to run even if a Before or Test method throws an exception. The @After methods declared in superclasses will be run after those of the current class.

**@Ignore:** Sometimes you want to temporarily disable a test or a group of tests. Methods annotated with Test that are also annotated with @Ignore will not be executed as tests. Also, you can annotate a class containing test methods with @Ignore and none of the containing tests will be executed. Native JUnit 4 test runners should report the number of ignored tests along with the number of tests that ran and the number of tests that failed.

You can also use @Ignore annotation at class level.

**Features of Junit**:

* Fixtures
* Test Suite
* Test runners
* Junit Classes

**Fixtures**:

Fixtures are the fixed state of a set of objects used as a base line for running tests.

setup() method which runs before every test invocation.

teardown() method which runs after every test method.

**Test Suite**:

Test Suite is a bundle of few unit test cases and run it together.

@RunWith and @Suite annotation are used to run the suite test.

**Test Runner**:

Test runner used to run the test cases.

Eg: JUnitCore.runClasses(TestJunit.class)

**Junit Classes**:

Assert which contain a set of assert methods.

TestCase which contain a test case defines the fixture to run multiple tests.

TestResult which contains a method to collect the result of executing a test case.

**Junit using Maven**:

Dependency to be added to use junit in maven is

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.12</version>

<scope>test</scope>

</dependency>

**4.4 Java 7/J2EE:**

Developers today increasingly recognize the need for distributed, transactional, and portable applications that leverage the speed, security, and reliability of server-side technology. **Enterprise applications** provide the business logic for an enterprise. They are centrally managed and often interact with other enterprise software. In the world of information technology, enterprise applications must be designed, built, and produced for less money, with greater speed, and with fewer resources.

With the Java Platform, Enterprise Edition (Java EE), development of Java enterprise applications has never been easier or faster. The aim of the Java EE platform is to provide developers with a powerful set of APIs while shortening development time, reducing application complexity, and improving application performance.

The Java EE platform is developed through the Java Community Process (JCP), which is responsible for all Java technologies. Expert groups composed of interested parties have created Java Specification Requests (JSRs) to define the various Java EE technologies. The work of the Java Community under the JCP program helps to ensure Java technology's standards of stability and cross-platform compatibility.

The Java EE platform uses a simplified programming model. XML deployment descriptors are optional. Instead, a developer can simply enter the information as an **annotation** directly into a Java source file, and the Java EE server will configure the component at deployment and runtime. These annotations are generally used to embed in a program data that would otherwise be furnished in a deployment descriptor. With annotations, you put the specification information in your code next to the program element affected.

In the Java EE platform, dependency injection can be applied to all resources a component needs, effectively hiding the creation and lookup of resources from application code. Dependency injection can be used in Enterprise JavaBeans (EJB) containers, web containers, and application clients. Dependency injection allows the Java EE container to automatically insert references to other required components or resources, using annotations.

**4.5Eclipse**

Eclipse is an integrated development environment (IDE). It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages through the use of plugins

**CHAPTER 5**

**PROJECT DESCRIPTION**

Authentication is the process by which a security principal— human, machine or otherwise—asserts and proves its identity. Username/password logins are a familiar example.Spring Security has rich support for authentication. We can source authentication data from databases, LDAP, OpenID providers, CAS, and more.

It performs the following tasks

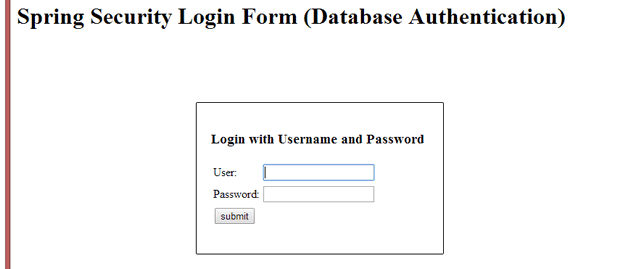
* A role based security mechanism can be realised byimplementing the UserDetailsService interface and configuring spring security to use this class.
* Upon authentication (e.g. submitting a login form) an Authentication Object is created which holds the login credentials.
* The authentication object is passed to an AuthenticationManager, which can be thought of as the controller in the authentication process.

.

To perform database authentication, you have to create tables to store the users and roles detail.JSP pages for custom login page.This page is password protected, only authenticated user “**ROLE\_ADMIN**” is allowed to access.

**FETCHING ROLES OF USER**

You can't modify the server roles or add new server roles. A login can belong to more than one role and acquires a combination of the privileges of the roles it belongs to. This API is used to fetch the roles of a user by invoking the service. It will return a List of roles and the access granted by the roles.



