**CHAPTER-1**

**INTRODUCTION**

In today’s world everything is computerized particularly in the government sector. So, in our project we have tried to automate the work in the signal and telecommunication department of south central railways. In our project we have mainly concentrated on the automation of Amenity issue, receive and issue of challans to various stations in which work is going on manually at present. Our main aim is to automate the work that is being done manually by making use of database and web technologies. This system reduces the amount of manual work and also reduces the errors that were done by mistake without their notice.

**1.1 Literature Survey**

We are proposing an automated system where we can overcome all the disadvantages of the existing system and come up with new system that works effectively and efficiently. Here we include a module to maintain the ledger of the amenities and we are also maintaining the challan inbox to identify whether acknowledgement was received or not. Administrator creates the accounts of software senior engineer’s and gives their user id and password.

**1.2 General**

The maintenance of all the Material and reports is being done manually in the existing system. So it is becoming very difficult to find out the current status of amenity availability, amenity delivered and so on. To overcome this drawback, we are computerizing the system. The staff can maintain the database of all the Material and challan status received by the department. The system automatically generates an acknowledged status when SSE clicks on a link.

**1.3 Statement of the Problem**

The maintenance of all details is being done manually in the existing system. So it is becoming very difficult to find out Material in a particular station, stations of a particular amenity, company details of particular station and amenity, availability details of the Material. To overcome this drawback, computerizing these system details of the stations, Material, company details, availability details of the Material is needed.. It is automatically updated in the

main server which is in Vijayawada so that we can easily dispatch the Material to various sub stations like BTTR,GDV,GDR,RJY and also we can easily check their delivery status.

**1.3.1 Existing System**

The existing system is very less accurate since all the work is done manually. In the existing system all the amenity details are maintained in ledger books. The stations should request for Material manually and the request is served manually.

**Disadvantages:**

* It takes a lot of time.
* More manpower is required hence less economical.
* Challans are maintained manually.
* No exact information about a particular amenity.

**1.3.2 Proposed System**

The proposed system automates the maintenance of all the Material replacing the ledger books. The request from clients is served faster than the existing system. The request is sent to the server as a mail. The monthly and yearly records about dispatching of goods are also maintained automatically.

**1.3.3 Features:**

* User friendly environment to get the required information.
* Providing the provision for flexibility in report generation.
* To Enhance loading with check on restrictions.
* Refreshing the database when unloading is done

**1.4 Objectives of the project:**

The main aim of this project is “To moving equipment for railway signal and telecommunication amenities maintenance, to send the request challan from different stores of Vijayawada division through online, used to know the amenity availability and used to send the SMS automatically when the material is required in the stores” using J2EE technology.

The equipment will be transferred to different stores located at each station. The firms or different stations may apply for registration for the supply of one or more groups of stores giving reference to Challan Number (CNO) of the group of the attached list.

This project is mainly useful for the following things

* To issue Material through online.
* To maintain Material in the database.
* To view the status of the dispatched Material.

**1.5 Requirement Analysis**

In the proposed system we are overcoming all the disadvantages of the existing manual system by making it automated and easing the result analysis also providing better data security.

**1.5.1 Functional Requirements:**

The system should satisfy the following requirements:

1. System should be able handle multiple users.

2. Database updating should follow transaction processing to avoid data inconsistency.

3. Only Registered users must be privileged to use the system.

**1.5.2 Non-Functional requirements:**

The major non-functional Requirements of the system are as follows

* **Usability:** The system is provided with a good user interface in order to make user to

Provide certain operations.

* **Reliability:** The system is more reliable because of the qualities that are inherited from the chosen platform java. The code built by using java is more reliable.
* **Performance:** Performing Result analysis includes high level of data processing which hampers performance of the system rapidly.
* **Supportability:** The system is designed to be the cross platform supportable. The system is supported on a wide range of hardware and any software platform, which is having JVM, built into the system.

**1.6 System Requriment Specifications**

**1.6.1 Hardware Requirements:**

RAM : 256MB

CACHE : 256 KB

PROCESSOR : Pentium4

HARD DISK : 40GB

**1.6.2 Software Requirements:**

OPERATING SYSTEM : WINDOWS XP

USER INTERFACE : JSP

PROGRAMMING LANGUAGE : JAVA

WEB TECHNOLOGY : J2EE

DATABASE : ORACLE 10g

TOOLS : NETBEANS IDE 6.1

WEB SERVER : APACHE TOMCAT 6.0

**CHAPTER-2**

**SYSTEM DESIGN**

**2.1 UML DIAGRAMS**

The UML diagrams namely the *class Diagram* for the system giving the relations between different classes of the system, a set of *Sequence Diagrams* depicting the flow between the objects of the classes of this system, *use Case diagrams* for each of the actors, *State Chart Diagrams* for show in the various states of each object, *Activity Diagram* for the system are as shown below.

**2.1.1. IMPORTANCE OF UML IN MODELLING**

This is a programming language that is used for object-oriented software development. To organize program code more efficiently, programmers often create “objects” that are sets of structured data within programs. UML, which has been standardized by the Object Management Group (OMG), was designed for this purpose. The language has gained enough support that it has become a standard language for visualizing and constructing software programs.

**AN OVERVIEW OF UML**

* Language for Visualizing.
* Language for Specifying
* Language for Constructing
* Language for Documenting

**2.1.2 A CONCEPTUAL MODEL OF UML**

* The UML`s basic building blocks.
* The rules that dictate how those building blocks may be put together.
* Some common mechanisms that apply throughout the UML.

**2.2 CLASS DIAGRAM**

Classes are the most important building block of any object oriented system. A class is a description of set of objects that share the same attributes, operations, relationships and semantics. A class implements one or more interfaces. It is graphically rendered as a rectangle

Classes may contain:

**Names:** Every class must have a name that distinguishes it from other classes

**Attributes:** It is name property of a class that describes a range of values that instances of the property may hold the system. A sequence diagram shows a set of objects and messages that are dispatched between those objects based on time-ordering.

**Classes:** A class is a description of set of objects that share the same attributes, operations, relationships and semantics

**Interfaces:** An interface is a collection of operations that are used to specify a service of a class or component

**Collaborations:** It is a society of classes, interfaces and other elements that work together to provide some Co-operative behavior that’s bigger than some of all its parts. It is rendered as an ellipse with dashed lines

**IDENTIFYING RELATIONSHIPS:**

In UML the ways that things can connect to each other wither logically or physically are modeled as relationships. In object-oriented modeling there are three kinds of relationships that are most important, they are

1. **Dependencies:** It is a using relationship that states that a change in specification of one thing may affect another thing that uses each. Graphically dependency is rendered as a dashed directed line.
2. **Generalization:** It is relationship between a general thing (called parent) and a more specific kind of that thing (called the child).Generalization is sometimes called ‘is-a-kind-of’ relationship. It means that objects of the child may be used anywhere that parent may appear. Graphically it is rendered as a solid directed line with a large open arrow head pointing to the parent.
3. **Association:** It is a structural relationship that specifies that objects of one thing are connected to the objects of another. The associations that connect more than two classes are called n-array associations. Graphically it is rendered as solid line connecting the same or different classes.

****

Figure 2.1 Class diagram of the DMTR system

**Description:** The above diagram is showing various classes like login, registration, amenity maintainence, ledger maintainence, and amenity issued and amenity received. The relationship between various classes is also clearly shown in the figure.

**2.3 SEQUENCE DIAGRAMS**

A sequence diagram emphasizes the time ordering of messages. These are used to model the dynamic aspects is a tall, thin rectangle that shows the period of time during which an object is performing an action, either directly or a sub ordinate procedure. The top of the rectangle is aligned with the start of the action the bottom is aligned with its completion.

Sequence-diagram has two features:

1. **The object life line:** An object life line is the vertical dashed line that represents the existence of an object over a period of time. Most objects that appear in an interaction diagram will be in existence for the duration of the interaction, so these objects are all aligned on the top of the diagram, with their lifelines drawn from the top of the diagram to the bottom.

2. **The focus of control** is a tall, thin rectangle that shows the period of time during which an object is performing an action, either directly or a sub ordinate procedure. The top of the rectangle is aligned with the start of the action the bottom is aligned with its completion.



Figure 2.2 Sequence Diagram for Sr. DSTE registering the users.

**Description:** The above diagram explains about registering the new users.



Figure 2.3 Sequence Diagram for DSTE registering the users

**Description:** The above diagram shows various objects like Sr. DSTE, DMTRS, Database and the messages that are passed between them while Sr. DSTE registers the users. Validation of the admin login takes place and after successful registration Sr. DSTE logout.



Figure 2.4 Sequence diagram for SSE/HQ issuing the Material

**Description:** The above sequence diagram shows the sequence of operations that are carried out when SSE/HQ issuing the Material.



Figure 2.5 Sequence diagram for SSE/HQ for maintaining the Material

**Description:** The above sequence diagram shows the sequence of messages that are carried out while maintaining the Material. SSE/HQ can add, delete, and update the Material.



Figure 2.6 Sequence diagram for SSE clicking on acknowledgement link on receiving the Material

**Description:** The above sequence diagram shows the sequence of actions that are carried out when the SSE receives the goods and send acknowledgement.

**2.4 Collaboration diagrams**

Collaboration diagrams and sequence diagrams are alternate representations of an interaction. A collaboration diagram is an interaction diagram that shows the order of messages that implement an operation or a transaction. A sequence diagram shows object interaction in a time-based sequence.

Collaboration diagrams show objects, their links, and their messages. They can also contain simple class instances and class utility instances. Each collaboration diagram provides a view of the interactions or structural relationships that occur between objects and object-like entities in the current model.

The Create Collaboration Diagram Command creates a collaboration diagram from information contained in the sequence diagram. The Create Sequence Diagram Command creates a sequence diagram from information contained in the interaction's collaboration diagram. The Sequence Diagram and Collaboration Diagram commands traverse between an interaction's two representations.

Collaboration diagrams contain icons representing objects. You can create one or more collaboration diagrams to depict interactions for each logical package in your model. Such collaboration diagrams are themselves contained by the logical package enclosing the objects they depict.

An Object Specification enables you to display and modify the properties and relationships of an object. The information in a specification is presented textually. Some of this information can also be displayed inside the icons representing objects in collaboration diagrams.

You can change properties or relationships by editing the specification or modifying the icon on the diagram. The associated diagrams or specifications are automatically updated.

