**AUTOMATED TELLER MACHINE SYSTEMS**

**PROBLEM STATEMENT:**

To develop software based **“ATM SYSTEMS”** for supporting in transactions of cash using software rational rose with various UML diagrams.

**PROBLEM DESCRIPTION:**

The system mainly used by the bank clients.  When a client comes to ATM center to update and delete their account.  It reduces the time consumption and lot of paperwork.  For any single operation it involves numerous references and updating also takes subsequent changes in other places.

**3) SRS:**

**a) INTROUCTION**

1. **PURPOSE**

The purpose of the document is to collect and analyze all assorted ideas that have come up to define the system, its requirements with respect to users. In short, the purpose of this SRS document is to provide a detailed overview of our software product, its parameters and goals. This document describes the project's target audience and its user interface, hardware and software requirements. It defines how our client, team and audience see the product and its functionality. Nonetheless, it helps any designer and developer to assist in software delivery lifecycle (SDLC) processes.

1. **SCOPE**

Primarily, the scope pertains to the ATM system. It focuses on cash transactions and helps them to get cash by providing money. The system is used all ATM. This system is used because it is user friendly. This system can be used with the minimum requirement specified. So this does not occupy more space.

1. **DEFINITION,ACRONYMS AND ABBREVIATIONS**

ATM- Automated Teller Machine

SRS- Software Requirement Specification

GUI- Graphical User Interface

HTML-Hyper Text Markup Language

SQL-Structured Query Language

JDBC-Java Database Connectivity

1. **SYSTEM REQUIREMENT SPECIFICATION**

**HARDWARE**:

PROCESSOR: PENTIUM IV 2.6 GHZ

RAM: 512 MB DD RAM

MONITOR: 15’’ COLOR

HARD DISK: 250GB

CDDRIVE: LG52X

KEYBOARD: SATNDARD 102 KEYS

MOUSE: OPTICAL MOUSE

**SOFTWARE:**

FRONT END: HTML

BACK END: MY SQL

OPERATING SYSYEM: WINDOWS XP, WINDOWS 8, WINDOWS 10.

**b)GENERAL DESCRIPTION**

The proposed system that is being developed is user friendly system. The processing speed is very high when compared to the existing system. The space occupied by the proposed system in the memory is also very less. Here, cash is just transferred IN and OUT by Deposit and Withdrawal.

**c)SPECIFIC REQUIREMENTS**

1. **EXTERNAL INTERFACE REQUIREMENTS:**

* **USER INTERFACE REQUIREMENTS:**

The user interface (UI) is everything designed into an information device with which a human being may interact -- including display screen, keyboard, mouse, light pen, the appearance of a desktop, illuminated characters, help messages, and how an application program or a Web site invites interaction and responds to it. In early computers, there was very little user interface except for a few buttons at an operator's console. The user interface was largely in the form of punched card input and report output. The user interface can arguably include the total "user experience," which may include the aesthetic appearance of the device, response time, and the content that is presented to the user within the context of the user interface.

1. Insert Card - The Insert Card helps to get into the ATM Systems.
2. Enter Pin -The Enter Pin is mandatory as that is a protected password for the customer and with that password only he can make correct transactions. Each module is clicked the respective module opens.
3. Choose your Transaction -In this module the customer can choose his transactions depending on whether he wants to deposit or withdraw or check his balance.
4. Deposit module- This Deposit module is for customers to deposit cash into the ATM to their account in certain bank
5. Withdrawal module- This Withdrawal module is used to withdraw/remove cash from the ATM and from their account.
6. Check Balance Module- This module is specifically used to check balance of accounts of the customers.

* **HARDWARE INTERFACE REQUIREMENTS**

|  |  |
| --- | --- |
| Ram | 1GB |
| Hard Disk | 160GB |
| Processor | Pentium-4 |
| Operating System | Win XP/2000/98 |
| Mother Board | Intel d856 |
| Keyboard | Microsoft keyboard |
| Mouse | Microsoft optical |
|  |  |
| Monitor | LG 17”flatiron CRT |
|  |  |
|  |  |

* **SOFTWARE INTERFACE REQUIREMENTS**

OPERATING SYSTEM: WIN2000/XP

PROGRAMMING PACKAGE: HTML

DATABASE: MY SQL

DESIGING SOFTWARE: RATIONAL ROSE 4.2.0

### **COMMUNICATION INTERFACE**

The MES shall use the HTTP protocol for communication over the internet and for the intranet communication will be through TCP/IP protocol suite.

1. **FUNCTIONAL REQUIREMENTS:**

**USER LOGIN:**

The login module helps to get into the ATM Systems. The insert card and enter pin screen pops up as it is mandatory in any ATM to enter their protected pin in order to access their account details and make transactions.

**SELECTION MODULE:**

The selection module is used to select one of the available options. The available options are deposit, withdraw and checkbalance. Each module is clicked the respective module opens.

**DEPOSIT DETAILS:**

In this Deposit details module, a person makes an deposit into the ATM and that deposit is updated to his account by the server and in the record database.

**WITHDRAWAL DETAILS:**

In this Withdrawal details module, a person makes an withdrawal into the ATM and that withdrawal is updated to his account by the server and in record database.

**CHECK BALANCE DETAILS:**

This module is used to check balance and give the available balance to the customer.

**USE CASES**

**Use Case1: User Login**

The login module helps to get into ATM system. The ATM card is inserted into the ATM and it is asked to enter the protected pin in order to access their account.

**ACTOR:**

Human being

**INPUT:**

The human beings enters the card.

**OUTPUT:**

The user can access the system and directed to selection module.

**Use Case2: Selection**

The selection module is used to select one of the available options. The available options are Deposit, withdraw and Checkbalance. Each module is clicked the respective module opens.

**ACTOR:**

Human being

**INPUT:**

The user can select the respective module.

**OUTPUT:**

The users are directed to selected modules.

**Use Case3: Deposit Details**

In this Deposit details module, a person makes an deposit into the ATM and that deposit is updated to his account by the server and in the record database.

**ACTOR:**

Human being

**INPUT:**

The amount to be Deposited is entered.

**OUTPUT:**

The data of the account is updated and cash is deposited into the ATM and customers account balance is updated depending upon how much cash is deposited.

**Use Case4: Withdrawal Details**

In this Withdrawal details module, a person makes an withdrawal into the ATM and that withdrawal is updated to his account by the server and in record database.

**ACTOR:**

Human being

**INPUT:**

The data is entered into the ATM

**OUTPUT:**

The cash comes out of the ATM and the customer receives the cash instant.

1. **NON FUNCTIONAL REQUIREMENTS:**

* **Security**

The system use SSL (secured socket layer) in all transactions that include any confidential information. The system must automatically log out all after a period of inactivity. The system should not leave any cookies on the ATM containing the user’s password. The system’s back-end servers shall only be accessible to authenticated administrator. Sensitive data will be encrypted before being sent over insecure connections like the internet.

* **Reliability**

The system provides storage of all databases on redundant computers with automatic switchover. The reliability of the overall program depends on the reliability of the separate components. The main pillar of reliability of the system is the backup of the database which is continuously maintained and updated to reflect the most recent changes. Thus the overall stability of the system depends on the stability of container and its underlying operating system.

* **Availability**

The system should be available at all times, meaning the user can access it using a web browser, only restricted by the down time of the server on which the system runs. In case of a of a hardware failure or database corruption, a replacement page will be shown. Also in case of a hardware failure or database corruption, backups of the database should be retrieved from the server and saved by the administrator. Then the service will be restarted. It means 24 X 7 availability.

* **Maintainability**

A commercial database is used for maintaining the database and the application server takes care of the site. In case of a failure, a re-initialization of the program will be done. Also the software design is being done with modularity in mind so that maintainability can be done efficiently.

* **Portability**

The application is HTML and scripting language based. So the end-user part is fully portable and any system using any web browser should be able to use the features of the system, including any hardware platform that is available or will be available in the future. An end-user can use this system on any OS; either it is Windows or Linux. The system shall run on PC, Laptops, and PDA etc.

1. **DESIGN CONSTRAINTS**

The system shall be built using a standard web page development tool that conforms to Microsoft’s GUI standards like HTML, XML etc.