**REMOVING LOGIN OF A USER**

You'll eventually need to remove logins from the Server. Server prevents you from removing a login if the login is associated with the user name of a user who owns objects in any database. This rule prevents objects from being orphaned, which would be a violation of referential integrity in the database. If you follow good programming practices, only the database owner (dbo) can own objects, so deleting a user and login account won't cause any conflicts. Also, you can't remove a login that's in use.

**SESSION MANAGEMENT**

Once the user logins with the correct credentials a session is created for the user and it remains active until the session is closed.You will need to identify the maximum age of any given session ID as well as a timeout for sessions. There is often the requirement to re-authenticate users during a session for security reasons.

**WRITING UP JUNIT MODILES**

For each of the methods involved in the package we write up separate junit modules to analyze the success ,failure and the various error thrown for the sample input. This provides us successful execution of each methods and the errors recorded helps us to do the failure trace and rectify them.

The junit modules for various methods in each of the package are separately tested and ensured that they function properly.Separate junittestcase are coded for different methods in each of the package and the maven test is performed.Thejunit modules gives output as errors, failures ,time for completion and successful runs.Thejunit test cases throws the assertion failure if there is a difference between the expected and the actual outcome.

The various scenarious for failures and errors are analysed and tested for their functionality.This way of automation in such a way that these modules will be invoked during build time and will check for any gaps programmatically.

**PROCESS OF CREATING JUNIT MODULES:**

Junit plugins are added to eclipse for the purpose of writing junit cases.Once they are added successfully we can write codes of junit modules and check their functionality. Dependency to be added to use junit in maven is

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.12</version>

<scope>test</scope>

</dependency>

**CHAPTER 6**

**TESTING OF THE PROJECT**

**6.1TESTING**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies and/or a finished product. It is the process of exercising software with the intent to ensure that the software system meets its requirements and user expectations and does not fail unacceptable manner.

**6.2UNIT TESTING**:

Unit testing is a level of software testing where individual units of software are tested.The purpose is to validate that each unit of the software performs as designed.A unit is the small testable part of a software .It usually has one or few inputs and a single output.In object oriented programming,the smallest unit is a method which may belong to a base/ super class,abstractclass.Unit testing framework ,drivers, stubs are used to assist in unit testing.Unit testing is a first level of testing and performs prior to integration testing.Unit testing is normally performed by software developers themseleves or their peers or in some cases independent software testers.

**BENEFITS:**

* Unit testing increases confidence in changing/maintaining code.If a good unit tests are written and if they run every time any code is changed,we will be able to promptly catch any defects introduced due to the change.
* Also,if codes are already made less interdependent to make unit testing possible,the unintended impact of changes to any code is less.
* Codes are more reusable.Inorder to make unit testing possible,codes need to be modular.This means codes are easier to reuse
* The cost of fixing a defect detected during unit testing is lesser in comparison to that of defects detected at higher levels.
* Debugging is easy.When a test fails,only the latest changes need to be debugged.With testing at higher levels,changes made over a span of several days/weeks/months need to be scanned.
* Codes are more reliable.

**6.3 Running unit test with Maven in command prompt:**

To run unit test via Maven, issue this command **:mvn test**

$ **mvn test**

[INFO] Scanning for projects...

[INFO] ------------------------------------------------------------------------

[INFO] Building junit-tests

[INFO] task-segment: [test]

[INFO] ------------------------------------------------------------------------

[INFO] [resources:resources {execution: default-resources}]

[WARNING] Using platform encoding (MacRoman actually) to copy filtered

resources, i.e. build is platform dependent!

[INFO] Copying 0 resource

[INFO] [compiler:compile {execution: default-compile}]

[INFO] Compiling 1 source file to /Users/Tim/Library/Code/sonatype/

maven- Security.auth -examples/unit/junit-tests/target/classes

[INFO] [resources:testResources {execution: default-testResources}]

[WARNING] Using platform encoding (MacRoman actually) to copy filtered

resources, i.e. build is platform dependent!