****

Figure 2.7 Collaboration diagram for Senior DSTE

**Description:** The above diagram shows the operations carried out by Sr. DSTE.

****

Figure 2.8 Collaboration diagram for DSTE

**Description:**The above diagram shows the ledger form details by DSTE after login.



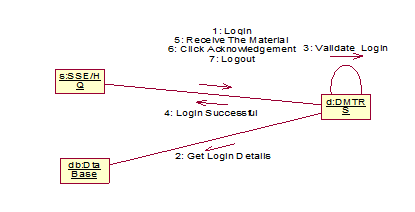
Figure 2.9 Collaboration diagram for SSE/HQ

**Description:** The above diagram displays about ledger form detail, filling challan form, issuing of material done by  SSE/HQ.

****

Figure 2.10 Collaboration diagram for SSE/HQ for maintaining the Material

**Description:** The above diagram is for maintaining the material that is adds, delete, and update the material in the store by SSE/HQ.

****Figure 2.11 Collaboration diagram for SSE clicking on acknowledgement link on receiving the Material

**Description:** The above diagram is for receiving material and sending the acknowledgement by SSE.

**2.5. USECASE DIAGRAM:**

A use case diagram shows a set of use cases and actors and their relationships. We apply the use case diagram to illustrate the static use case view of a system. Use case diagram are especially important in organizing and modeling the behaviors of a system.

**Use case:** A use case describes what a system (or a subsystem, class or interface) does but it does not specify how it does it. For dynamic aspects of systems, use case diagram is used to model the behavior of the system, sub-system or classes. Each one shows a set of use case, actors and their relationship with role names

**Actor:** An actor is the one plays an active role in the system. Typically an actor represents a role that a human, a hardware device, or even another system plays with a system



Figure 2.12 Use case diagram for Senior DSTE

**Description:** The above use-case diagram displays the information about the registering the new users by Sr.DSTE.



Figure 2.13 Use case diagram for DSTE

**Description:** The above use-case diagram describes about validating the login, viewing the reports done by DSTE.



Figure 2.14 Use case diagram for SSE/HQ

**Description:** The above diagram shows amenity adds, update, delete details and issuing the materials by SSE/HQ.



Figure 2.15 Use case diagram for SSE

**Description:** The above diagram displays about receiving amenities details and verifying acknowledgement details done by SSE.

**2.6 Activity diagram**

The purpose of an activity diagram is to provide a view of flows and what is going on inside a use case or among several classes. An activity diagram is just to explain the internal operations performed and also the transitions that are triggered by the completion of the particular operations. At the abstract level it explains the sequence of the activities. This focus on the events occurring to a single object as it responds to messages, an activity diagram can be used to model an entire process.

**Transitions**

When the action or activity of a state completes flow of control passes immediately to the next action or activity state. You specify this flow by using transitions to show the path from one action or activity state to next action or activity state. You represent this as simple directed line.

**Object Flow**

Objects may be involved in the flow of control associated with an activity diagram. The use of dependency relationships and objects is called an object flow because it represents participation of an object in the flow of control.



Figure 2.16 Activity diagram for Sr. DSTE

**Description:** The above activity diagram describes about registering the new users and viewing the overall reports done by Sr. DSTE.



Figure 2.17 Activity diagram for SSE/HQ

**Description:** The above activity diagram displays about amenity stock maintainance, sending challan form, acknowledgement receiving done by SSE/HQ.

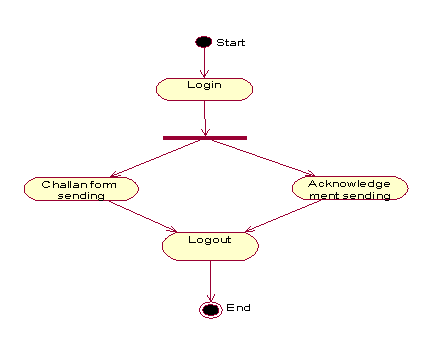


Figure 2.18 Activity diagram for SSE

**Description:** The above activity diagram shows the details of challan form and verifying the acknowledgement details done by SSE.

**2.7 ER- diagram for the entire system**

Registration

has

login

Maintains

Material

issue

Challan1

has

status

Issuebox

includes

2.19 ER- diagram for the entire system

**Description**: An entity-relationship model  is an abstract and conceptual representation of data. Entity-relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database, and its requirements in a top-down fashion.

**CHAPTER-3**

**IMPLEMENTATION**

The above projected is implemented in java language. The features of java language have made it apt to use it for the implementation. The portions of java used here are jsp, servlets, jdbc, and html.

**3.1 CODE DESIGN CHARACTERISTIC**

**3.1.1 Java Script**

**Introduction**

JavaScript is a script-based programming language that was developed by Netscape Communication Corporation. JavaScript was originally called Live Script and renamed as JavaScript to indicate its relationship with Java. JavaScript supports the development of both client and server components of Web-based applications. On the client side, it can be used to write programs that are executed by a Web browser within the context of a Web page. On the server side, it can be used to write Web server programs that can process information submitted by a Web browser and then update the browser’s display accordingly. Even though JavaScript supports both client and server Web programming, we prefer JavaScript at Client side programming since most of the browsers supports it. JavaScript is almost as easy to learn as HTML, and JavaScript statements can be included in HTML documents by enclosing the statements between a pair of scripting tags <SCRIPTS> </SCRIPT>. Here are a few things we can do with JavaScript:

* Validate the contents of a form and make calculations.
* Add scrolling or changing messages to the Browser’s status line.
* Animate images that change when we move the mouse over them.
* Detect the browser in use and display different content for different browsers.
* Detect installed plug-ins and notify the user if a plug-in is required.
* We can do much more with JavaScript, including creating entire application.

**3.1.1.1 JavaScript Vs Java**

JavaScript and Java are entirely different languages. A few of the most glaring differences are: Java applets are generally displayed in a box within the web document; JavaScript can affect any part of the Web document itself.

While JavaScript is best suited to simple applications and adding interactive features to Web pages, Java can be used for incredibly complex applications. There are many other differences but the important thing to remember is that JavaScript and Java are separate languages. They are both useful for different things; in fact they can be used together to combine their advantages.

**Advantages:**

JavaScript can be used for Sever-side and Client-side scripting. It is more flexible than VBScript. JavaScript is the default scripting languages at Client-side since all the browsers supports it.

**3.1.2 Hyper Text Markup Language:**

Hypertext Markup Language (HTML), the languages of the World Wide Web (WWW), allows users to produces Web pages that include text, graphics and pointer to other Web pages (Hyperlinks). HTML is not a programming language but it is an application of ISO Standard 8879, SGML (Standard Generalized Markup Language), but specialized to hypertext and adapted to the Web. The idea behind Hypertext is that instead of reading text in rigid linear structure, we can easily jump from one point to another point. We can navigate through the information based on our interest and preference. A markup language is simply a series of elements, each delimited with special characters that define how text or other items enclosed within the elements should be displayed. Hyperlinks are underlined or emphasized works that load to other documents or some portions of the same document.HTML can be used to display any type of document on the host computer, which can be geographically at a different location. It is a versatile language and can be used on any platform or desktop.

HTML provides tags (special codes) to make the document look attractive. HTML tags are not case-sensitive. Using graphics, fonts, different sizes, color, etc., can enhance the presentation of the document. Anything that is not a tag is part of the document itself. The advantages are:

* A HTML document is small and hence easy to send over the net. It is small because it does not include formatted information.
* HTML is platform independent.
* HTML tags are not case-sensitive.

**3.1.3 Java Database Connectivity:**

JDBC is a Java API for executing SQL statements. (As a point of interest, JDBC is a trademarked name and is not an acronym; nevertheless, JDBC is often thought of as standing for Java Database Connectivity. It consists of a set of classes and interfaces written in the Java programming language. JDBC provides a standard API for tool/database developers and makes it possible to write database applications using a pure Java API. Using JDBC, it is easy to send SQL statements to virtually any relational database. One can write a single program using the JDBC API, and the program will be able to send SQL statements to the appropriate database. The combinations of Java and JDBC lets a programmer write it once and run it anywhere. What Does JDBC Do? Simply put, JDBC makes it possible to do three things. They are:

* Establish a connection with a database.
* Send SQL statements
* Process the results.

**3.1.3.1 JDBC versus ODBC and other APIs**

At this point, Microsoft's ODBC (Open Database Connectivity) API is that probably the most widely used programming interface for accessing relational databases. It offers the ability to connect to almost all databases on almost all platforms. So why not just use ODBC from Java? The answer is that you can use ODBC from Java, but this is best done with the help of JDBC in the form of the JDBC-ODBC Bridge, which we will cover shortly. The question now becomes "Why do you need JDBC?" There are several answers to this question: ODBC is not appropriate for direct use from Java because it uses a C interface. Calls from Java to native C code have a number of drawbacks in the security, implementation, robustness, and automatic portability of applications. A literal translation of the ODBC C API into a Java API would not be desirable. For example, Java has no pointers, and ODBC makes copious use of them, including the notoriously error-prone generic pointer "void \*". You can think of JDBC as ODBC translated into an object-oriented interface that is natural for Java programmers. ODBC is hard to learn. It mixes simple and advanced features together, and it has complex options even for simple queries. JDBC, on the other hand, was designed to keep simple things simple while allowing more advanced capabilities where required. A Java API like JDBC is needed in order to enable a "pure Java" solution. When ODBC is used, the ODBC driver manager and drivers must be manually installed on every client machine. When the JDBC driver is written completely in Java, however, JDBC code is automatically installable, portable, and secure on all Java platforms from network computers to mainframes.

**3.1.4 Two-tier and three-tier Models**

The JDBC API supports both two-tier and three-tier models for database access. In the two-tier model, a Java applet or application talks directly to the database. This requires a JDBC driver that can communicate with the particular database management system being accessed. A user's SQL statements are delivered to the database, and the results of those statements are sent back to the user. The database may be located on another machine to which the user is connected via a network. This is referred to as a client/server configuration, with the user's machine as the client, and the machine housing the database as the server. The network can be an Intranet, which, for example, connects employees within a corporation, or it can be the Internet. In the three-tier model, commands are sent to a "middle tier" of services, which then send SQL statements to the database. The database processes the SQL statements and sends the results back to the middle tier, which then sends them to the user. MIS directors find the three-tier model very attractive because the middle tier makes it possible to maintain control over access and the kinds of updates that can be made to corporate data. Another advantage is that when there is a middle tier, the user can employ an easy-to-use higher-level API which is translated by the middle tier into the appropriate low-level calls. Finally, in many cases the three-tier architecture can provide performance advantages. Until now the middle tier has typically been written in languages such as C or C++, which offer fast performance.

