SMART CARD FOR SMART CITY CITIZEN –CREDIT CARD CUSTOMER PREDICTION USING SMART CARD

A PROJECT REPORT

Submitted by

N. JOHNSON SAJJIN RAJ	**210214205004**
S. K. NIVESH	**210214205007**

In partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

APOLLO ENGINEERING COLLEGE

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2018

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"Smart Card For Smart City Citizen-Credit Card Customers Prediction Using Smart Card**" is the bonafide work of "**N.Johnson Sajjin Raj (Registration Number:210214205004) & S.K.Nivesh(Registration number:210214205007)**" who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation on the basis of the which degree or award was conferred on an earlier occasion on this or any other candidate.

SUPERVISOR	**HEAD OF THE DEPARTMENT**
Dr.M.P. Sivaram Kumar M.E,Ph.D	**Mr. E. Murali M. TECH,(Ph.D)**
Principal and Professor	**HOD and Assistant professor**
Department of CSE	Department of CSE
Apollo Engineering College	Apollo Engineering College
Kanchipuram-600 025	Kanchipuram-600 025

Submitted to project and Viva Examination held on _____

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

We express our profound gratitude to our Chairman, Vice Chairman and Secretary of our college, for providing us the necessary facilities to carry out the project work.

We would like to express my sincere thanks to **Dr.M.P.Sivaram Kumar M.E,Ph.D.,** Principal and **Prof. K.Velusamy,** Vice Principal of our college for their constant support and encouragement towards completion of this project.

We like to express our heartfelt thanks to **E.Murali M.TECH,(Ph.D.),** and Head of the Department, Professor and internal guide, Department of Information Technology, **Mrs. R.Sudha M.TECH.**, Project Coordinator for help in completing the project.

We would like to express my heartfelt thanks to the department staffs and family members for their continuous support and encourage us.

ABSTRACT

A government take many decisions to make a cites to digitalized. Many plans have to improve the small cites in to the smart cities. In the modern world people can't like to hold many things in the packet. Instead of using many card use one single card that contains many information about that person like medical, personal, bank, loan, gold, property. Using that statement government able to take a prediction low level people. And government give the service to the low level people. This card has a unique identification number and analysing the all bank name, date of birth, address based on these information card verifies. So provides integrity for the card user. Based on the card the credit card agency can able to access the mini statement of the bank account. Using that predict the valid user. And health information also recorded using that card that use for doctor easily identify the peoples previous medical statement.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER-1

INTRODUCTION

1.1 INTRODUCTION

In the modern world people try to save time for each and every places. In the medical department the doctors need the previous check-up report but they forget to bring it. And the interview we need the 10^{th}, 12^{th} and degree statement are get it from digitally by card. The single card holds all the information about the person. The government is providing confidentiality to the person detail from others. Only the person and authorized agent can able access the data. No need to hold all the document to all places. Because all over data are the government digital database so people easy to access the information at any time. The card contains bank, loan, property, gold data of a single person. So using the data the government able to predict the people status and who need financial helps. we doing the one application of the card predict the credit card customers by using bank statements. That useful for many applications like bus charge for senior citizens, Medical suggestion to each pupil, Ration card fraud detection etc.

1.2 OVERVIEW OF PROJECT

Generate the credit card by the smart card by using the bank statement. The bank statement is get when we register the smart card the smart card agency request to the corresponding bank to the mini statement about the account number the agency send the account number, name, address to the bank if the details are match then the mini statement are update to the smart card database.

1.3 OBJECTIVE OF THE PROJECT

- Predict the customers financial state by the bank accounts.
- Based on different bank account statement credit card issued.

CHAPTER-2

LITERATURE SURVEY

1. Credit card fraud detection using Machine Learning Techniques

Financial fraud is an ever growing menace with far consequences in the financial industry. Data mining had played an imperative role in the detection of credit card fraud in online transactions. Credit card fraud detection, which is a data mining problem, becomes challenging due to two major reasons – first, the profiles of normal and fraudulent behaviours change constantly and secondly, credit card fraud data sets are highly skewed. The performance of fraud detection in credit card transactions is greatly affected by the sampling approach on dataset, selection of variables and detection technique(s) used. This paper investigates the performance of naïve Bayes, k-nearest neighbour and logistic regression on highly skewed credit card fraud data. Dataset of credit card transactions is sourced from European cardholders containing 284,807 transactions. A hybrid technique of under-sampling and oversampling is carried out on the skewed data. The three techniques are applied on the raw and pre-processed data. The work is implemented in Python. The performance of the techniques is evaluated based on accuracy, sensitivity, specificity, precision, Matthews correlation coefficient and balanced classification rate. The results show of optimal accuracy for naïve Bayes, k-nearest neighbour and logistic regression classifiers are 97.92%, 97.69% and 54.86% respectively. The comparative results show that k-nearest neighbour performs better than naïve Bayes and logistic regression techniques.