[INFO] Copying 0 resource

[INFO] [compiler:testCompile {execution: default-testCompile}]

[INFO] Compiling 1 source file to /Users/Tim/Library/Code/sonatype/

maven- Security.auth /unit/junit-tests/target/test-classes

[INFO] [surefire:test {execution: default-test}]

[INFO] Surefire report directory: /Users/Tim/Library/Code/sonatype/

maven- Security.auth /unit/junit-tests/target/surefire-reports

T E S T S

-------------------------------------------------------

Running org. Security.auth.SeriousComponentTest

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.065 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESSFUL

[INFO] ------------------------------------------------------------------------

[INFO] Total time: 5 seconds

[INFO] Finished at: Wed Nov 25 14:42:39 CST 2009

[INFO] Final Memory: 17M/80M

[INFO] ------------------------------------------------------------------------

If the unit test fails the same file will contain a stack trace that points to a failed assertion on a specific line of the unit test as shown below:

-------------------------------------------------------------------------------

Test set: org. Security.auth.SeriousComponentTest

-------------------------------------------------------------------------------

Tests run: 1, Failures: 1, Errors: 0, Skipped: 0, Time elapsed: 0.062 sec

<<< FAILURE!

testSeriousness(org. Security.auth.SeriousComponentTest) Time elapsed:

0.017 sec <<< FAILURE!

junit.framework.AssertionFailedError: null

atjunit.framework.Assert.fail(Assert.java:47)

atjunit.framework.Assert.assertTrue(Assert.java:20)

atjunit.framework.Assert.assertTrue(Assert.java:27)

atorg.Security.auth.SeriousComponentTest.testSeriousness(

SeriousComponentTest.java:15)

**RUNNING MAVEN TEST WITH VM ARGUMENTS**

To run a specific unit test in a Maven build, pass the name of the test class to the parameter test on the command line.

$ **mvn test -Dtest=SeriousComponentTest**

[INFO] Scanning for projects...

[INFO] ------------------------------------------------------------------------

[INFO] Building testng-groups

[INFO] task-segment: [test]

[INFO] ------------------------------------------------------------------------

[INFO] [resources:resources {execution: default-resources}]

[WARNING] Using platform encoding (MacRoman actually) to copy filtered

resources, i.e. build is platform dependent!

[INFO] Copying 0 resource

[INFO] [compiler:compile {execution: default-compile}]

[INFO] Nothing to compile - all classes are up to date

[INFO] [resources:testResources {execution: default-testResources}]

[WARNING] Using platform encoding (MacRoman actually) to copy filtered

resources, i.e. build is platform dependent!

[INFO] Copying 0 resource

[INFO] [compiler:testCompile {execution: default-testCompile}]

[INFO] Nothing to compile - all classes are up to date

[INFO] [surefire:test {execution: default-test}]

[INFO] Surefire report directory: /Users/Tim/Library/Code/sonatype/

maven-/AUTH/SECURITY/unit/testng-groups/target/surefire-reports

-------------------------------------------------------

T E S T S

-------------------------------------------------------

Running TestSuite

Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.466 sec

Results :

Tests run: 5, Failures: 0, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESSFUL

[INFO] ------------------------------------------------------------------------

[INFO] Total time: 5 seconds

[INFO] Finished at: Thu Nov 26 13:01:58 CST 2009

[INFO] Final Memory: 11M/80M

[INFO] ------------------------------------------------------------------------

**CHAPTER 7**

**IMPLEMENTATION AND MAINTENANCE**

**7.1 IMPLEMENATION**

This project is implemented in an organization to provide authorized user access to resources. It fetches the necessary information for different kinds of authorized user based on their roles and validate them .

**7. 2 MAINTENANCE**

System maintenance is the phase of software engineering that helps the user and the developer to maintain the software in an environment required by the user.. It is this phase in which any sort of implementation notification are made to incorporate the changes that the user need.he Maintenance Phase occurs once the system is operational. It includes implementation of changes that software might undergo over a period of time, or implementation of new requirements after the software is deployed at the customer location. The maintenance phase also includes handling the residual errors that may exist in the software even after the testing phase. This phase also monitors system performance, rectifies bugs and requested changes are made.

**CHAPTER 8**

**CONCLUSION**

The proposed system is found to be working efficiently and effectively. The system is developed with much attention over its quality and reliability. This system is found to be satisfactorily running under real environment. The system is perfectly tested for its efficiency, flexibility, accuracy and user friendly. The new system makes the task more secure and faster.