**3.1.4.1 JDBC Driver Types**

* JDBC-ODBC Bridge plus ODBC driver.
* Native-API partly-Java driver.
* JDBC-Net pure Java driver.
* Native-protocol pure Java driver.
* JDBC-ODBC Bridge.

If possible, use a Pure Java JDBC driver instead of the Bridge and an ODBC driver. This completely eliminates the client configuration required by ODBC. It also eliminates the potential that the Java VM could be corrupted by an error in the native code brought in by the Bridge (that is, the Bridge native library, the ODBC driver manager library, the ODBC driver library, and the database client library).

**What Is the JDBC- ODBC Bridge?**

The JDBC-ODBC Bridge is a JDBC driver, which implements JDBC operations by translating them into ODBC operations. To ODBC it appears as a normal application program. The Bridge implements JDBC for any database for which an ODBC driver is available. The Bridge is implemented as the Sun. jdbc. Odbc Java package and contains a native library used to access ODBC. The Bridge is a joint development of Intervolve and Java Soft.

**3.1.5 Java Server Pages (JSP)**

Java server Pages is a simple, yet powerful technology for creating and maintaining dynamic-content web pages. Based on the Java programming language, Java Server Pages offers proven portability, open standards, and mature re-usable component model .The Java Server Pages architecture enables the separation of content generation from content presentation. This separation not eases maintenance headaches; it also allows web team members to focus on their areas of expertise. Now, web page designer can concentrate on layout, and web application designers on programming, with minimal concern about impacting each other’s work.

**3.1.5.1 Features of JSP**

* **Portability:** Java Server Pages files can be run on any web server or web-enabled application server that provides support for them. Dubbed the JSP engine, this support involves recognition, translation, and management of the Java Server Page lifecycle and its interaction components.
* **Components:** It was mentioned earlier that the Java Server Pages architecture can include reusable Java components. The architecture also allows for the embedding of a scripting language directly into the Java Server Pages file. The components current supported include Java Beans, and Servlets.
* **Processing**: A Java Server Pages file is essentially an HTML document with JSP scripting or tags. The Java Server Pages file has a JSP extension to the server as a Java Server Pages file. Before the page is served, the Java Server Pages syntax is parsed and processed into a Servlet on the server side. The Servlet that is generated outputs real content in straight HTML for responding to the client.
* **Access Models**: A Java Server Pages file may be accessed in at least two different ways. A client’s request comes directly into a Java Server Page. In this scenario, suppose the page accesses reusable Java Bean components that perform particular well-defined computations like accessing a database. The result of the Beans computations, called result sets is stored within the Bean as properties. The page uses such Beans to generate dynamic content and present it back to the client. In both of the above cases, the page could also contain any valid Java code. Java Server Pages architecture encourages separation of content from presentation.

**3.1.5.2 Steps in the execution of a JSP Application:**

The client sends a request to the web server for a JSP file by giving the name of the JSP file within the form tag of a HTML page. This request is transferred to the Java Web Server. At the server side Java Web Server receives the request and if it is a request for a JSP file server gives this request to the JSP engine. This Servlet is loaded in the memory and then it is executed and the result is given back to the Java Web Server and then it is transferred back to the result is given back to the Java Web Server and then it is transferred back to the client.

**CHAPTER-4**

**SAMPLE CODE**

**Newlogin.jsp:**

<html>

<head>

<script type = "text/javascript" >

function disableBackButton()

{

window.history.forward(1);

}

//setTimeout("disableBackButton()", 0);

</script>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>JSP Page</title>

<link href="default.css" rel="stylesheet" type="text/css" />

<link href="style.css" rel="stylesheet" type="text/css" />

</head>

<body onload="disableBackButton()">

<center>

<table class="tablestyle" cellpadding="0" cellspacing="0">

<tr>

<td align="center" colspan="2" bgcolor="white">

<div id="header">

<table cellspacing="5"> <tr><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td><img src="main\_logo.jpg" alt="jmage invalid"/></td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td><img src="scrlogo.jpg" alt="image invalid"/></td>

<td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td><img src="emblem.jpg" alt="image invalid"/></td>

</tr>

</table>

</div> <br>

<font style="font-family: sans-serif "><b>WELCOME TO DAILY MATERIAL TRANSACTION REGISTER SYSTEM</b></font>

<br><br>

<form method="get" action="loginhandler.jsp">

<center>

<fieldset style="height:330px;width:400px;border:1px solid green" >

<legend><b>LOGIN</b></legend>

<table border=0 cellspacing=15 cellpadding=10>

<tr>

<td> User Name </td>

<td> : </td>

<td> <input type="text" name="rn"/> </td>

</tr>

<tr>

<td> Password </td>

<td> : </td>

<td> <input type="password" name="pw"/> </td>

</tr>

</table>

<br>

<input type="submit" value="Submit"/>

&nbsp; &nbsp;

<input type="reset" value="Reset"/>

<br> <br>

</fieldset>

</center>

<br><br>

</form>

</td>

</tr>

<tr>

<td colspan="2" id="footer">

<marquee behavior="scroll" width="1000" direction="left"><b> Developed by E-WORLD/Software Training and Development Center/SCRWWO/BZA</b></marquee>

</td> </tr>

</table></center>

</body>

</html>

**Home.jsp**

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dtd">

<%

String uname=(String) session.getAttribute("uname");

if(uname==null)

{

//response.sendRedirect("login.jsp");

}

%>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>JSP Page</title>

<link href="default.css" rel="stylesheet" type="text/css" />

<link href="style.css" rel="stylesheet" type="text/css" />

</head>

<body>

<center>

<table class="tablestyle" cellpadding="0" cellspacing="0">

<tr>

<th>

<td align="center" colspan="2" bgcolor="white">

<div id="header">

<table cellspacing="5">

<tr><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td><img src="main\_logo.jpg" alt="jmage invalid"/></td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td><img src="scrlogo.jpg" alt="image invalid"/></td>

<td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td><img src="emblem.jpg" alt="image invalid"/></td>

</tr>

</table>

</div>

</td>

</tr>

<tr ><td id="hedder1"><a href="home.jsp"></a>&nbsp;&nbsp;<td id="hedder2"><a href="logout.jsp"></a></td></tr>

<tr>

<td valign="top" width="100" height="300" bgcolor="#fdd">

<div>

<ul id="navmenu">

<li><a href="welcome.jsp" target="destination" class="top\_parent" >&nbsp; HOME</a>

</li>

<li><a href="registration1.jsp" target="destination" class="top\_parent" >&nbsp; REGISTRATION</a>

</li>

<li><a href="ledger.jsp" target="destination" class="top\_parent" >&nbsp; LEDGER</a>

</li>

<li><a href="newlogin.jsp" target="\_self" class="top\_parent" >&nbsp; LOGOUT</a>

</li>

</ul>

</div>

</td>

<td valign="top">

<iframe name="destination" style=" border-style:none ; background-color: white" height="450" width="906" align="left" scrolling="yes">

<img src="images/main.jpg" alt="image not found"/>

</iframe>

</td>

</tr>

<tr>

<td colspan="2" id="footer">

<marquee><b>Developed by E-WORLD/Software Training and Development Center/SCRWWO/BZA</b></marquee></td></tr></table></center> </body></html>

**CHAPTER-5**

**DATABASE TABLES**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **FIELD NAME** | **CONSTRAINT** | **TYPE** |
| 1. | MNO | NOTNULL | VARCHAR2(10) |
| 2. | MNAME |  | VARCHAR2(10) |
| 3. | MTYPE |  | VARCHAR2(10) |
| 4. | ADDRESS | NOT NULL | VARCHAR2(10) |

**Table 5.1 Material Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **FIELD NAME** | **CONSTRAINT** | **TYPE** |
| 1. | WHEN1 | NOT NULL | VARCHAR2(10) |
| 2. | CUNO1 | NOT NULL | VARCHAR2(10) |
| 3. | CHNO1 | NOT NULL | VARCHAR2(10) |
| 4. | FTO1 |  | VARCHAR2(10) |
| 5. | DESC1 |  | VARCHAR2(10) |
| 6. | NO1 | NOT NULL | VARCHAR2(10) |
| 7. | WEIGHT1 | NOT NULL | VARCHAR2(10) |
| 8. | REM1 |  | VARCHAR2(10) |
| 9. | STATION1 | NOT NULL | VARCHAR2(10) |

**Table 5.2 Challan1 Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **FIELD NAME** | **CONSTRAINT** | **TYPE** |
| 1. | SNO | NOT NULL | VARCHAR2(10) |
| 2. | MMNAME |  | VARCHAR2(10) |
| 3. | MMTYPE |  | VARCHAR2(10) |
| 4. | QUAN |  | VARCHAR2(10) |
| 5. | ITO |  | VARCHAR2(10) |
| 6. | CUNO | NOT NULL | VARCHAR2(10) |
| 7. | CHALNO | NOT NULL | VARCHAR2(10) |

**Table 5.3 Issue box Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **FIELD NAME** | **CONSTRAINT** | **TYPE** |
| 1. | SNO | NOT NULL | VARCHAR2(10) |
| 2. | MMNAME |  | VARCHAR2(10) |
| 3. | MMTYPE |  | VARCHAR2(10) |
| 4. | QUAN |  | VARCHAR2(10) |
| 5. | RFROM |  | VARCHAR2(10) |
| 6. | CUNO | NOT NULL | VARCHAR2(10) |
| 7. | CHALNO | NOT NULL | VARCHAR2(10) |

**Table 5.4 Receive box Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **FIELD NAME** | **CONSTRAINT** | **TYPE** |
| 1. | USERNAME | NOT NULL | VARCHAR2(10) |
| 2. | PASSWORD | NOT NULL | VARCHAR2(10) |

**Table 5.5 Login Table**

|  |  |  |  |
| --- | --- | --- | --- |
| S.NO | FIELD NAME | CONSTRAINT | TYPE |
| 1. | USERNAME | NOT NULL | VARCHAR2(10) |
| 2. | PASSWORD | NOT NULL | VARCHAR2(10) |
| 3. | NAME |  | VARCHAR2(10) |
| 4. | PFNO | NOT NULL | VARCHAR2(10) |
| 5. | DESIGNATION | NOT NULL | VARCHAR2(10) |
| 6. | DEPT |  | VARCHAR2(10) |
| 7. | PLACE |  | VARCHAR2(10) |

**Table 5.6 Registration Table**

|  |  |  |  |
| --- | --- | --- | --- |
| S.NO | FIELD NAME | CONSTRAINT | TYPE |
| 1. | CHALNO | NOT NULL | VARCHAR2(10) |
| 2. | STATUS INFORMATION |  | VARCHAR2(10) |