2. Adversarial Learning in Credit Card Fraud Detection

Credit card fraud is an expensive problem for many financial institutions, costing billions of dollars to companies annually. Many adversaries still evade fraud detection systems because these systems often do not include information about the adversary's knowledge of the fraud detection mechanism. This project

aims to include information about the “fraudster’s” motivations and knowledge base into an adaptive fraud detection system. In this project, we use a game theoretical adversarial learning approach in order to model the fraudster’s best strategy and pre-emptively adapt the fraud detection system to better classify these future fraudulent transactions. Using a logistic regression classifier as the fraud detection mechanism, we initially identify the best strategy for the adversary based on the number of fraudulent transactions that go undetected, and assume that the adversary uses this strategy for future transactions in order to improve our classifier. Prior research has used game theoretic models for adversarial learning in the domains of credit card fraud and email spam, but this project adds to the literature by extending these frameworks to a practical, real-world data set. Test results show that our adversarial framework produces an increasing AUC score on validation sets over several iterations in comparison to the static model usually employed by credit card companies.

3. Credit Card Fraud Detection Based on Transaction Behaviour

The proliferation of the EMV (Europay- MasterCard-VISA) chip card design in the credit card business mostly resolved the problem posed by the old Magnetic stripe card technology. However, several papers are starting to question the design and implementation of the EMV. This paper is suggesting that a detection model must be available to capture the possible anomalous transactions – a fall back in case the technology will fail. Several classifiers were evaluated during the model creation however only the Random Tree and J48 yielded the highest accuracy value of 94.32% and 93.50% respectively. By thorough analysis of these two (2) classifiers, it shows that the J48 is more fit in understanding the transaction logs data.

CHAPTER-3

3.SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

In the existing system that analysing customers by the statement of the bank account transaction. The transaction information gets from the bank and the credit card agencies analysing the transaction statement and give the credit card.

3.1.1 Disadvantage

In the system is fails when the customers have number of bank accounts. And that cannot analysing the information of loan information so many people get the credit card but cannot validate the people who return the amount.

3.2 PROPOSING SYSTEM

In the system connect all the information into the single card. The card named as smart card that useful for the smart cities. The card contains bank, loan, property, relationship, medical, etc. information. Using in this card we evaluate the customers by the all the bank statement and loan statement. And useful for the medical information. If the customer is suddenly admitting to the hospital. That use to analysing the previous medical information.

3.2.1 Advantage

Identify the people who financially poor in the society and identify the senior citizen of the society like information. Reduce the fraud people in the society by unique identification number.