* **Hardware Limitations:**

There must be 64 MB on board memory

* **Control Functions:**

The software must be very user friendly and display appropriate error messages.

* **Interfaces to other applicants**:

Not applicable

* **Parallel operations**:

It must support many users simultaneously.

* **Reliability requirements:**

Data redundancy and use of special/blank characters must be avoided.

* **Safety/security considerations**:

The applicant must be exited always normally.

1. **LOGICAL DATABASE REQUIREMENTS**

Database is a place/container where all the data is stored. The database management system used in the system is MY SQL. MY SQL is an [open-source](https://en.wikipedia.org/wiki/Open-source) [relational database management system](https://en.wikipedia.org/wiki/Relational_database_management_system)(RDBMS).Its name is a combination of "My", the name of co-founder [Michael Widenius](https://en.wikipedia.org/wiki/Michael_Widenius)' daughter, and "[SQL](https://en.wikipedia.org/wiki/SQL)", the abbreviation for [Structured Query Language](https://en.wikipedia.org/wiki/Structured_Query_Language). In a database, we would be grouping only related data together and storing them under one group name called table. This helps in identifying which data is stored and under what name. It reduces the time to search for a particular data in a whole database. The MES must be able to use several data formats according to the data formats that are provided by the data bases of expert systems. A transaction should have all the properties of a data base transaction (Atomicity, Consistency, Isolation, Durability).In this system we have three tables to store the data. Patients table to store the symptoms that have been entered by the human being. Doctor’s table to store doctors details who act as an admin. The treatment table would give remedies for the disease.

1. **OTHER REQUIREMENTS**

NIL

**4) DESIGN AND IMPLEMENTATION:**

**UML:**

The **Unified Modeling Language** is a standard visual modeling language intended to be used for

* modeling business and similar processes,
* analysis, design, and implementation of software-based systems

UML is a common language for business analysts, software architects and developers used to describe, specify, design, and document existing or new business processes, structure and behavior of artifacts of software systems.

UML can be applied to diverse **application domains** (e.g., banking, finance, internet, aerospace, healthcare, etc.) It can be used with all major object and component **software development methods** and for various **implementation platforms** (e.g., J2EE, .NET).

UML is a standard modeling **language**, not a **software development process.**

* provides guidance as to the order of a team’s activities,
* specifies what artifacts should be developed,
* directs the tasks of individual developers and the team as a whole, and
* offers criteria for monitoring and measuring a project’s products and activities.

**TYPES OF UML DIAGRAMS:**

* Use-case diagrams
* Sequence diagrams
* Collaboration diagrams
* Activity diagrams
* Class diagrams
* State diagrams

**USE CASE DIAGRAM:**

**DEFINITON:**

A use case illustrates a unit of functionality provided by the system. The main purpose of the use-case diagram is to help development teams visualize the functional requirements of a system, including the relationship of "actors" to essential processes, as well as the relationships among different use cases. Use-case diagrams generally show groups of use cases -- either all use cases for the complete system, or a breakout of a particular group of use cases with related functionality. A use-case diagram is typically used to communicate the high-level functions of the system and the system's scope.

**DESCRIPTION:**

Actor

Use case

Include

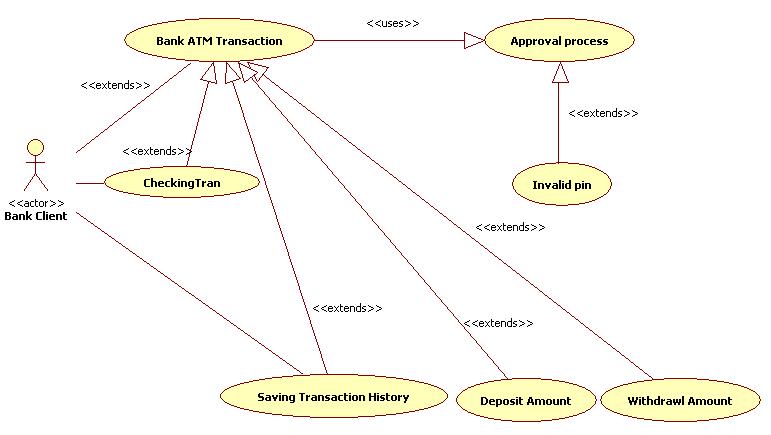
Generalization

Extends

**EXPLANATION:**

In this case, the transactions startup is done which is done by entering the card and then asking for the security pin which when only entered correctly helps in using the account for deposit, withdrawal and for checking the balance.

**USE CASE DIAGRAM FOR ATM SYSTEM**



**SEQUENCE DIAGRAM**

**DEFINITION:**

Sequence diagrams show a detailed flow for a specific use case. They show the calls between the different objects in their sequence. A sequence diagram has two dimensions: the **vertical dimension** shows the sequence of messages/calls in the time order that they occur, the **horizontal dimension** shows the object instances to which the messages are sent. A sequence diagram is very simple to draw. Across the top of your diagram, identify the class instances. If a class instance sends a message to another class instance, draw a line with an open arrowhead pointing to the receiving class instance; place the name of the message/method above the line. Optionally, for important messages, you can draw a dotted line with an arrowhead pointing back to the originating class instance label the return value above the dotted line.

**DESCRIPTION**

Actor

Class instance

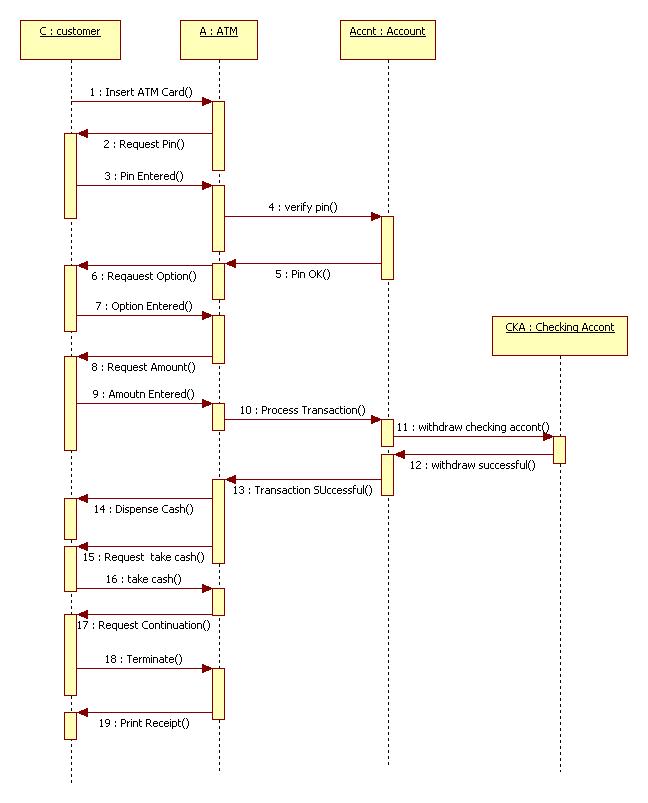
Connection lines

Messages

**EXPLANATION**

The first step is to put the card and enter the pin and chose your desired options, once the option is choosen we should enter the desired amount to deposit or withdraw or to check our account balance.

**SEQUENCE DIAGRAM FOR ATM SYSTEM**



**COLLOBORATION DIAGRAM**

**DEFINITION:**

Collaboration diagrams provide a view of the interactions or structural relationships between objects in the current model. The collaboration diagram emphasizes the relationship between objects whereas sequence diagrams emphasize the sequence of events. Collaboration diagrams contain objects, links, and messages. Use collaboration diagrams as the primary vehicle to describe interactions that express decisions about system behavior.