**Table 5.7 Status Table**

**CHAPTER-6**

**SCREEN SHOTS**

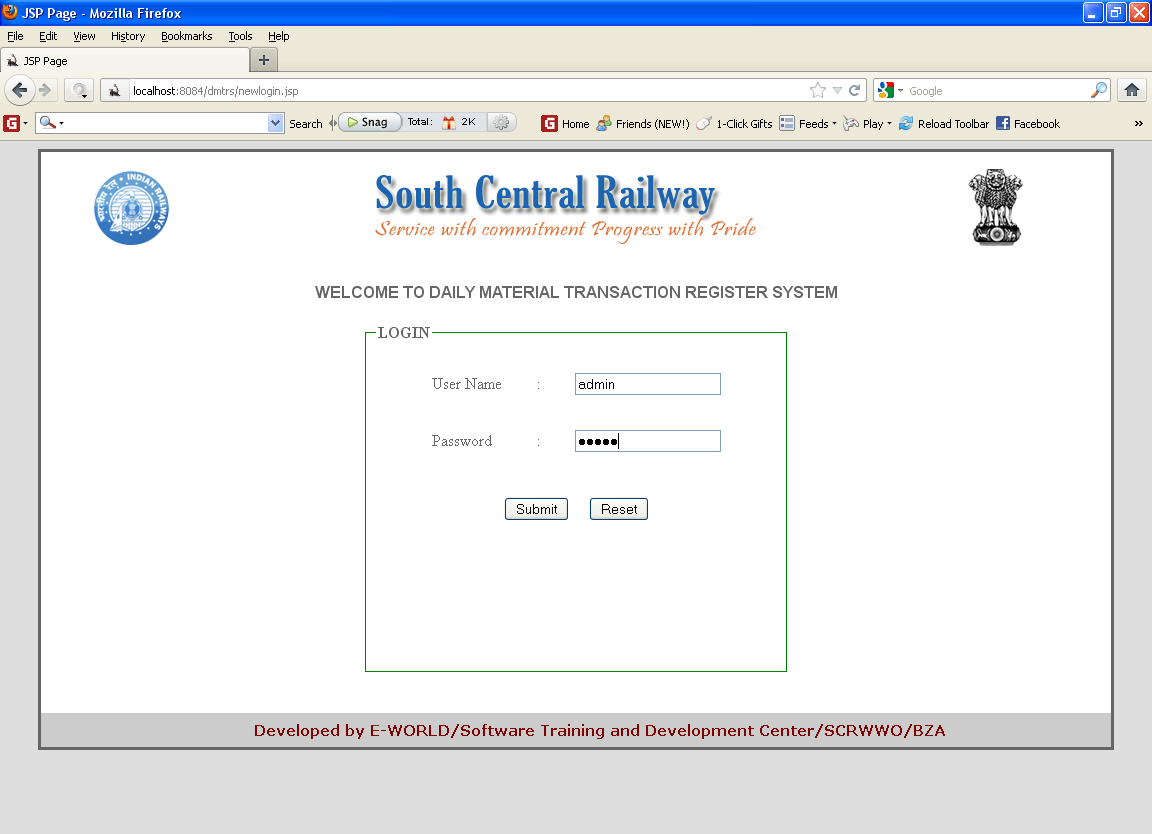
****

Figure 6.1 Login page

**Description:** The above form is the user login page. Here validation takes place**.**

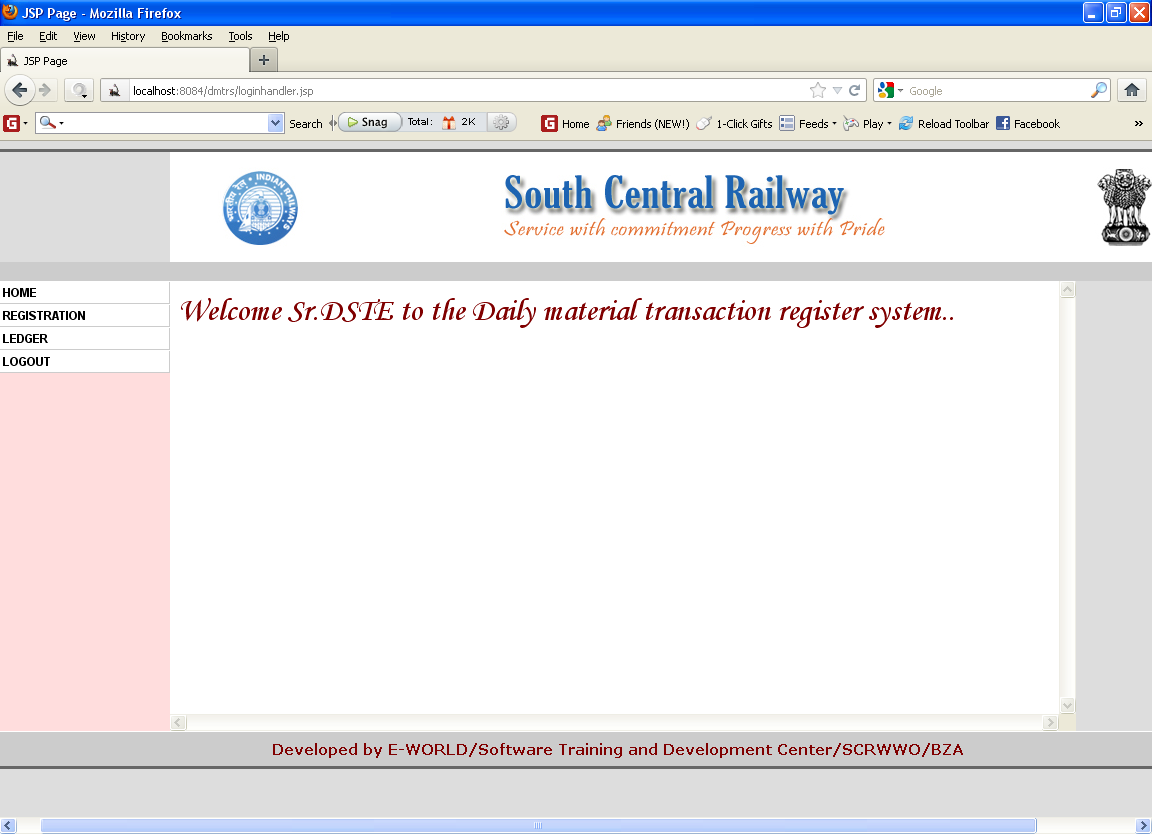
****

Figure 6.2 Home page for the Sr. DSTE

**Description:** The above form is login page of Sr. DSTE. This will come after the verification of username and password.

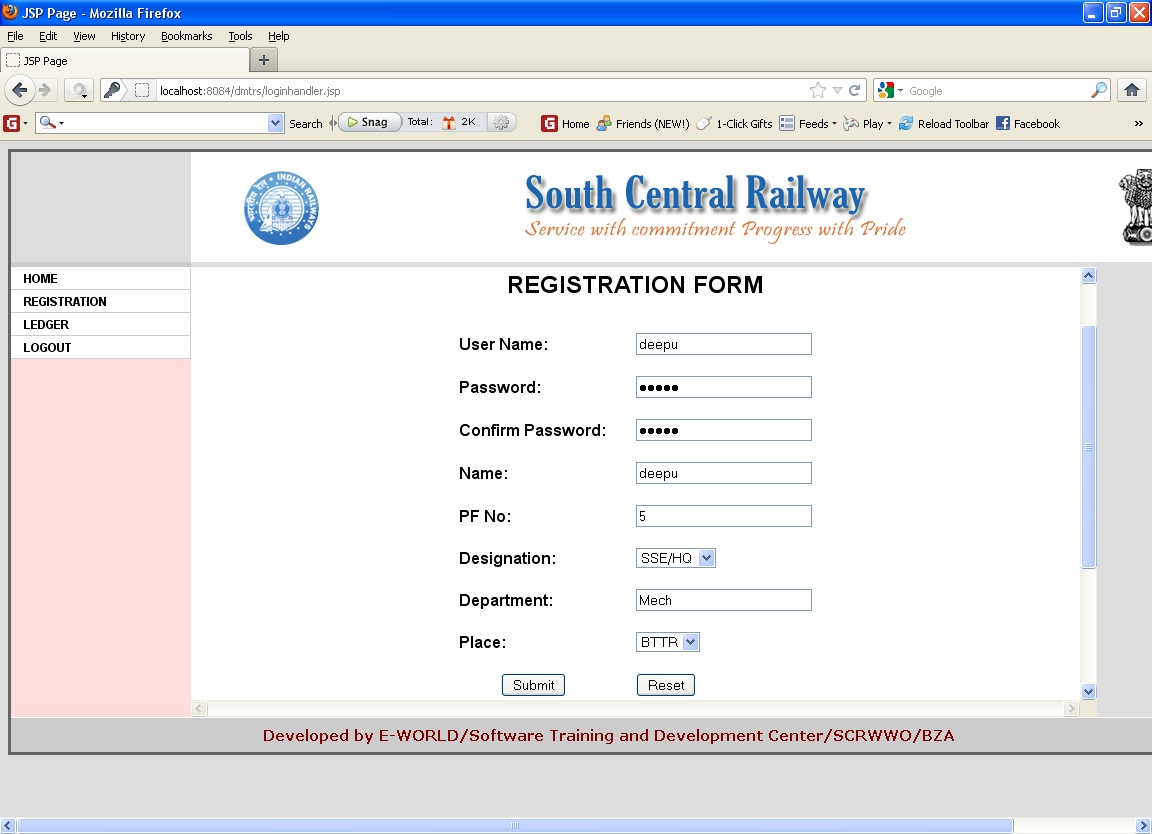


Figure 6.3 Registration form

**Description:** The above form is registration form which is used to register the valid users. After entering all the required values then click on submit button.