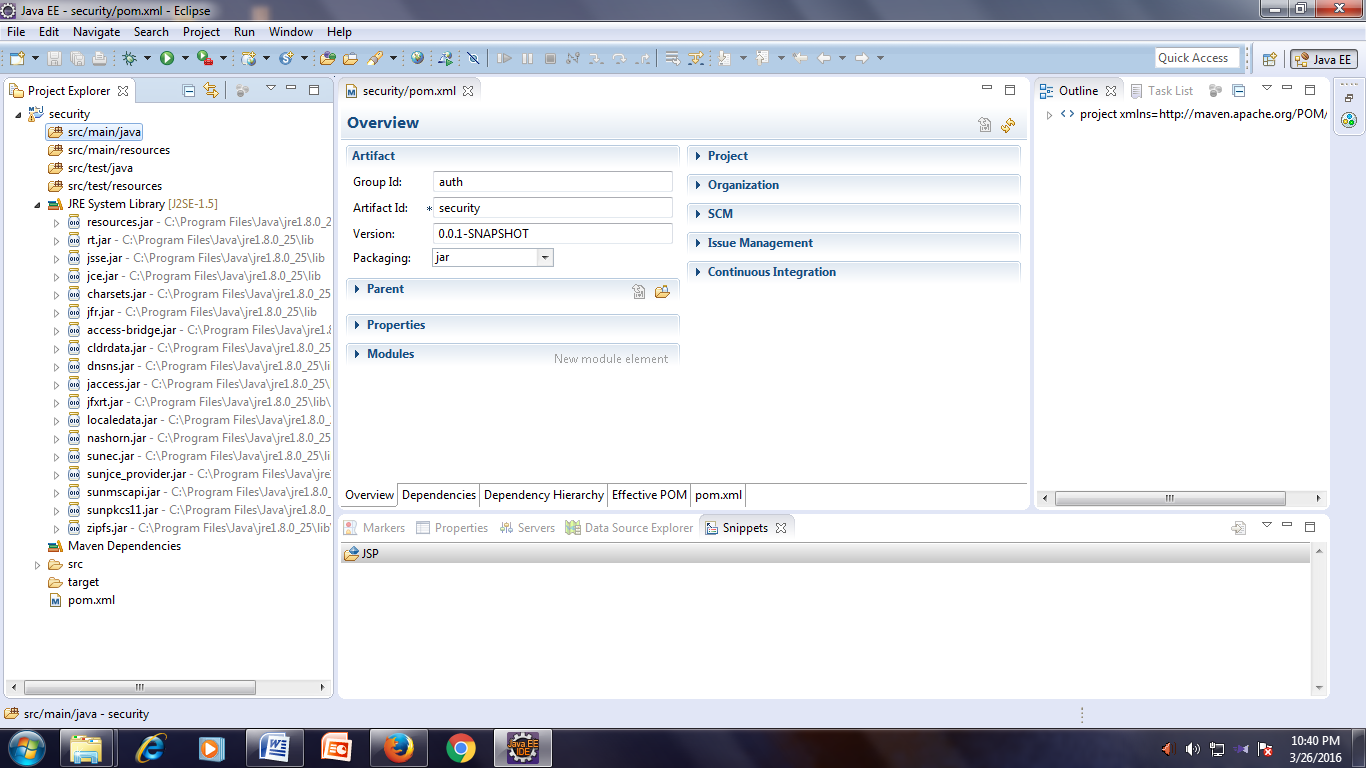
Any errors in the program will be detected during the running of junit testcase and the necessary changes are made it to run successfully. Automation is provided in such a way that these modules will be invoked during build time and will check for any gaps programmatically.

Thus this project provides automatic security login for accessing resources without any issues by checking for errors during the build time itself which makes it much more effective working.