3.3 SYSTEM ARCHITECHURE

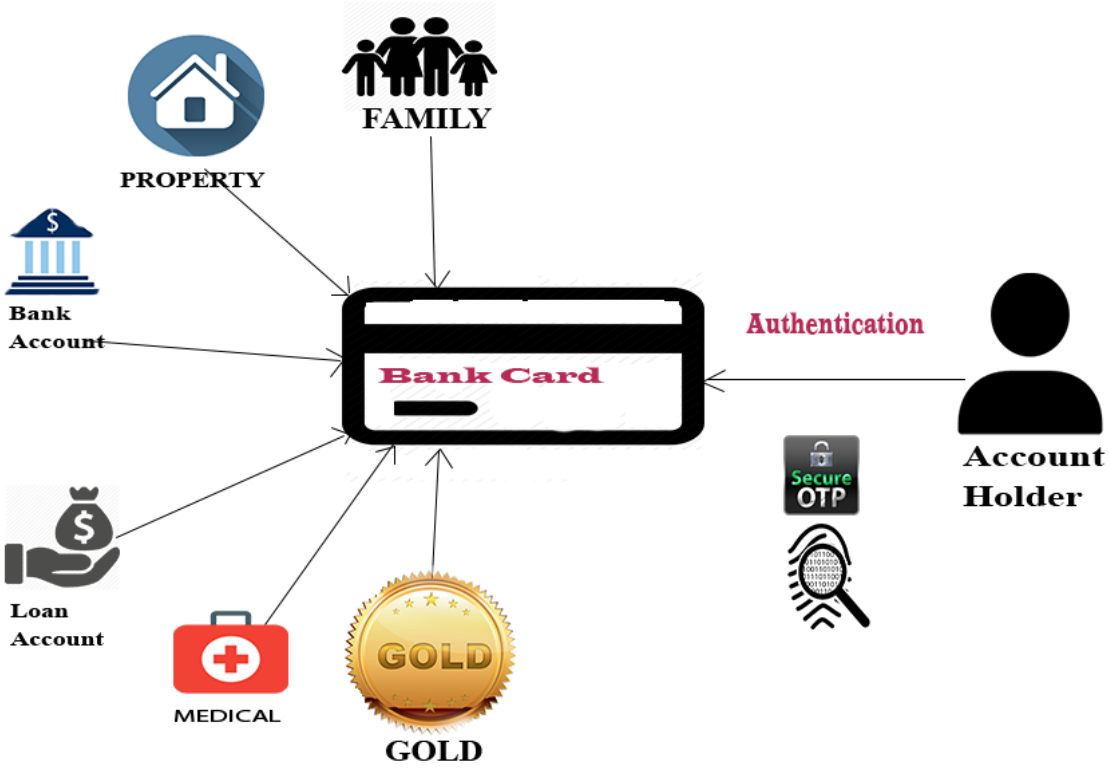


Figure 3.3.1 smart card registration

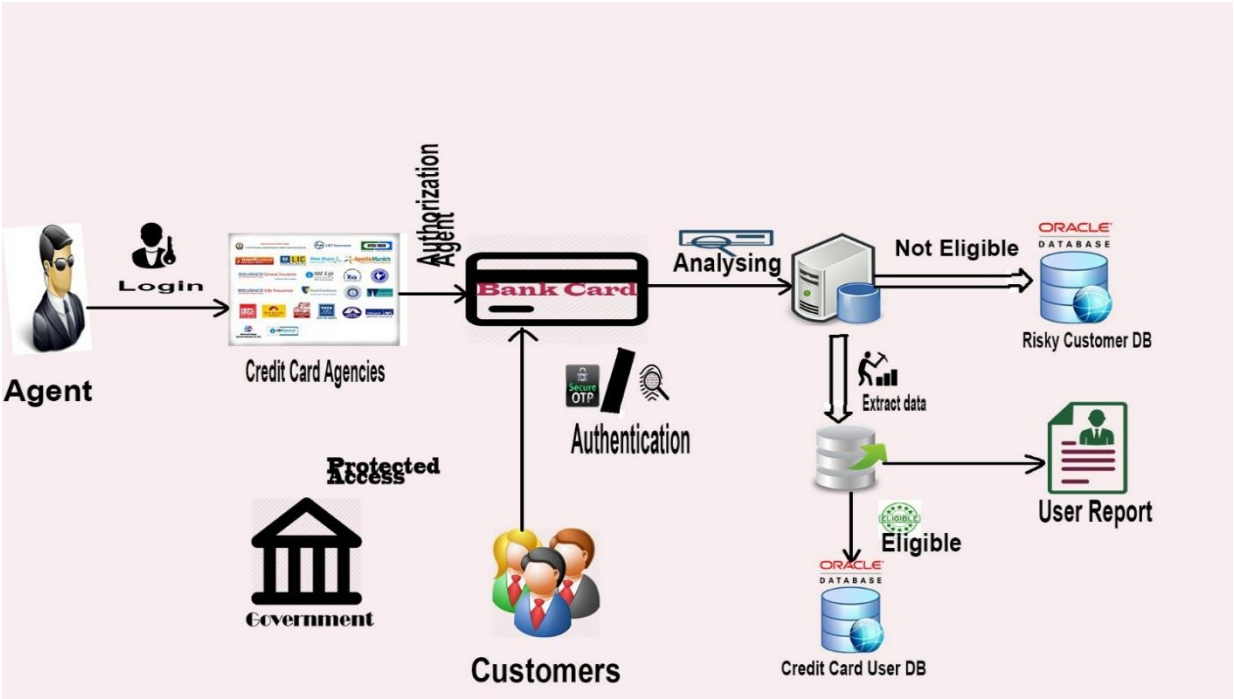


Figure 3.3.2 credit card user prediction

CHAPTER-4

SYSTEM DESIGN

4.1 UML DIAGRAM

4.1.1 USE CASE DIAGRAM

A use case diagram is a dynamic or behaviour diagram in UML. Use case diagrams model the functionality of a system using actors and use cases. Use cases are a set of actions, services, and functions that the system needs to perform.