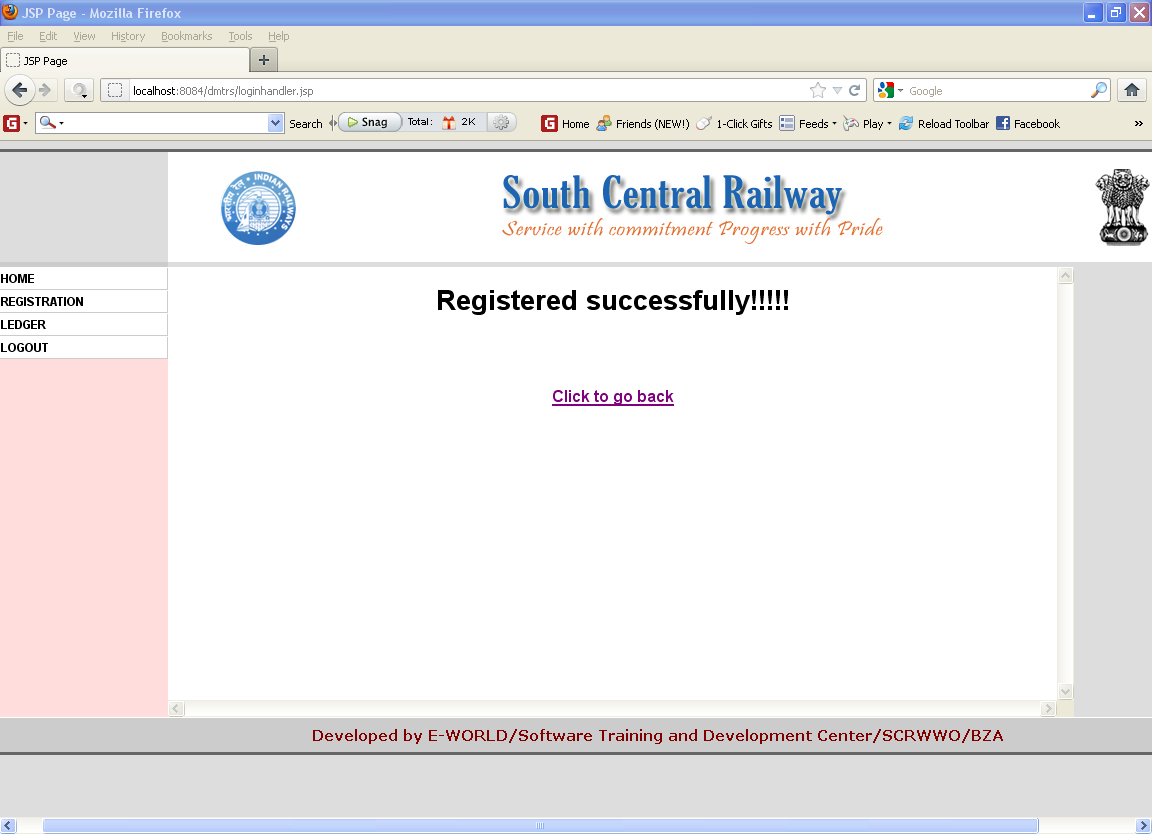
****

Figure 6.4 Page after a successful registration

**Description:** If the validation done is satisfied, registered successfully page will be displayed.

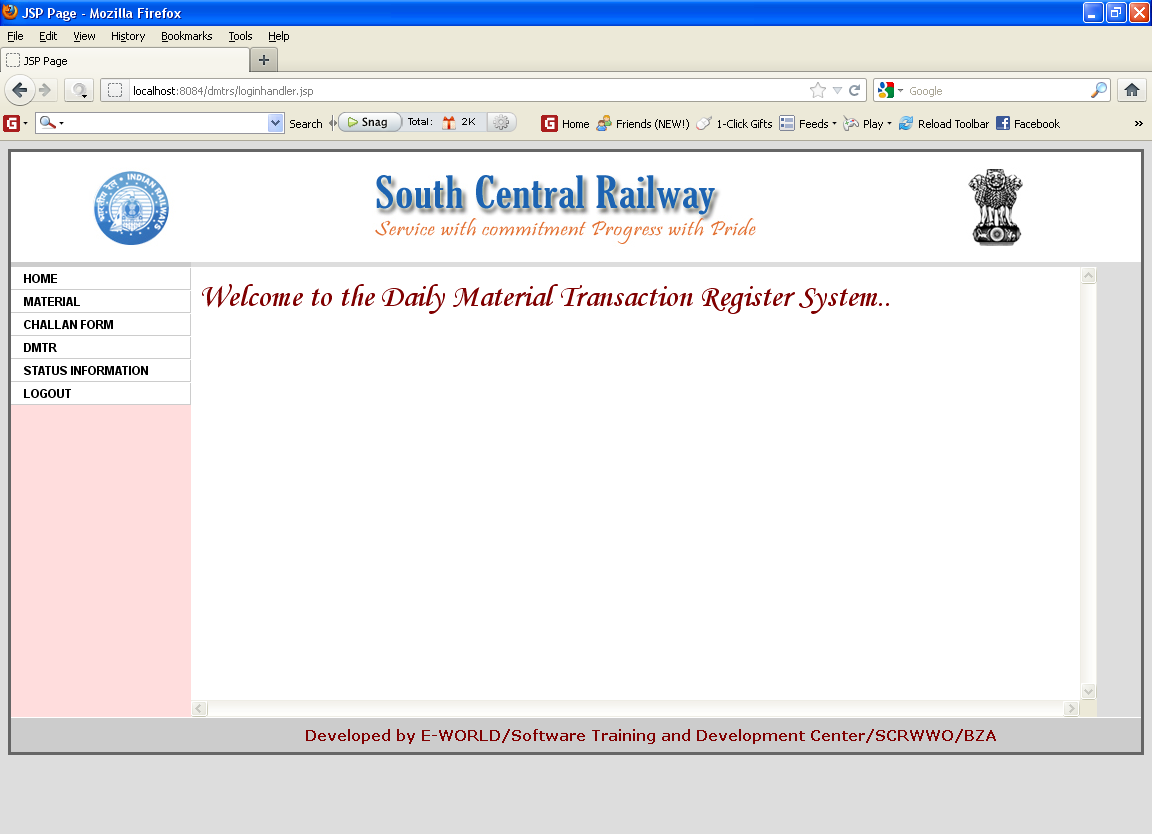


Figure 6.5 Home page for the SSE/HQ

**Description:** The above form is home page of DSTE. This will come after the verification of username and password.

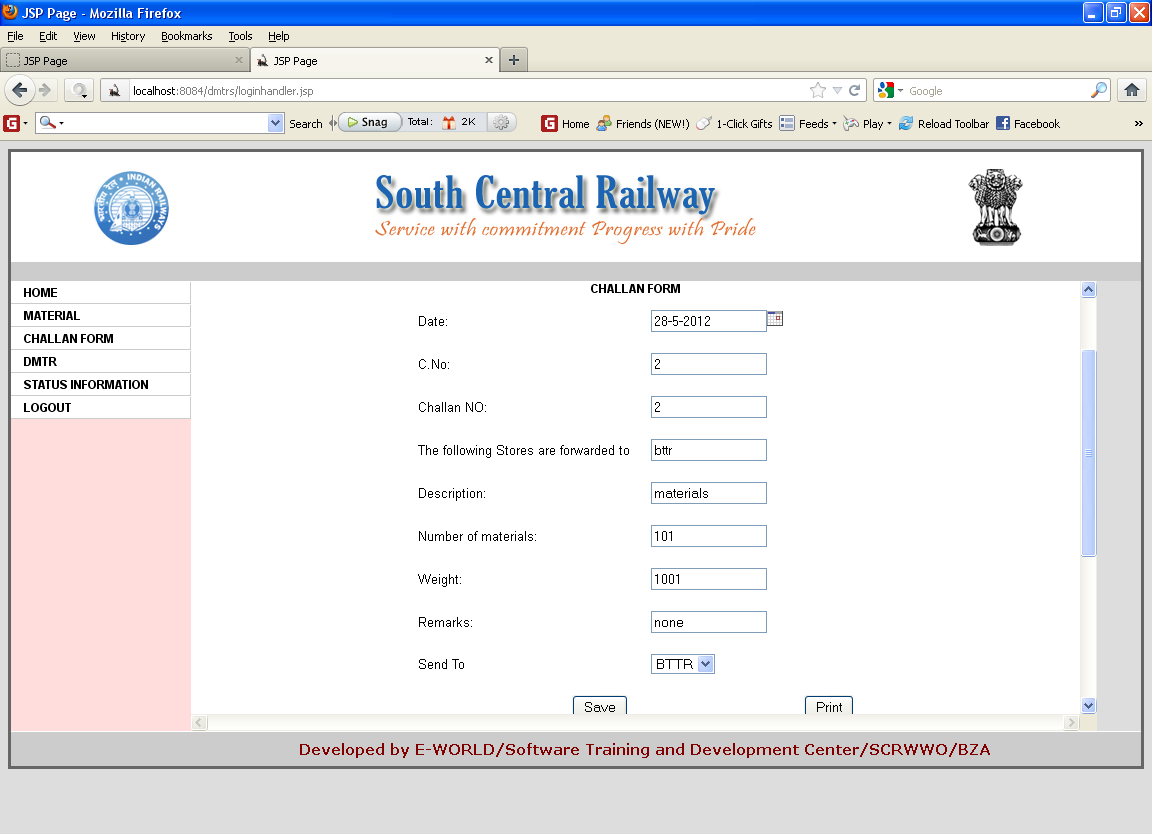
****

Figure 6.6 Page showing the values inserting into the challan form

**Description:** The above form is challan form. This will come when we click the challan form option**.**.

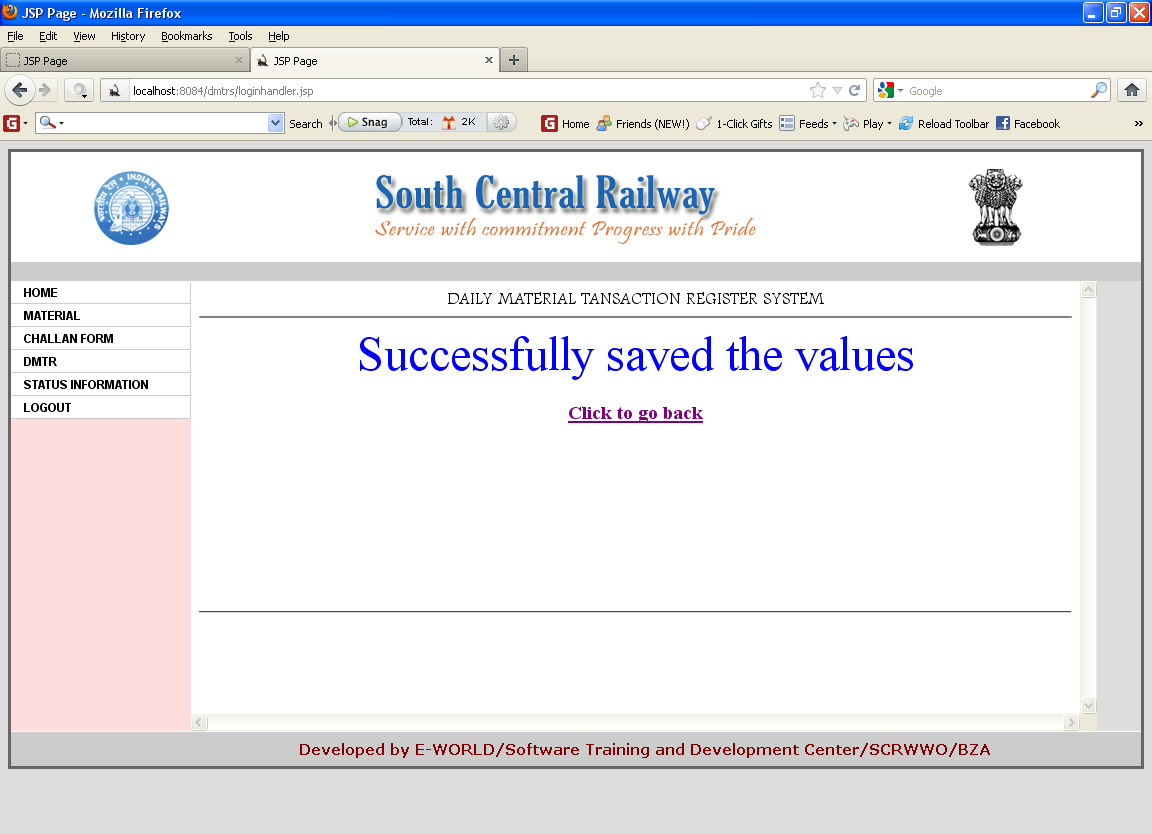


Figure 6.7 Successful insertion of values into challan form