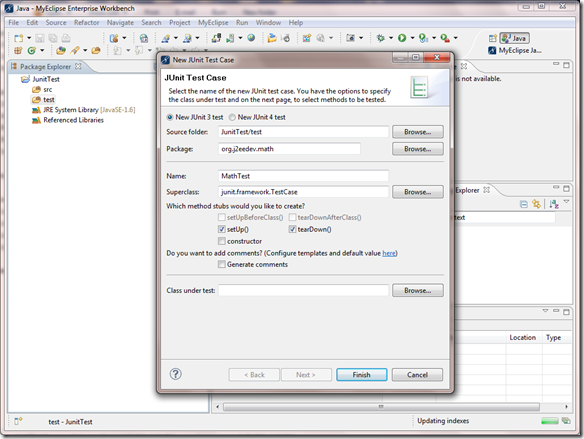
**APPENDIX 1**

**SCREENSHOTS**

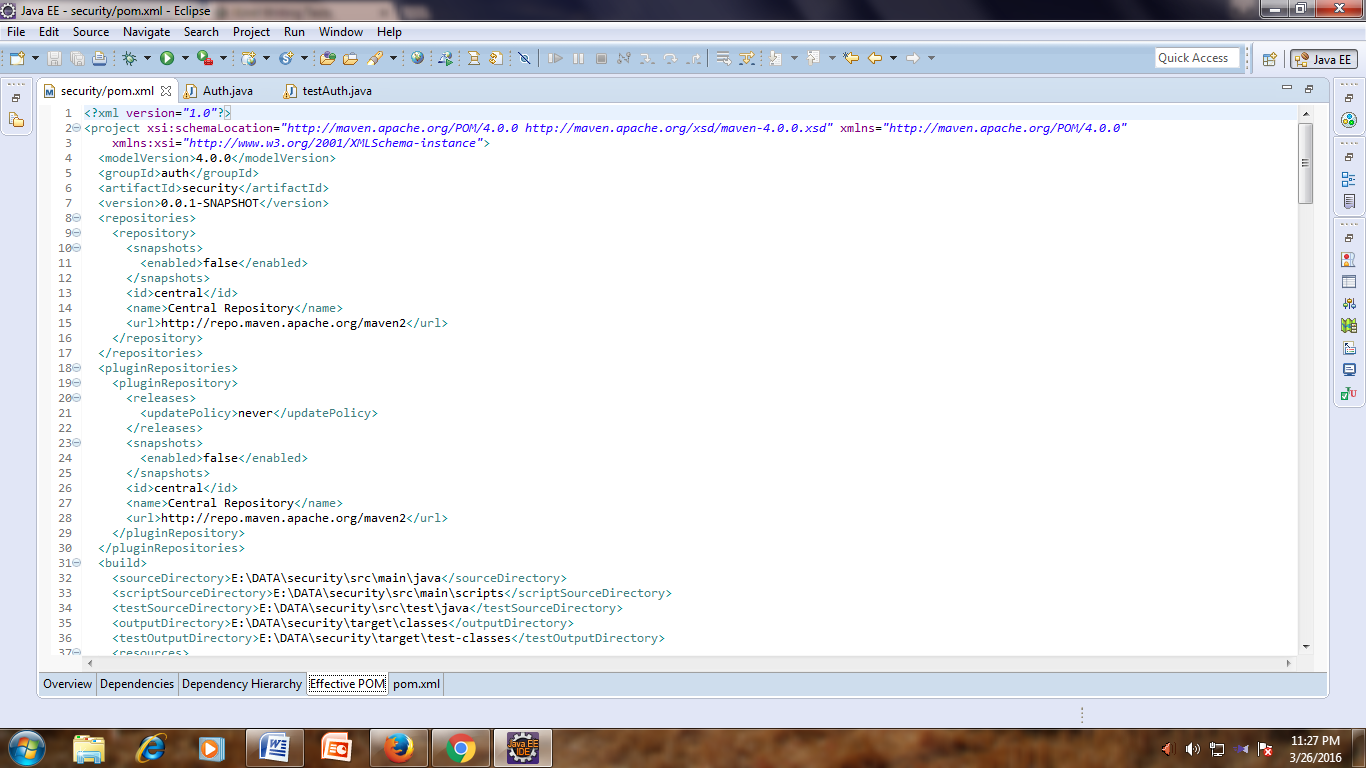
**MAVEN PROJECT SNAPSHOTS**

****

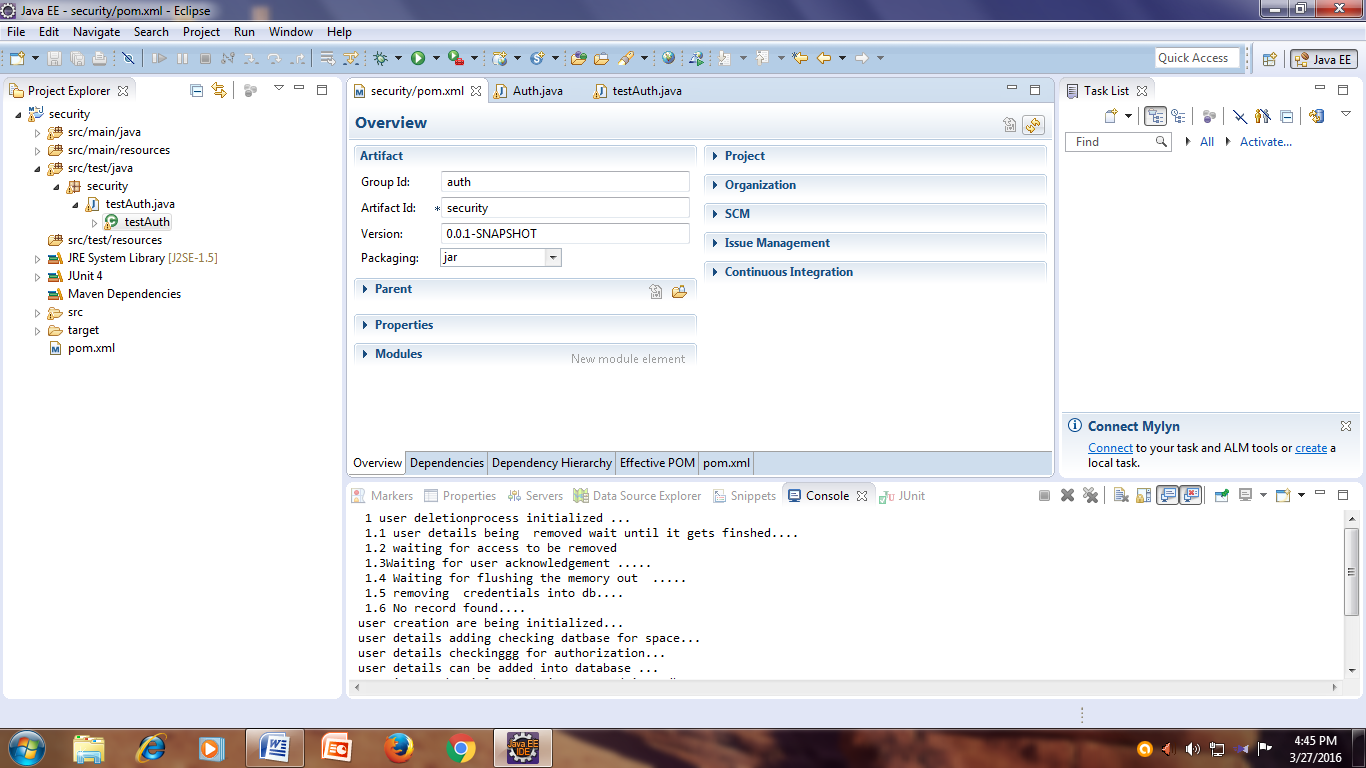