**DESCRIPTION**

Actor

Class instance

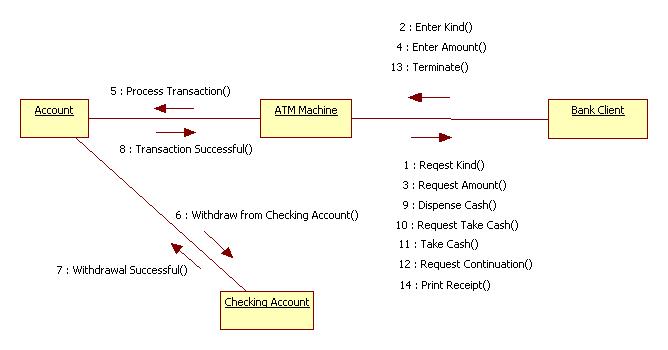
Connection lines

Messages

**EXPLANATION**

The actor here is the client. Then the source client is the ATM and the server used to do the transactions. The operator gives a connection between them. Once the connection get, the clients starts their communication and ends call after finish.

**COLLOBORATION DIAGRAM FOR ATM SYSTEM**



**ACTIVITY DIAGRAM**

**DEFINITION**

Activity diagrams show the procedural flow of control between two or more class objects while processing an activity. Activity diagrams can be used to model higher-level business process at the business unit level, or to model low-level internal class actions. Activity diagrams are best used to model higher-level processes. This is because activity diagrams are "less technical" in appearance, compared to sequence diagrams. The activity is modeled by drawing a rectangle with rounded edges, enclosing the activity's name. Activities can be connected to other activities through transition lines, or to decision points that connect to different activities guarded by conditions of the decision point. Activities that terminate the modeled process are connected to a termination point. Optionally, the activities can be grouped into swim lanes.  THERE CAN BE ONLY ONE START AND ONE STOP.

**DESCRIPTION**

Start

Stop

Swim lanes

Rectangular rounded edged box

Decision box-diamond

Merge

Fork

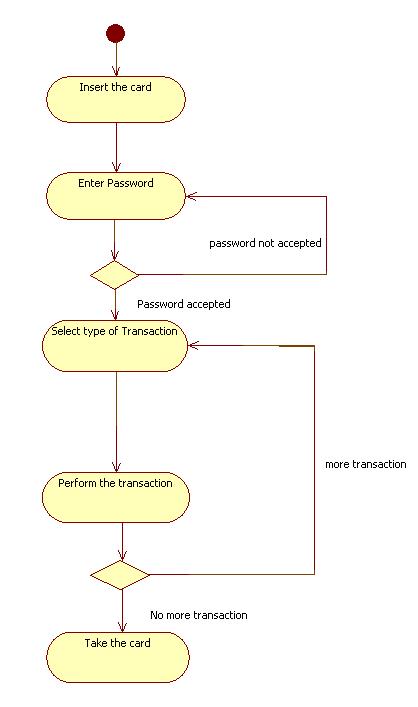
Join

Connection lines

**EXPLANATION**

The first step deposit or to withdraw cash depending upon what the client has chosen. Then the system identifies the transaction. After identifying it gives details of the respected account to the client.

**ACTIVITY DIAGRAM FOR ATM SYSTEM**



**CLASS DIAGRAM**

**DEFINITION:**

The class diagram shows how the different entities relate to each other. It shows the static structures of the system. A class diagram can be used to display logical classes, which are typically the kinds of THINGS, the business people in an organization talk about. Class diagrams can also be used to show implementation classes, which are the things that programmers typically deal with. An implementation class diagram will probably show some of the same classes as the logical class diagram. The implementation class diagram won't be drawn with the same attributes, however, because it will most likely have references to things like Vectors and Hash Maps.

**DESCRIPTION:**

Rectangular box with three horizontal sections

Generalization

Association

Aggregation

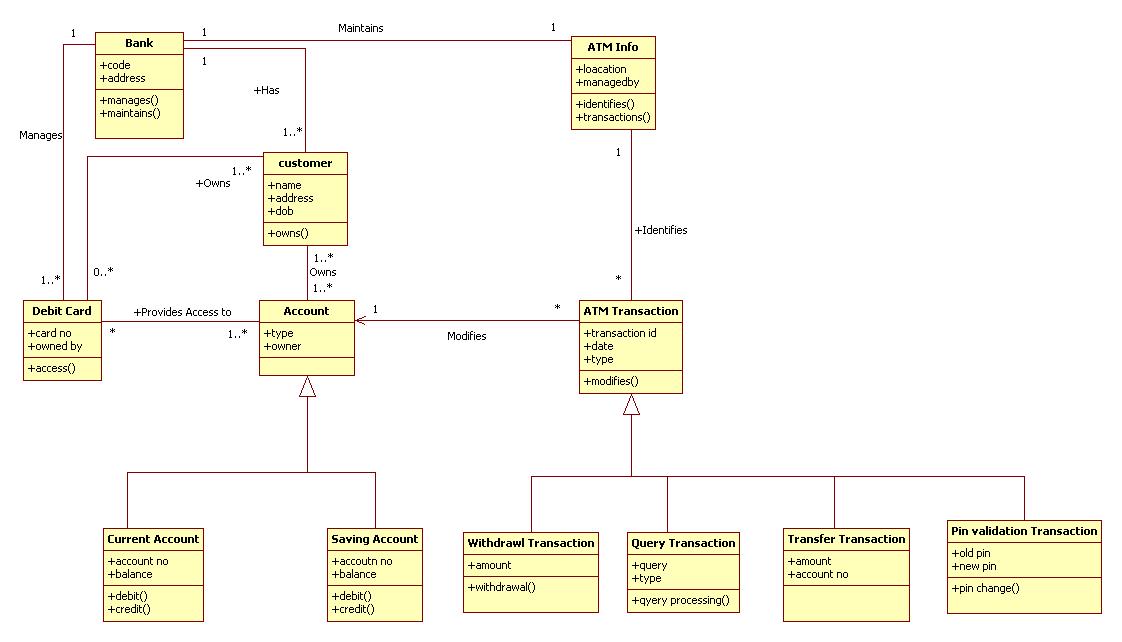
Navigation

Multiplicity

**EXPLANATION:**

The detail of both Client (Human being) and ATM all combine together to give the details. Generalization is show here. Multiplicity is show between Client and the ATM.

**CLASS DIAGRAM FOR ATM SYSTEM**



**STATE DIAGRAM:**

State diagram describes different states of a component in a system. The states are specific to a component/object of a system. A State chart diagram describes a state machine.

**PURPOSE :**

State chart diagram is one of the five UML diagrams used to model dynamic nature of a system. They define different states of an object during its lifetime. And these states are changed by events. State chart diagrams are useful to model reactive systems. Reactive systems can be defined as a system that responds to external or internal events.

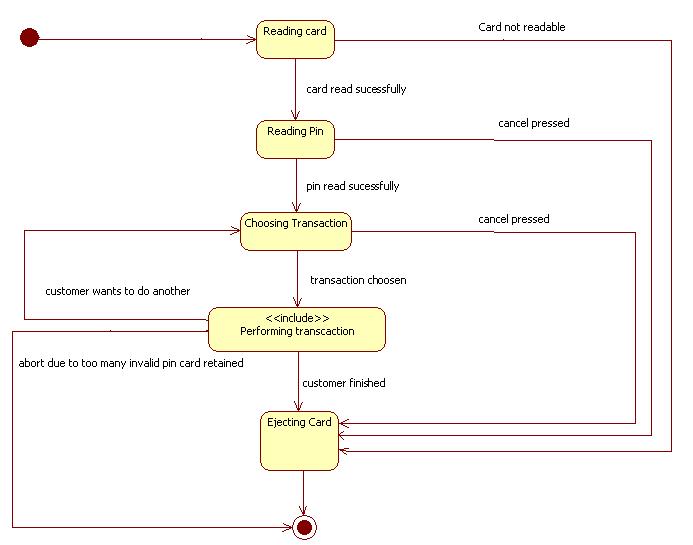
State chart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. So the most important purpose of State chart diagram is to model life time of an object from creation to termination.

State chart diagrams are also used for forward and reverse engineering of a system. But the main purpose is to model reactive system.

Following are the main purposes of using State chart diagrams:

* To model dynamic aspect of a system.
* To model life time of a reactive system.
* To describe different states of an object during its life time.
* Define a state machine to model states of an object.