**Description:** After filling this form then click on save button. The above successfully saved page will be appeared.



Figure 6.8 Material issue form

**Description:** This form is the main thing, which issues the material request.

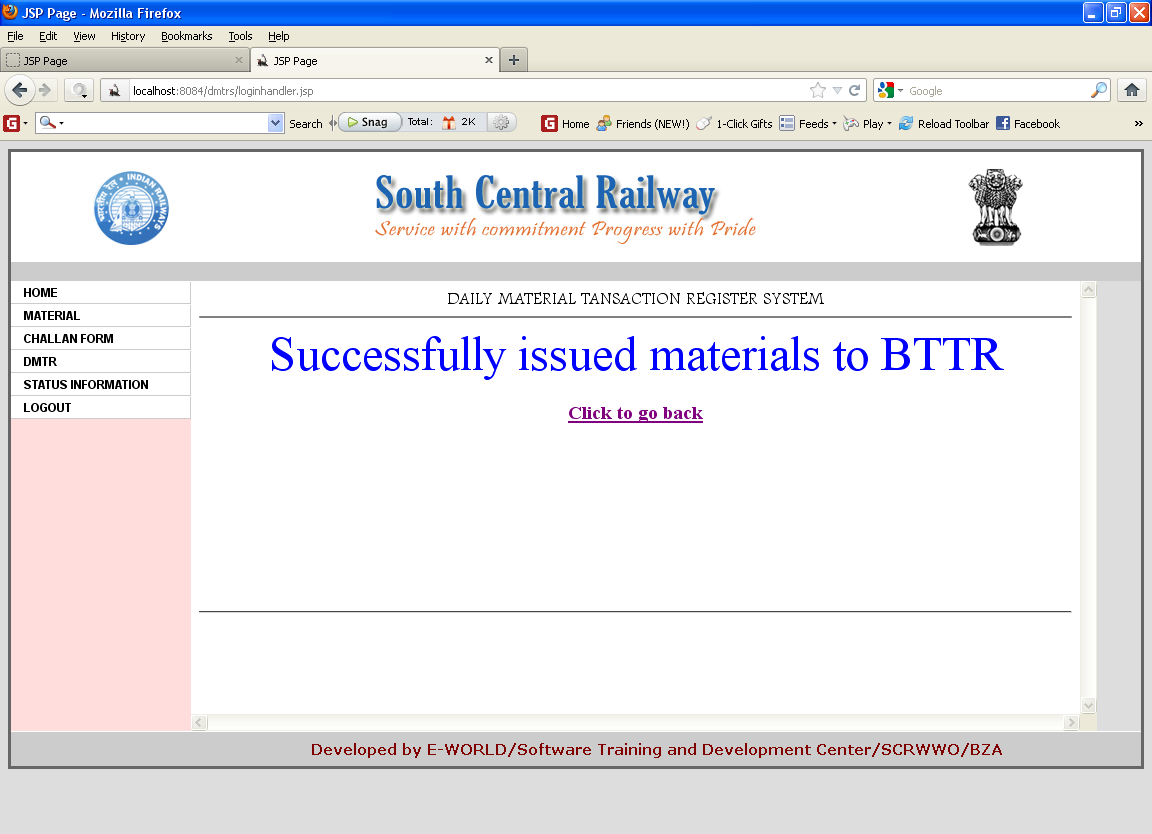


Figure 6.9 Page after successful issue of materials

**Description:** After filling the issue form then click on save button. The above successfully issued materials page will be appeared.

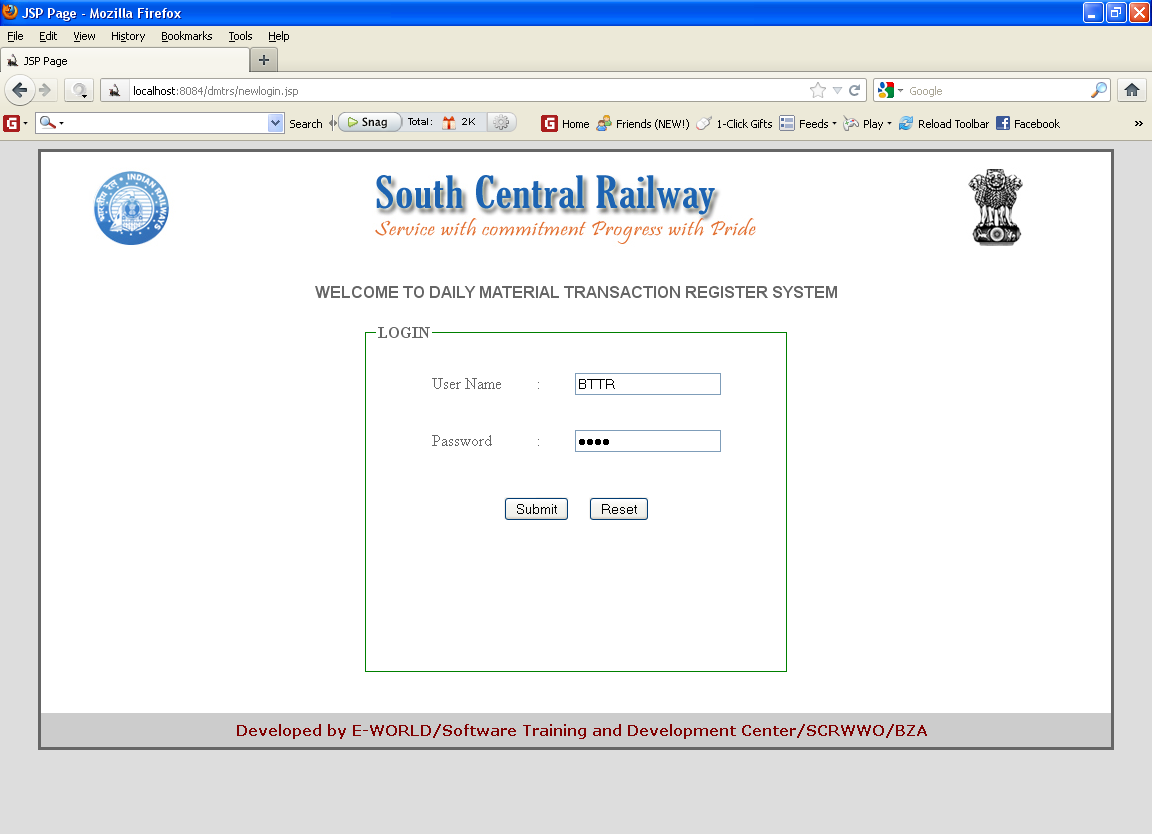


Figure 6.10 Home page for SSE

**Description:**The SSE can also login and verify the mails in the challan inbox.

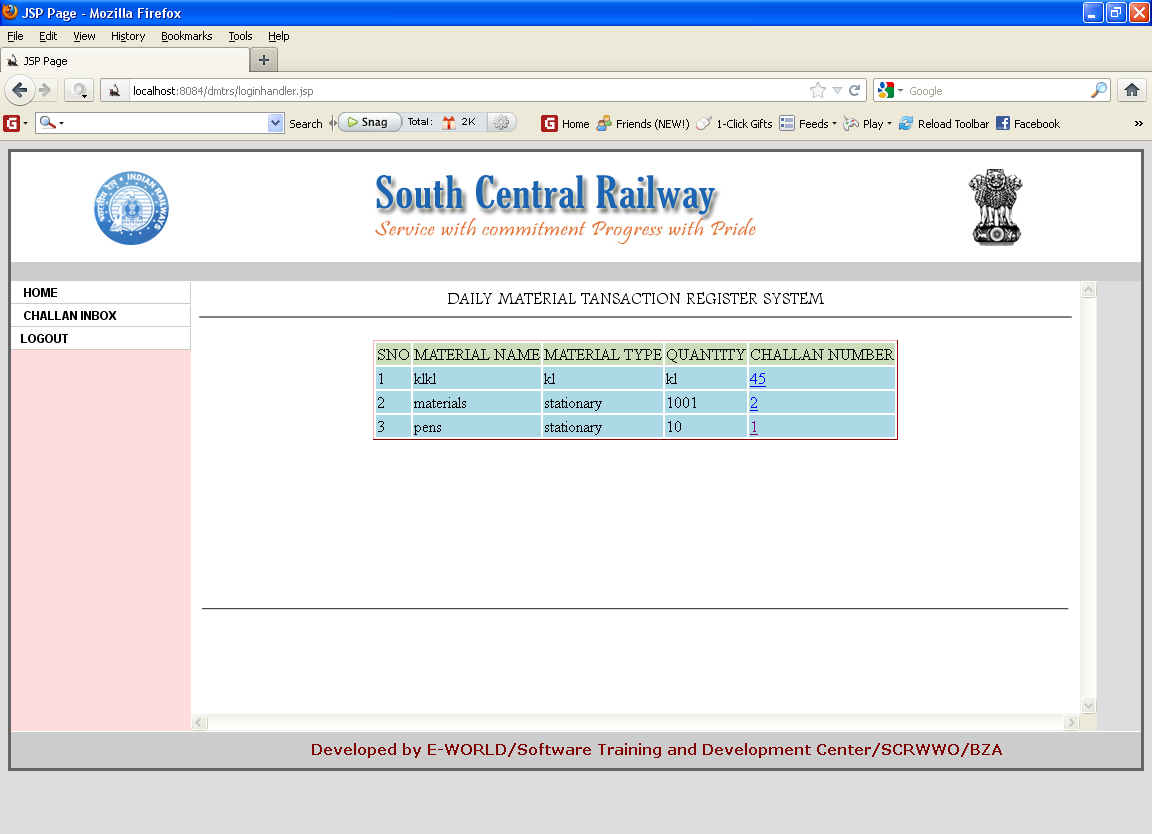


Figure 6.11 Challan inbox for SSE

**Description:** After SSE login, click on challan inbox hyperlink then the issued material request will appear in the form mail inbox.