**CREATING NEW JUNIT TESTCASE**



**POM . XML**

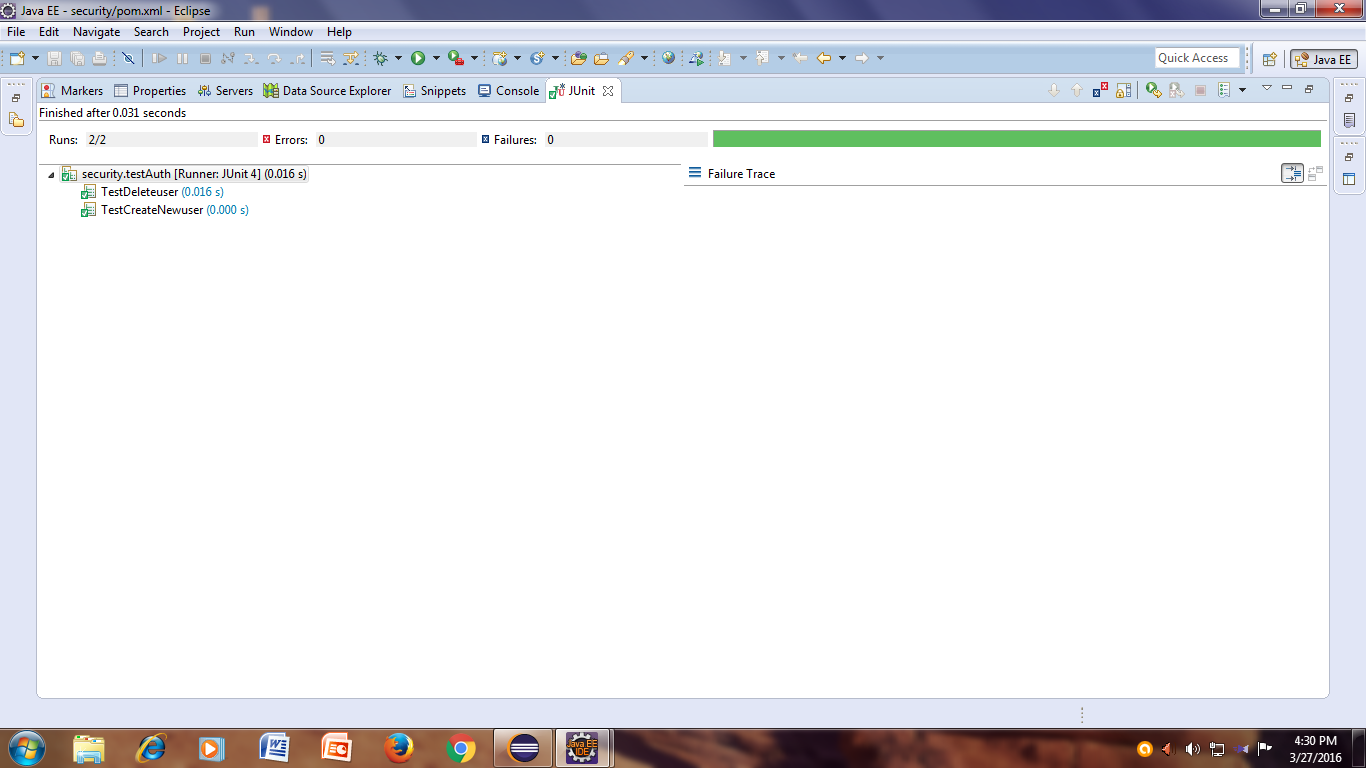


**CONSOLE VIEW:**

****

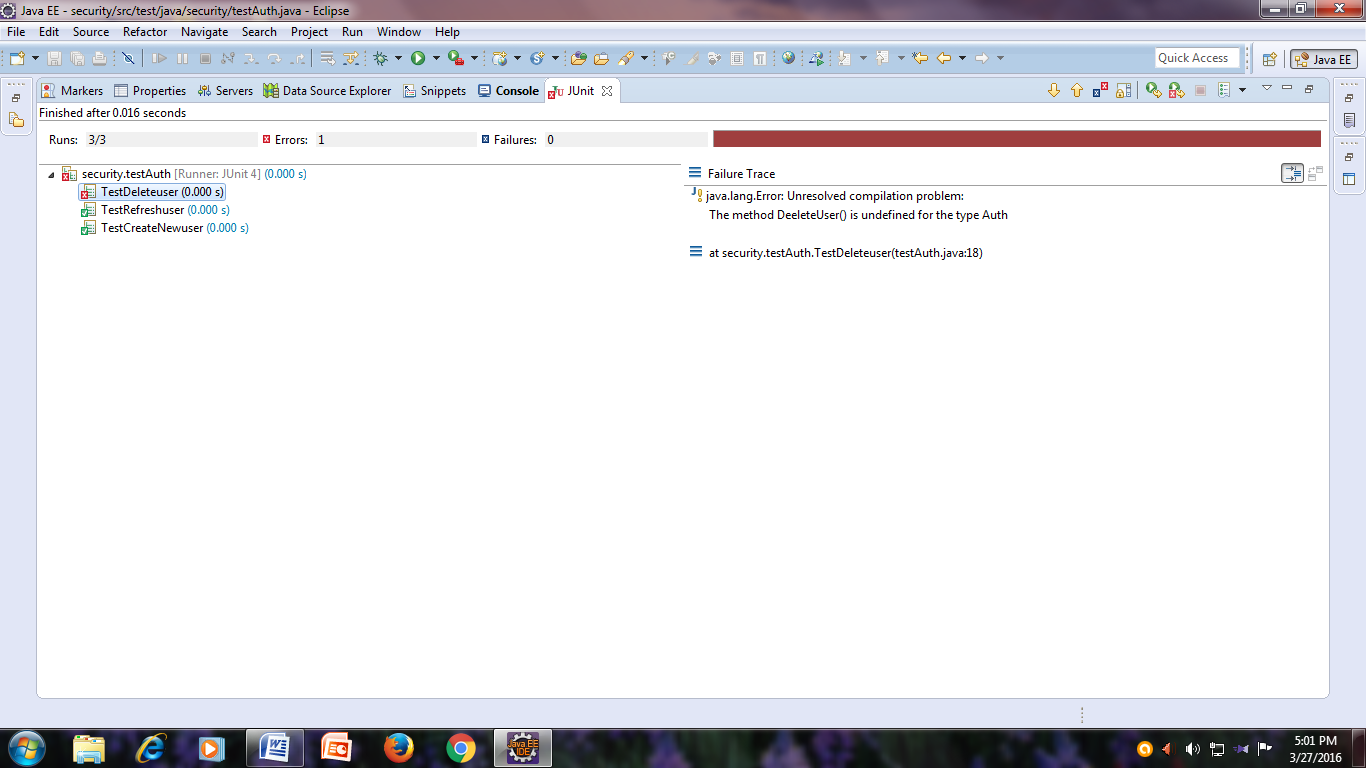
Console view shows the actual execution of the methods taking place in the package.It serves as an output to the users on the execution of the package.