**STATE DIAGRAM FOR ATM SYSTEM**



**5 ) CODING:**

**FRONT END (HTML&CSS)**

**1.INSSERT CARD SCREEN**

<link rel="stylesheet" type="text/css" href="login1.css">

<form action="action\_page.php">

<div class="imgcontainer">

<img src="login.png" alt="Images" class="images">

</div>

<div class="container">

<label><b>Insert card</b></label>

</div>

<div class="container" style="background-color:#f1f1f1">

</div>

</form>

form {

border: 3px solid #f1f1f1;

}

/\* Full-width inputs \*/

input[type=text], input[type=password] {

width: 100%;

padding: 12px 20px;

margin: 8px 0;

display: inline-block;

border: 1px solid #ccc;

box-sizing: border-box;

}

/\* Set a style for all buttons \*/

button {

background-color: #4CAF50;

color: white;

padding: 14px 20px;

margin: 8px 0;

border: none;

cursor: pointer;

width: 100%;

}

/\* Extra style for the cancel button (red) \*/

.cancelbtn {

width: auto;

padding: 10px 18px;

background-color: #f44336;

}

/\* Center the avatar image inside this container \*/

.imgcontainer {

text-align: center;

margin: 24px 0 12px 0;

}

/\* Avatar image \*/

img.avatar {

width: 40%;

border-radius: 50%;

}

/\* Add padding to containers \*/

.container {

padding: 16px;

}

**2.ENTER PIN MODULE**

</style>

<body>

<h2>WELCOME TO ATM</h2>

<form action="action\_page.asp">

<body background="kameshwari1.jpg">

</div>

<div class="container">

<label><b>ENTER PIN</b></label>

<input type="text" placeholder="Enter pin" name="uname" required>

<div class="container" style="background-color:#f1f1f1">

<button type="button"class=cancelllbtn><a href="Screen3.html">ENTER</a></button>

<button type="button" class="cancelbtn">CANCEL</button>

</div>

</form>

</body>

</html>

**3. SELECTION MODULE**

<!DOCTYPE html>

<html>

<style>

form {

border: 3px solid #f1f1f1;

margin: 0 auto;

width: 400px;

}

h2{

text-align: center;}

body {

background-image:hosp.jpg;

background-size: cover;

background-attachment: fixed;

}

button {

background-color: #4CAF50;

color: white;

padding: 14px 20px;

margin: 8px 0;

border: none;

cursor: pointer;

width: 100%;

}

.cancelbtn {

width: 100%;

padding: 14px 20px;

background-color: #f44336;

}

/\* Change styles for span and cancel button on extra small screens \*/

</style>

<body>

<h2> ATM SYSTEM </h2>

<form action="action\_page.php">

<body background="hosp.jpg">

<div class="container">

<button type="submit"><a href="patient.html">DEPOSIT DETAILS</a></button>;

<button type="submit"><a href="doctor.html">WITHDRAW DETAILS</a></button>;

<buttontype="submit"><ahref="doctor.html">CHECKBALANCEDETAILS</a></button>;

<button type="button" class="cancelbtn">EXIT</button>

</div></form></body></html>

**4.DEPOSIT DETAILS:**

<html>

<style>

form {

border: 20px solid #f1f1f1;

margin: 0 auto;

width: 400px;

}

h2{

text-align: center;

}

body {

background-image:kameshwari1.jpg;

background-size: cover;

background-attachment: fixed;

}

input[type=text], input[type=password]{

width: 100%;

padding: 10px 10px;

display: inline-block;

border: 10px solid #ccc;

box-sizing: border-box;

}

button {

background-color: #4CAF50;

color: white;

padding: 14px 20px;

margin: 8px 0;

border: none;

cursor: pointer;

width: 100%;

}

.cancelbtn {

width: auto;

padding: 10px 18px;

background-color: #f44336;}

.cancellbtn {

width: auto;

padding: 10px 18px;

background-color: #0000FF;}

.cancelllbtn {

width: auto;

padding: 10px 18px;

background-color:#00FF00 ;}

}

}

</style>

<body>

<h2>DEPOSIT TRANSACTION</h2>

<form action="action\_page.asp">

<body background="kameshwari1.jpg">

</div>

<div class="container">

<label><b>ENTER AMOUNT</b></label>

<input type="text" placeholder="Enter amount" name="uname" required>

<buttontype="button"class=cancelllbtn><a href="Screen7.html">ENTER</a></button>

<div class="container" style="background-color:#f1f1f1">

<button type="button" class="cancelbtn">CANCEL</button>

</div>

</form>

</body>

</html>

**5.WITHDRAW DETAILS**

<html>

<style>

form {

border: 20px solid #f1f1f1;

margin: 0 auto;

width: 400px;

}

h2{

text-align: center;

}

body {

background-image:kameshwari1.jpg;

background-size: cover;

background-attachment: fixed;

}

input[type=text], input[type=password]{

width: 100%;

padding: 10px 10px;

display: inline-block;

border: 10px solid #ccc;

box-sizing: border-box;

}

button {

background-color: #4CAF50;

color: white;

padding: 14px 20px;

margin: 8px 0;

border: none;

cursor: pointer;

width: 100%;

}

.cancelbtn {

width: auto;

padding: 10px 18px;

background-color: #f44336;}

.cancellbtn {

width: auto;

padding: 10px 18px;

background-color: #0000FF;}

.cancelllbtn {

width: auto;

padding: 10px 18px;

background-color:#00FF00 ;}

}

}

</style>

<body>

<h2>WITHDRAWAL TRANSACTION</h2>

<form action="action\_page.asp">

<body background="kameshwari1.jpg">

</div>

<div class="container">

<label><b>ENTER AMOUNT</b></label>

<input type="text" placeholder="Enter amount" name="uname" required>

<buttontype="button"class=cancelllbtn><a href="Screen8.html">ENTER</a></button>

<div class="container" style="background-color:#f1f1f1">

<button type="button" class="cancelbtn">CANCEL</button>

</div>

</form>

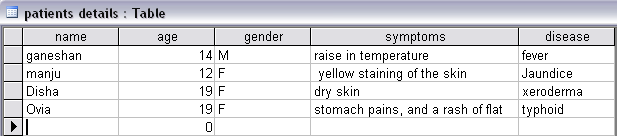
</body>

</html>

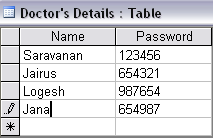
**DATABASE:**

**TABLES:**

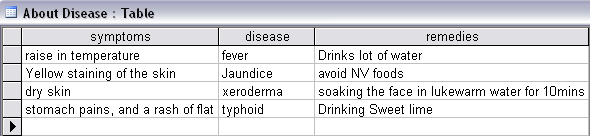
PATIENT DETAILS:



DOCTORS DETAILS:



ABOUT DISEASE:



**JDBC:(jsp)**

**PATIENT MODULE**

<%@ page import ="java.sql.\*" %>

<%@ page import ="javax.sql.\*" %>

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

<!DOCTYPE html>

<%

String a=request.getParameter("pName");

String b=request.getParameter("age");

String c=request.getParameter("gender");

String d=request.getParameter("symptoms");

try{

Class.forName("com.mysql.jdbc.Driver");

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/medical", "root", "guruji");

Statement st=con.createStatement();

int i=st.executeUpdate("insert into patients\_details(pname,age,gender,symptoms)values ('"+a+"','"+b+"','"+c+"','"+d+"')");

}

catch(Exception e){

System.out.print(e);

}

%>

**DOCTOR MODULE**

<%@ page import ="java.sql.\*" %>

<%@ page import ="javax.sql.\*" %>

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

<!DOCTYPE html>

<%

String a=request.getParameter("uname");

String b=request.getParameter("pswd");

try{

Class.forName("com.mysql.jdbc.Driver");

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/medical", "root", "guruji");

Statement st=con.createStatement();

int i=st.executeUpdate("insert into doctors\_details(name,password)values ('"+a+"','"+b+"')");

}

catch(Exception e){

System.out.print(e);

}

response.sendRedirect("expert.html");

//link to html

%>

**TREATMENT MODULE**

<%@ page import="java.sql.\*" %>

<%@ page language="java" import="java.sql.\*"%>

<HTML>

<style>

h2{

text-align: center;