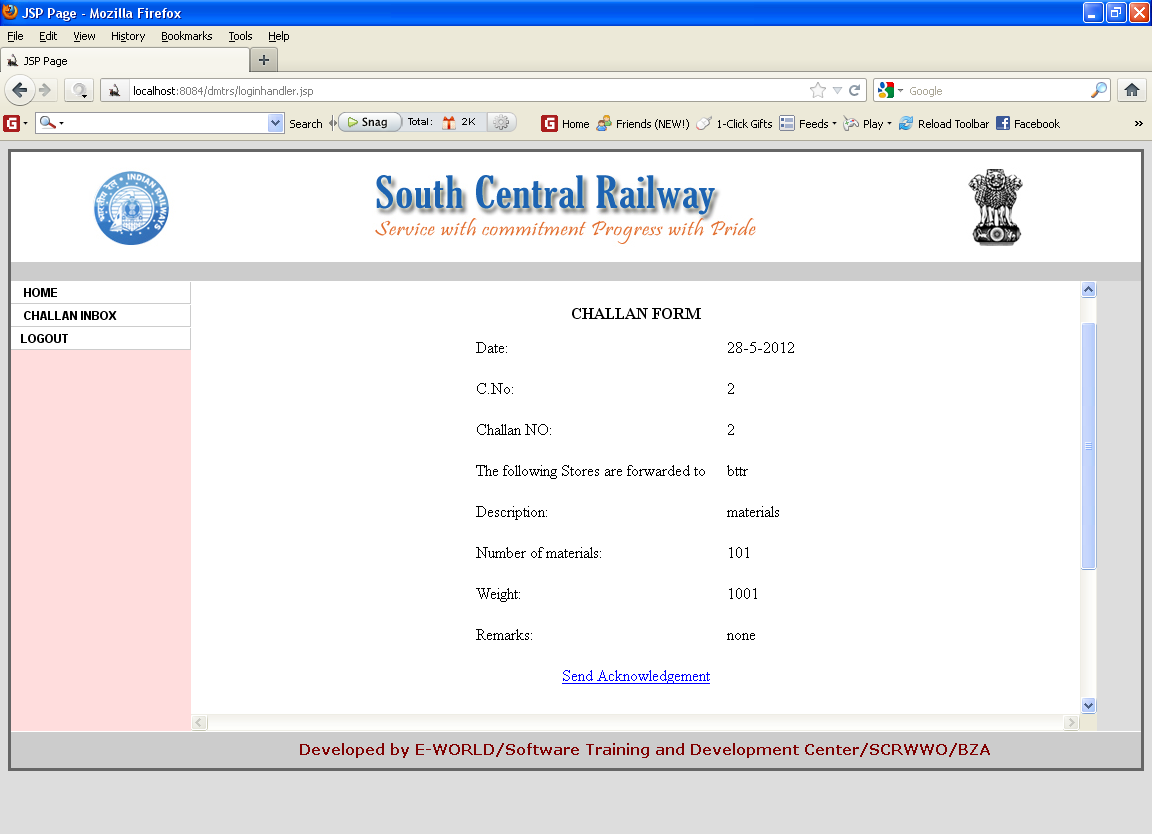


Figure 6.12 Challan details and acknowledgement link

**Description:** By clicking on the challan number hyperlink we can observe a send acknowledgement hyperlink, now click on that hyperlink then the acknowledgement will sent to the status information.