**JUNIT MODULES ON SUCCESSFUL RUNNING**



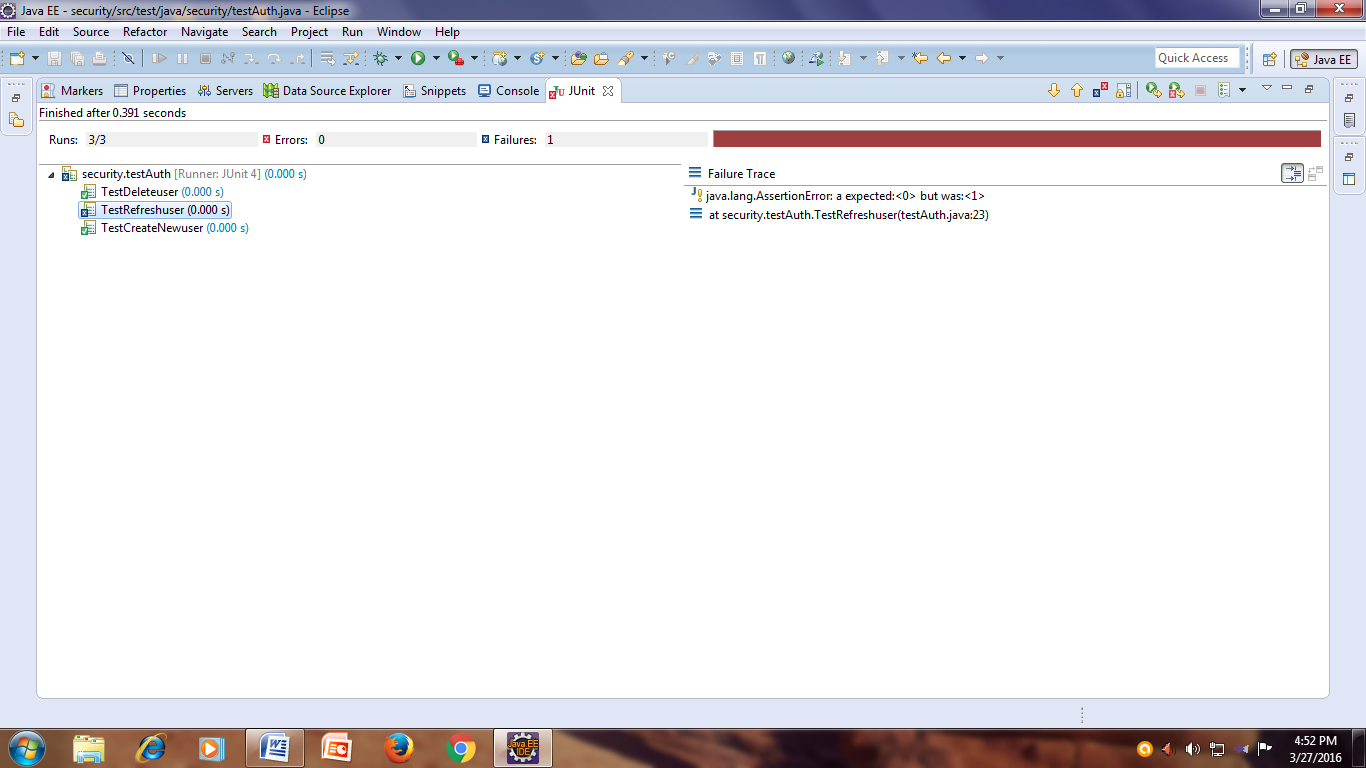
Here in this case, both the methods are running successfully without any error.If any error is detected in the method it will be shown in error .

**JUNIT MODULES ON ERROR**

****

Errors in the methods are thrown on execution of the junit modules and the failure trace helps in correcting the errors.

**JUNIT MODULES ON FAILURE:**

****

An assertion error is thrown , indicating a failure of method. The methods which are failed are differentiated from the other methods .In the above program the second method has failed while the other 2 methods are running without any issues.The failure in the above case is the assertion failure where an output is expected but the actual result is different from the expected one.

**REFERENCES :**

1. <https://books.sonatype.com/mcookbook/reference/unit-sect-junit>.
2. <http://faculty.washington.edu/stepp/courses/2005spring/tcss360/handouts/6-junit_eclipse_2>.
3. [https://www.informit.com/guides/content.aspx/java - junit reference 4](https://www.informit.com/guides/content.aspx/java%20-%20junit%20reference%204)
4. <http://frightanic.com/software-development/adding-references-junit-tests/>
5. <http://automatic-generation/methods-of-unit-test-cases-for-net-and-java>
6. <https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html>