}

body {

background-image:treat.jpg;

background-size: cover;

background-attachment: fixed;

}

table, th, td {

border: 4px solid black;

}

th {

background-color: #4CAF50;

color: white;

}

</style>

<body background="treat.jpg">

<h2>TREATMENT</h2>

<%

String d=request.getParameter("symptoms");

Class.forName("com.mysql.jdbc.Driver");

Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/medical", "root", "guruji");

Statement statement = connection.createStatement() ;

ResultSet resultset =

statement.executeQuery("select \* from disease where symptoms = '" + d + "' " ) ;

%>

<TABLE BORDER="1">

<table align="center">

<TR>

<TH>symptoms</TH>

<TH>disease</TH>

<TH>remedies</TH>

</Tr>

<% while(resultset.next()){ %>

<TR>

<TD> <%= resultset.getString(1) %></td>

<TD> <%= resultset.getString(2) %></TD>

<TD> <%= resultset.getString(3) %></TD>

</Tr>

<% } %>

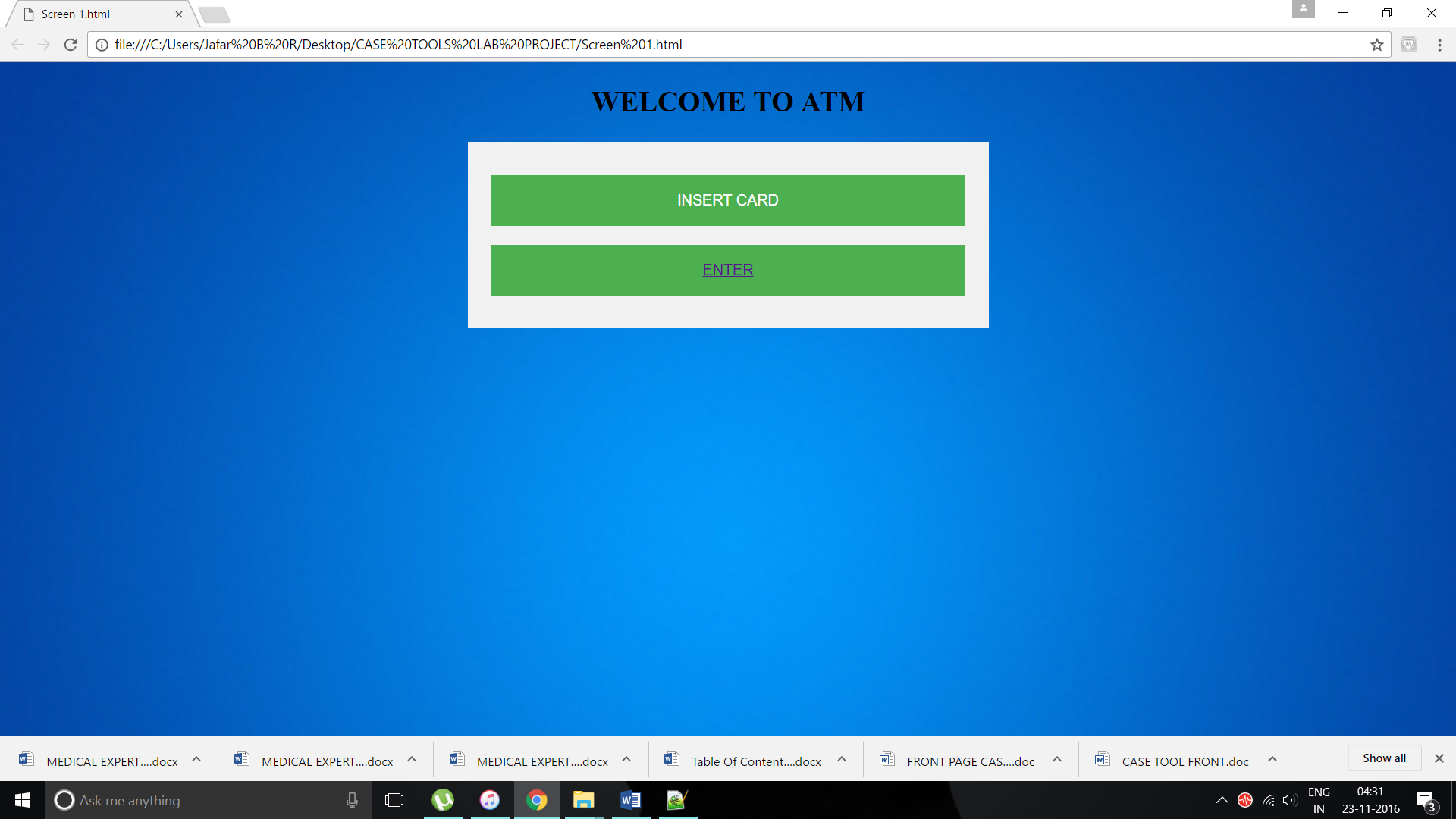
</TABLE>

</BODY>

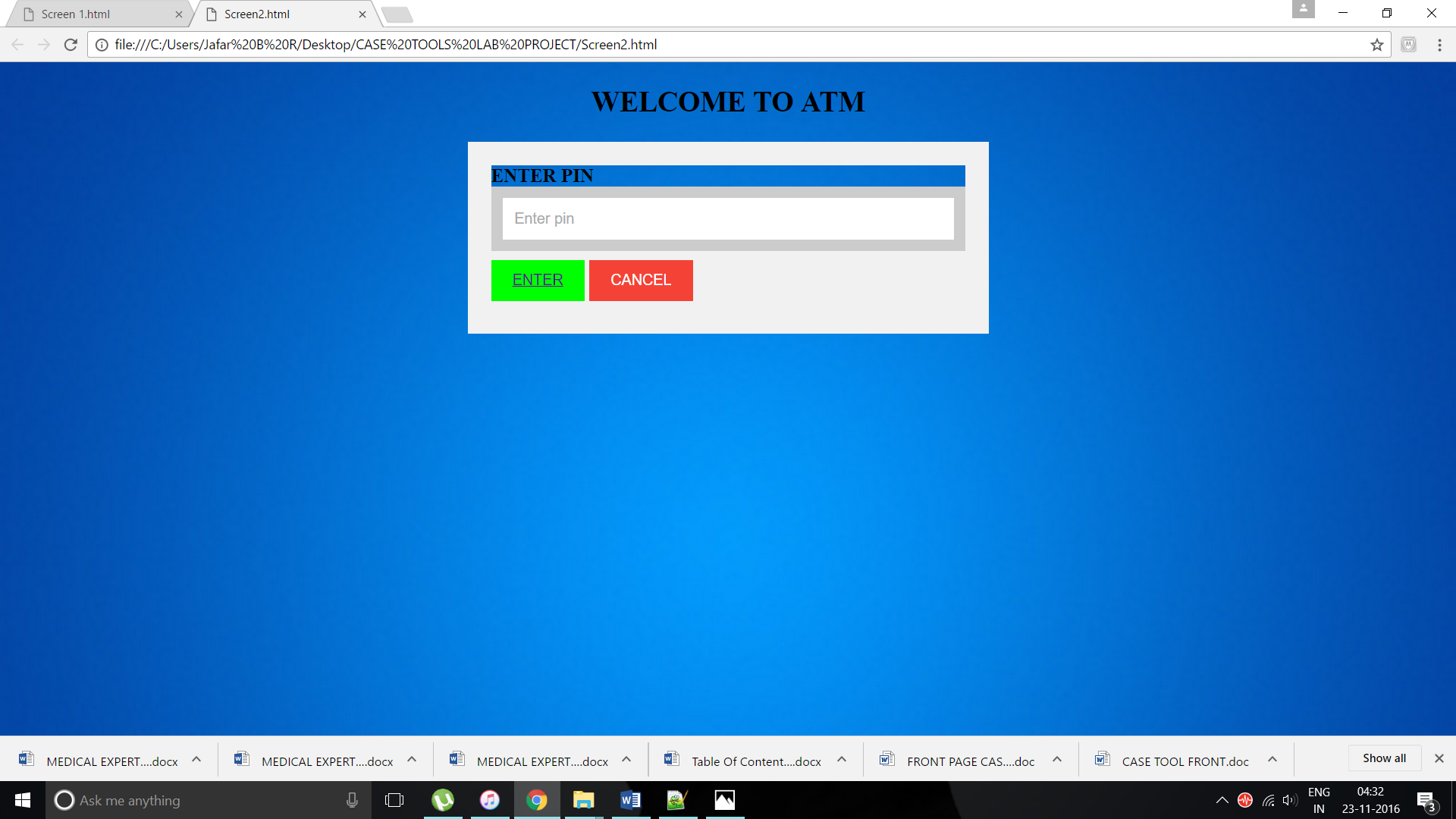
</HTML>

**6.OUTPUT**

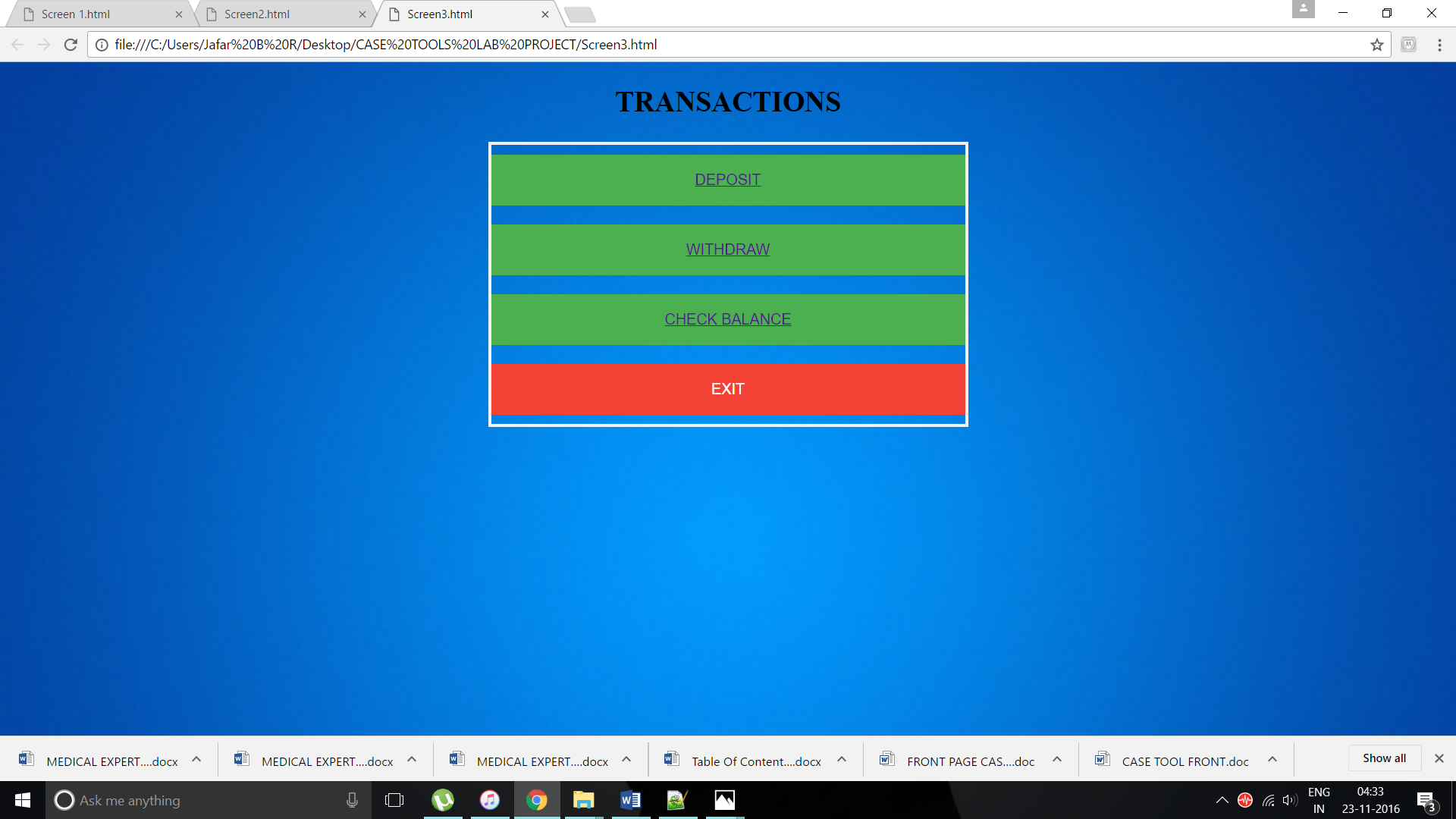
**INSERT CARD MODULE:**



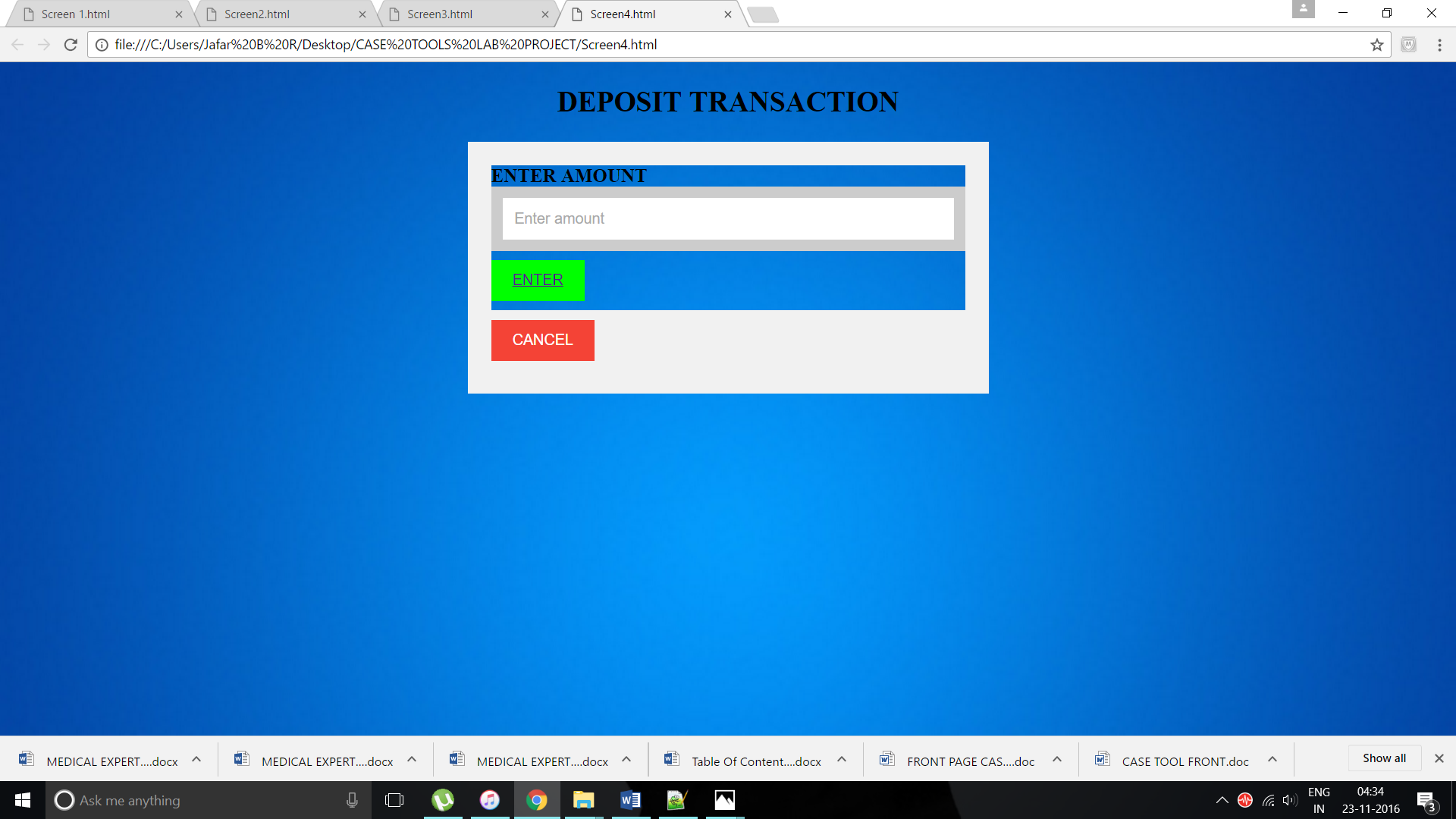
**ENTER PIN MODULE**:



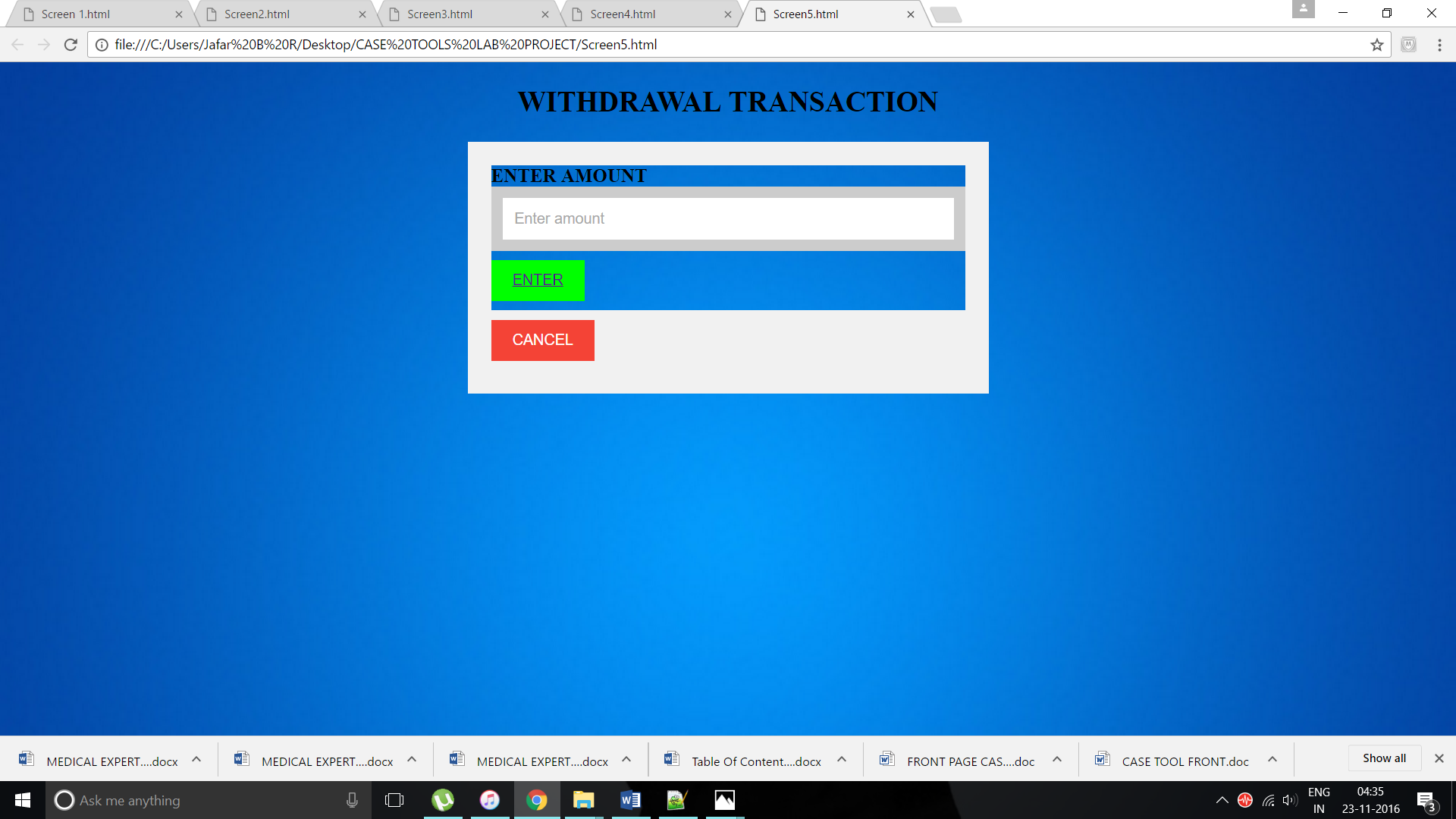
**SELECTION MODULE**:



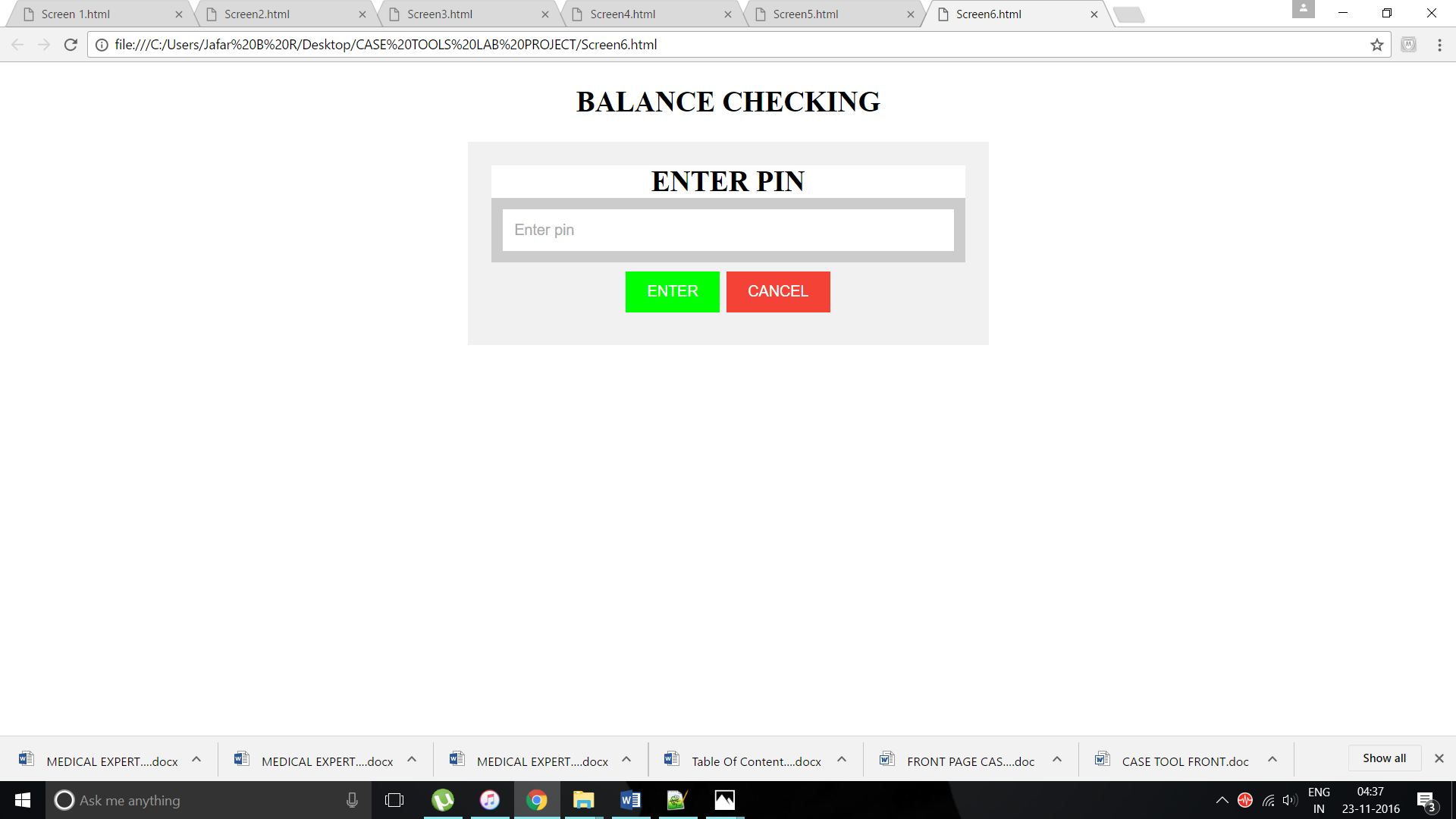
**DEPOSIT MODULE:**



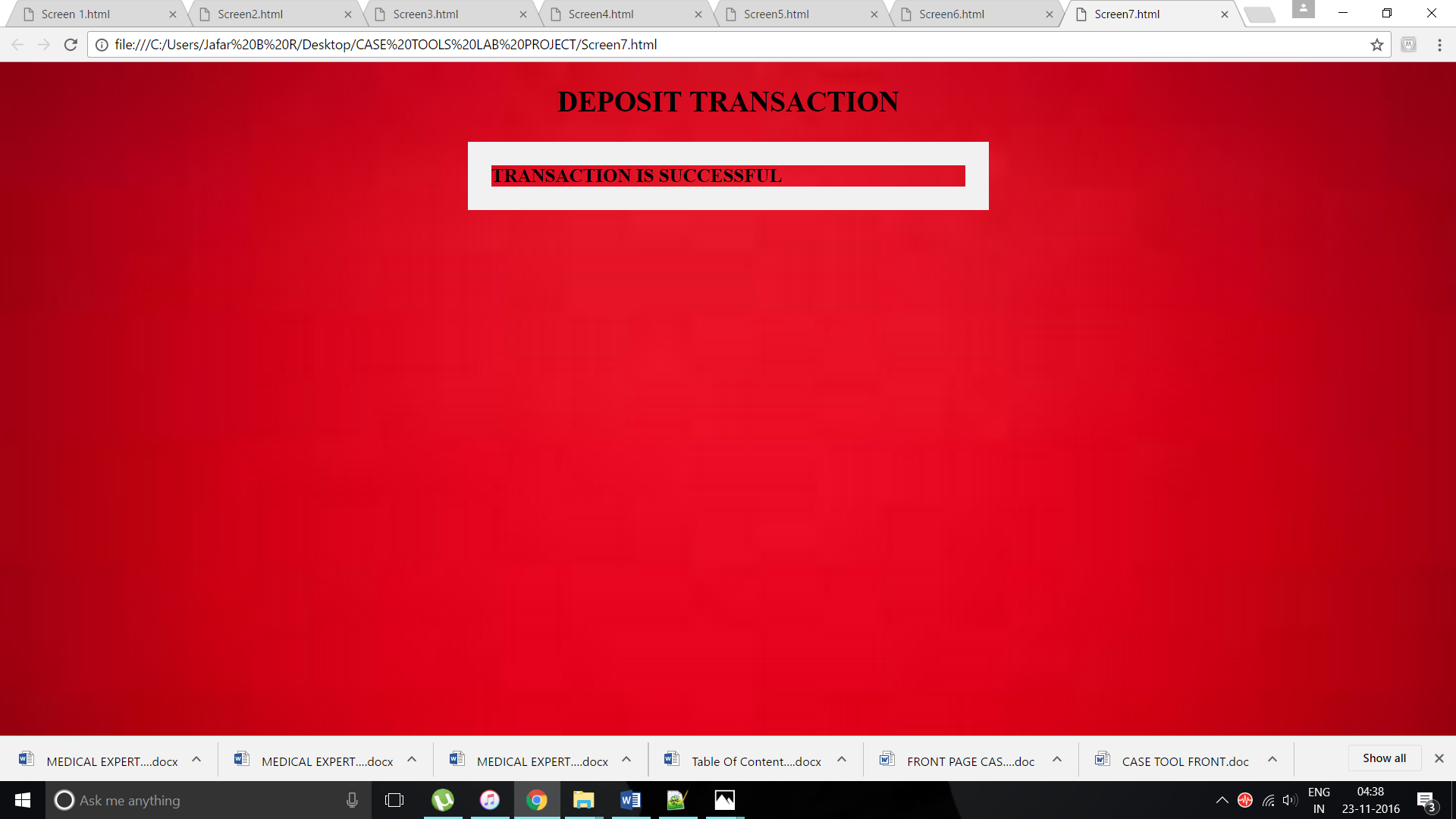
**WITHDRAWAL MODULE**:



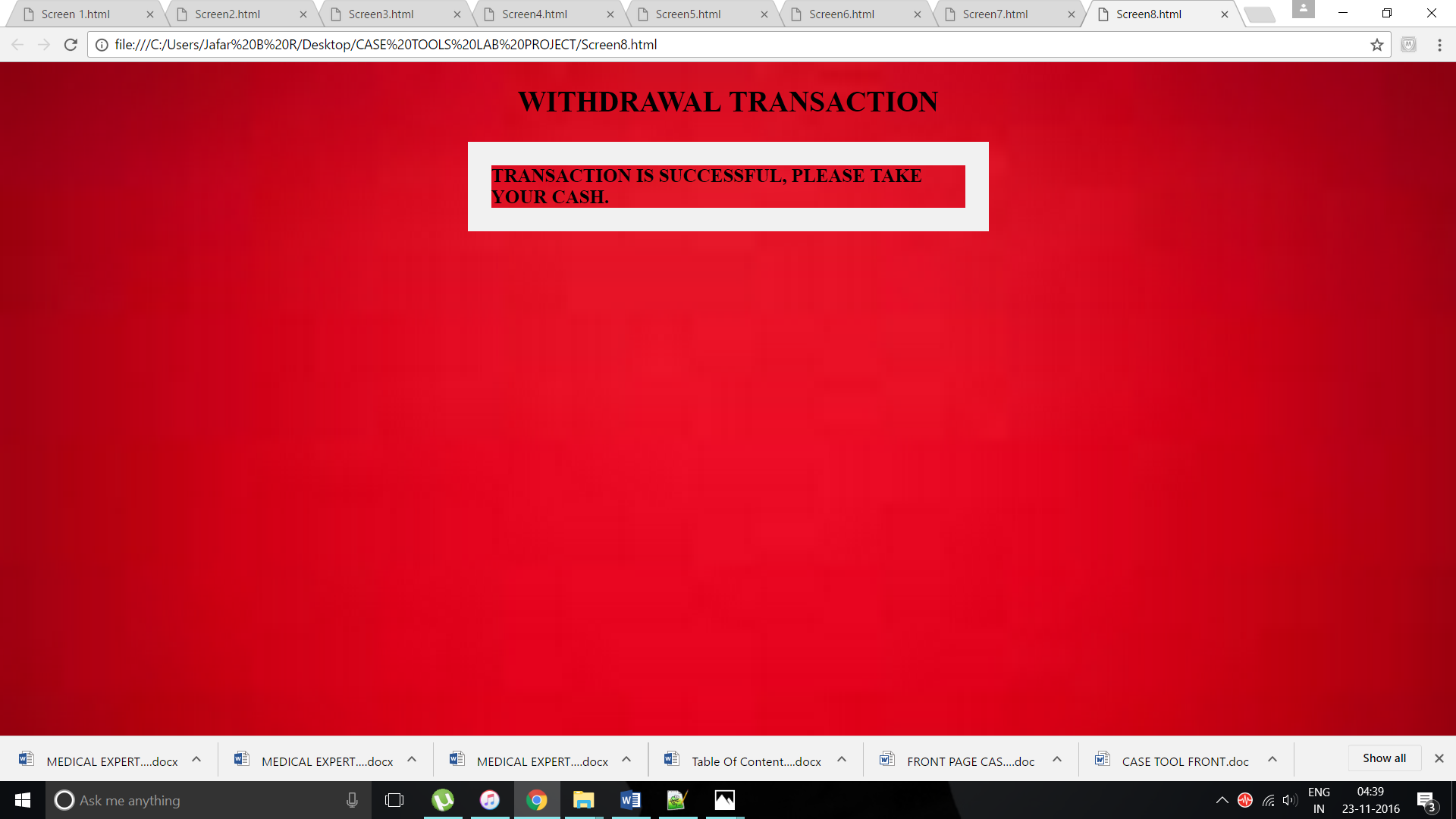
**CHECKING BALANCE MODULE:**



**DEPOSIT TRANSACTION MODULE:**



**WITHDRAWAL TRANSACTION MODULE:**



**7.CONCLUSION:**

The ATM system is developed using software rational rose using UML diagrams.