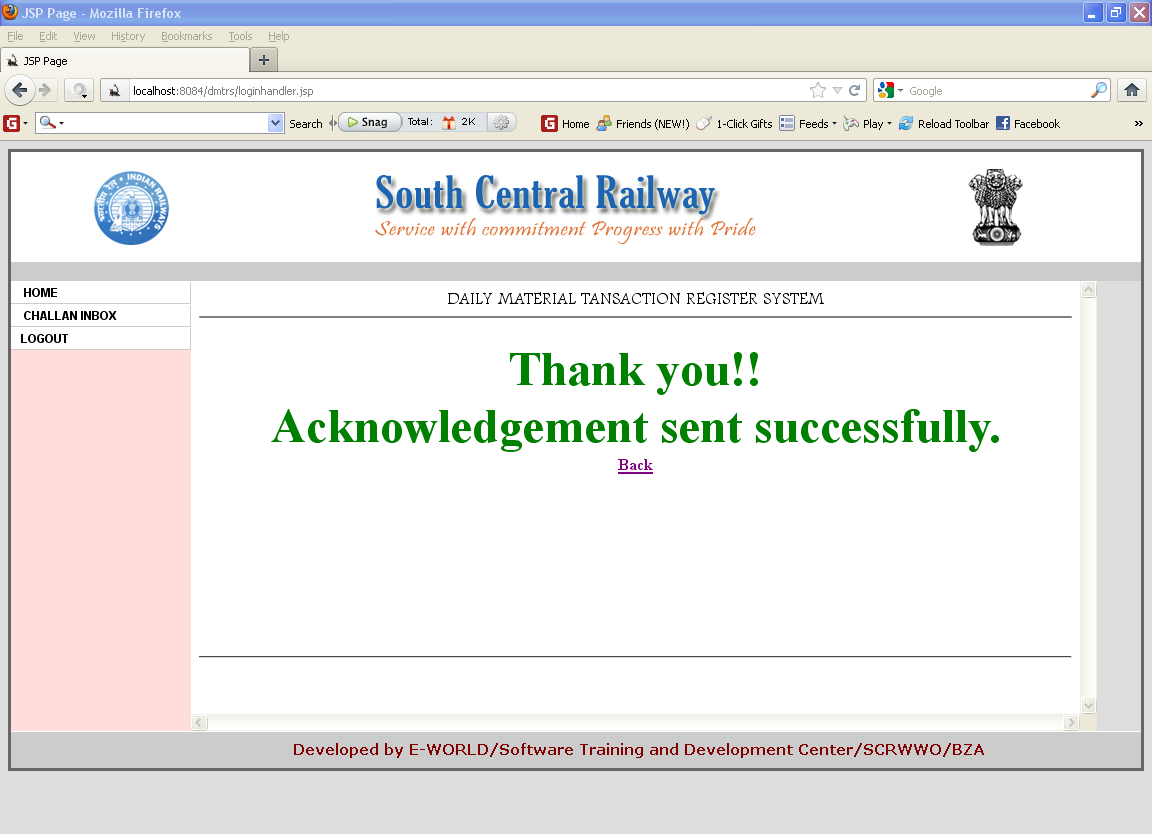


Figure 6.13 Acknowledgement sent successfully

**Desciption:** If the process is valid, then the above acknowledgement sent successfully page will be displayed.

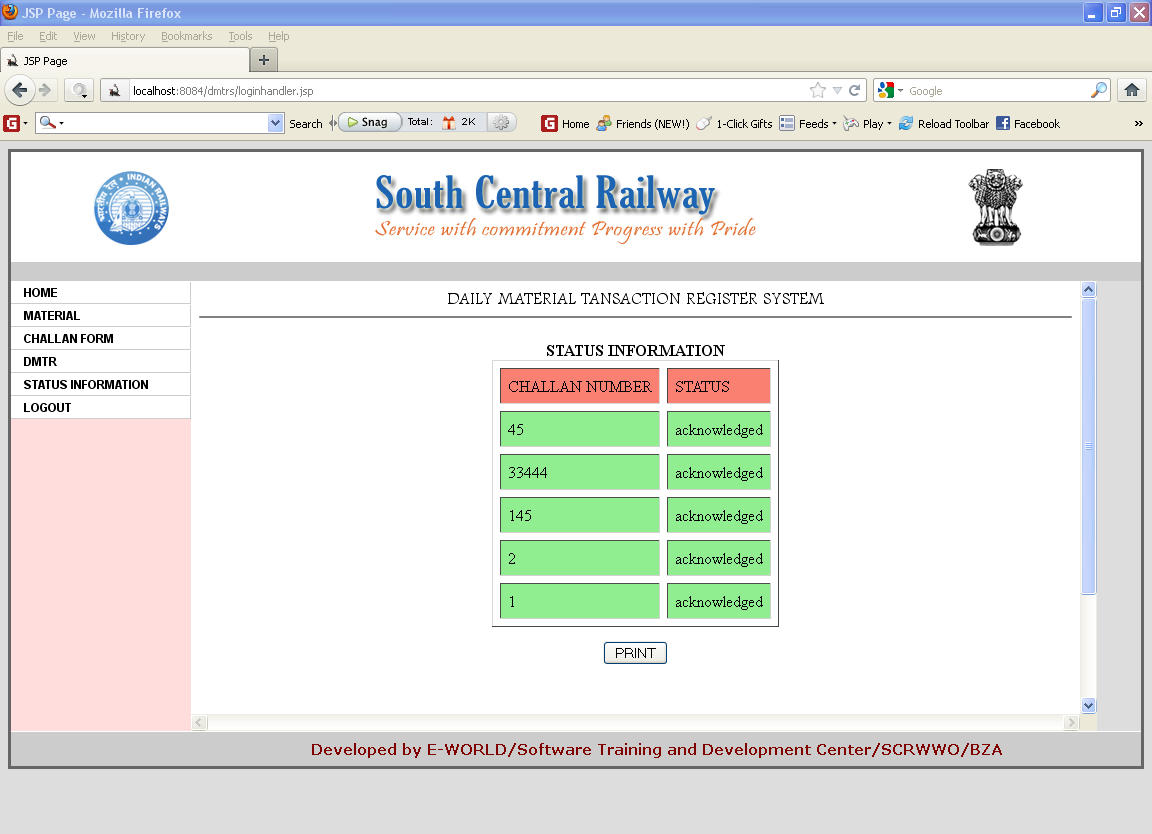
****

Figure 6.14 Status information page in SSE/HQ

**Description:** To verify the acknowledgement send by the SSE, the SSE/HQ will verify the status information.

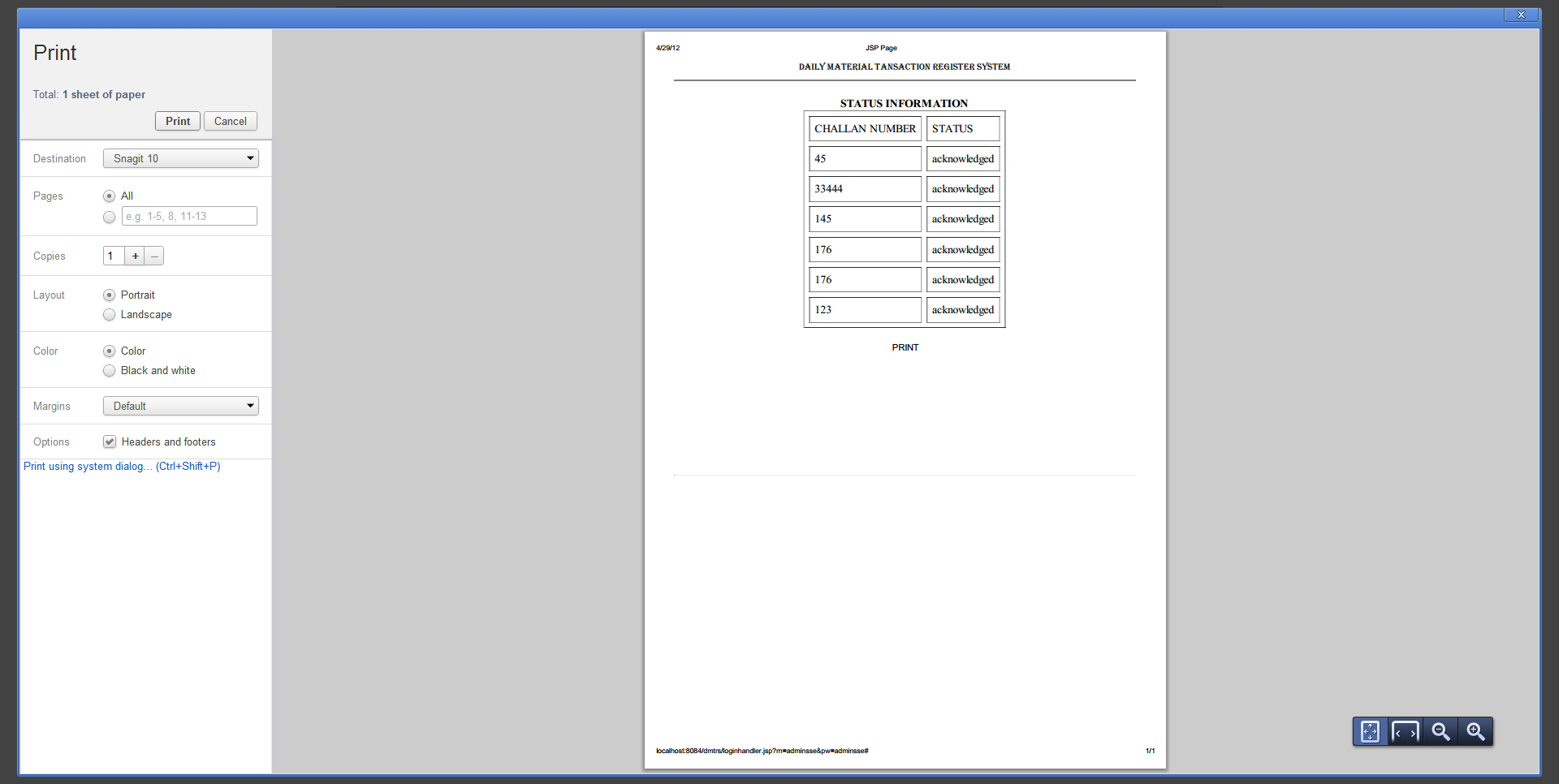
****

Figure 6.15 Print Page For Status Information

**Description:** Finally , this is the page that displays the status information of issued (or) received goods by making use of details such as challan number with status acknowledged.

**CHAPTER-7**

**TESTING**

**System testing:**

Software testing is a critical element of software quality assurance and represents the ultimate reviews of specification, design and coding. Testing represents interesting anomaly for the software. During earlier definition and development phases, it was attempted to build software from an abstract concept to tangible implementation.

The testing phase involves the testing of the developed system using various test data. Preparation of the test data plays a vital role in the system testing. After preparing the test data the system under study was tested using those test data. While testing the system, errors were found and corrected by using the following testing steps and corrections are also noted for future use. Thus, a series of testing is performed for the proposed system, before the system was ready for the implementation. The various types of testing done on the system are:

**Unit Testing:**

Unit testing focuses verification effort on the similar unit of software design the form. This is known as form testing. Since the proposed project has so many forms, the testing is done individually on each form. Using the unit test plans, prepared in design phase of the system development as a guide, important control paths are tested to uncover error within the boundary of the module. In this testing step, each module is found to be working satisfactorily, as regard to the expected output from the module.

**Integrated Testing:**

Data can be lost across an interface, one module can have an adverse effect on another sub-functions, when combined, may not produce the desired major function. Integration is testing, a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with the interface. All modules are combined in the testing step. Then the entire program is tested as a whole.

**Validation Testing:**

At the culmination of the integration testing, the software is completely assembled as a package. Interfacing errors have been uncovered and corrected and final series of software validation testing begins

**Output Testing:**

After performing validation testing, the next steps are output testing of the proposed system, since no system could be useful if it does not produce the desired output in the specified format. The output generated are displayed by the system under consideration or tested by asking the user about the format required by them. Here the output format is considered in two ways. One is the screen and the other is on the printed form.

**Testing Methods:**

Testing is a process of executing a program to find out errors. If testing is conducted successfully, it will uncover all the errors in the software. Any testing can be done basing on two ways:

**White Box Testing:**

It is a test case design method that uses the control structures of the procedural design to design to derive test cases. Using this testing a software Engineer can derive the following test cases:

Exercise all the logical decisions on either true or false sides. Execute all loops at their boundaries and with their operational boundaries. Exercise the internal data structures to assure their validity.

**Black Box Testing:**

It is a test case design method used on the functional requirements of the software. It will help a software engineer to derive sets of input conditions that will exercise all the functional requirements of the program. Block Box testing attempts to find errors in the following categories:

* Incorrect or missing functions
* Interface errors
* Errors in data structures
* Performance errors
* Initialization and termination errors

**User Acceptance Testing:**

User acceptance of a system is the key factor for the success of any system. The system under consideration was tested for user acceptance by constantly keeoing in touch with the perspective system users at the time of developing and making changes whenever required. This is done with regard to the following points:

1. Input screen design

2. Output screen design

3. Online message to guide the user

4. Menu driven system

**Test cases:**

A test case is a software testing document, which consists of event, action, input, output, expected result and actual result. Clinically defined (IEEE 829-1998) a test case is an input and an expected result. This can be as pragmatic as 'for condition x your derived result is y', whereas other test cases described in more detail the input scenario and what results might be expected.

It can occasionally be a series of steps (but often steps are contained in a separate test procedure that can be exercised against multiple test cases, as a matter of economy) but with one expected result or expected outcome. The optional fields are a test case ID, test step or order of execution number, related requirement(s), depth, test category, author, and check boxes for whether the test is automatable and has been automated.

Larger test cases may also contain prerequisite states or steps, and descriptions. A test case should also contain a place for the actual result. These steps can be stored in a word processor document, spreadsheet, database or other common repository. In a database system, you may also be able to see past test results and who generated the results and the system configuration used to generate those results. These past results would usually be stored in a separate table.

The term test script is the combination of a test case, test procedure and test data. Initially the term was derived from the byproduct of work created by automated regression test tools. Today, test scripts can be manual, automated or a combination of both.

The most common term for a collection of test cases is a test suite. The test suite often also contains more detailed instructions or goals for each collection of test cases. It definitely contains a section where the tester identifies the system configuration used during testing. A group of test cases may also contain prerequisite states or steps, and descriptions of the following tests.

**Test case 1:**

|  |  |
| --- | --- |
| ID | 01 |
| Test | To Login into the application |
| Condition | Username and password should match |
| Test inputs | username and password |
| Expected outputs | Login successfully or shows wrong username or password |
| Actual output | Home page will be displayed |
| Pass/fail | Pass |
| Comments | The code to handle this test is written and checking is handled through it. |

Table 7.1 Test case

**Test case 2:**

|  |  |
| --- | --- |
| ID | 02 |
| Test | To add/delete/update the material details |
| Condition | The jsp file should be compiled successfully |
| Test inputs | All fields should be entered according to validations in jsp |
| Expected outputs | Material details are inserted into the database or error message will be displayed |
| Actual output | Details should be entered into the database |
| Pass/fail | Pass |
| Comments | The code to handle this test is written and checking is handled through it. |

Table 7.2 Test case

**Test case 3:**

|  |  |
| --- | --- |
| ID | 03 |
| Test | To generate ledger report |
| Condition | Materials should be issued/received |
| Test inputs | No input |
| Expected outputs | Ledger report will be generated or shows no records message |
| Actual output | Details should be entered into the database |
| Pass/fail | Pass |
| Comments | The code to handle this test is written and checking is handled through it. |

Table 7.3 Test Case

**CHAPTER-8**

**CONCLUSION AND FUTURE SCOPE**

**8.1 CONCLUSION**

In this project, we provide an interface that acts as a good communicating agent for the people working in this department. It acts as a wonderful platform for them, as it provides necessary details about the availability of Material, issue of Material and their delivery status and so on. All the details related to amenity issue will also be clearly indicated. Clear format of challan is designed so that user can clearly understand about the goods that he has received.

**8.2 FUTURE SCOPE**

The project has a wide scope in automation where we can overcome all the disadvantages of the existing system and come up with new system that works effectively and efficiently. In future, we can include a module that supports the automatic generation of SMS without intimation of necessity of goods, so as to implement the project among various divisions in the country.

**CHAPTER-9**

**REFERENCES**

[1] Larne Pekowsky. Java Server Pages, Second Ed. Reading, Boston: Addison Wesley, 2003.

[2] Ivan Bayross. SQL, PL/SQL Programming language for Oracle, second Ed. Reading, East Delhi.

[3] Robert. D. Nero. Java Server Pages Programming: A visual Professional guide for Design

[4] William. S. Robert. Database Management System.

[5] Roger. S. Pressman. Software Engineering

[6] Grady Booch .Object Oriented Analysis and Design with Applications (2nd Edition)