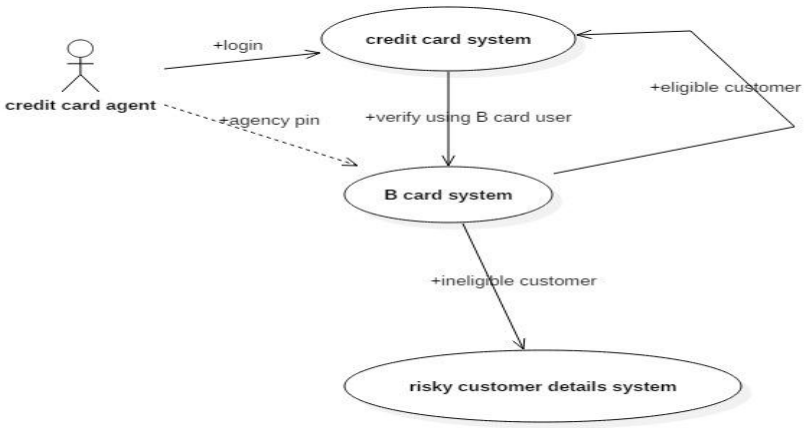


Figure 4.1.2.1 use case diagram

4.1.2 ACTIVITY DIAGRAM

Activity diagrams illustrate the dynamic nature of a system by modelling the flow of control from activity to activity. An activity represents an operation on some class in the system that results in a change in the state of the system. Typically, activity diagrams are used to model workflow or business processes and internal operation**.**

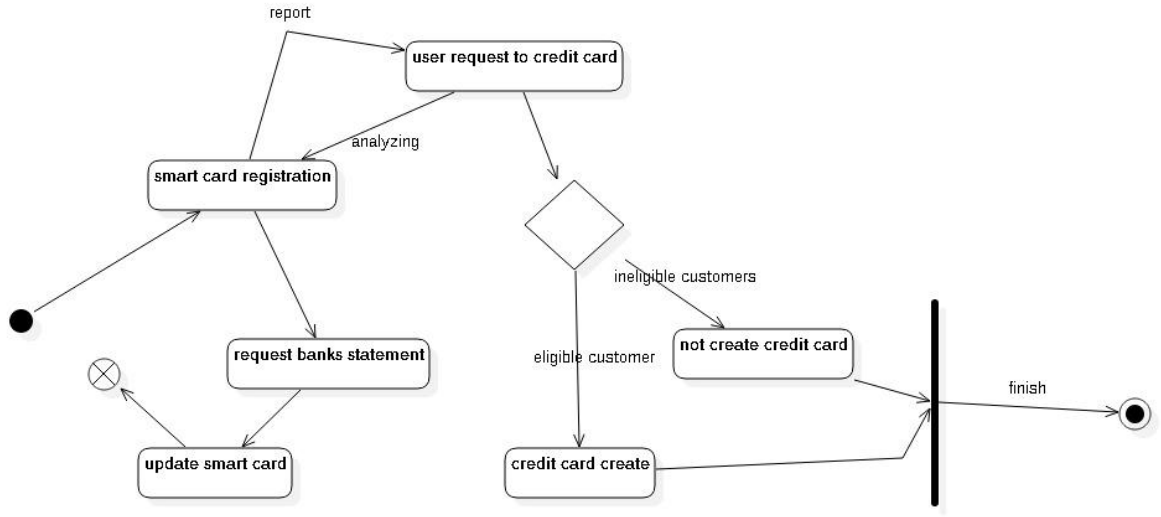


Figure 4.1.2.1 Activity diagram

4.1.3 CLASS DIAGRAM

A class diagram models the static structure of a system. It shows relationships between classes, objects, attributes, and operations.

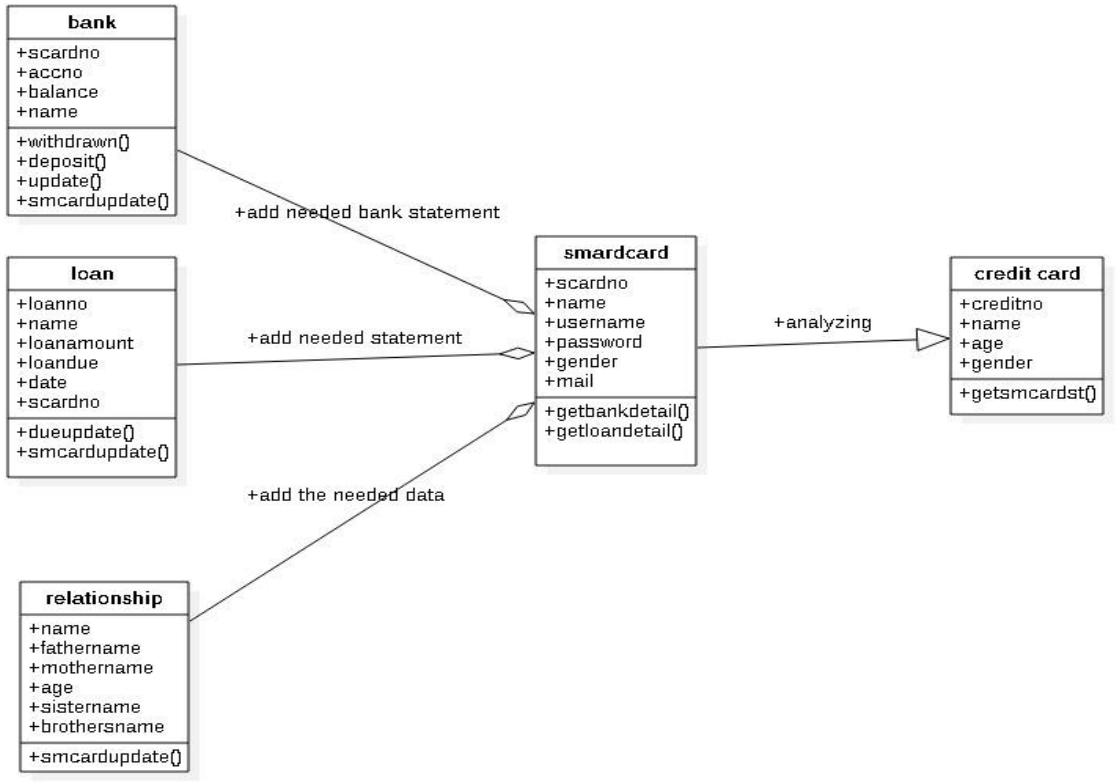


Figure 4.1.3.1 Class diagram

4.1.4 PACKAGE DIAGRAM

Package diagrams are a subset of class diagrams, but developers sometimes treat them as a separate technique. Package diagrams organize elements of a system into related groups to minimize dependencies between packages.

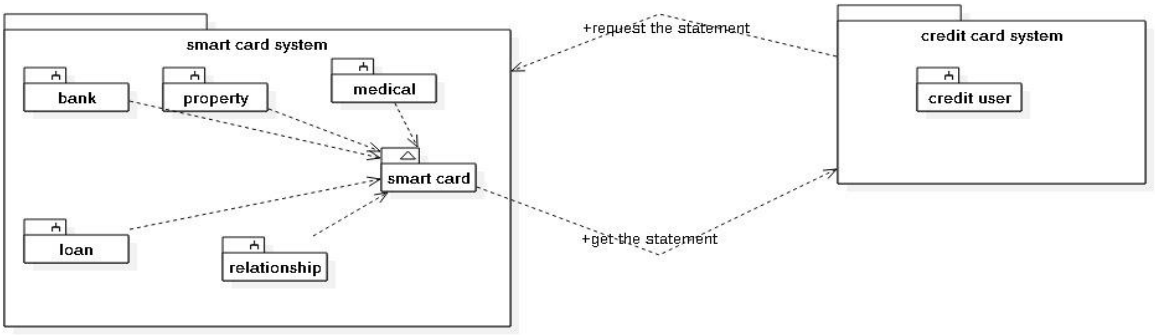


Figure 4.1.4.1 Package diagram

4.1.5 SEQUENCE DIAGRAM

Sequence diagrams describe interactions among classes in terms of an exchange of messages over time. They're also called event diagrams. A sequence diagram is a good way to visualize and validate various runtime scenarios.

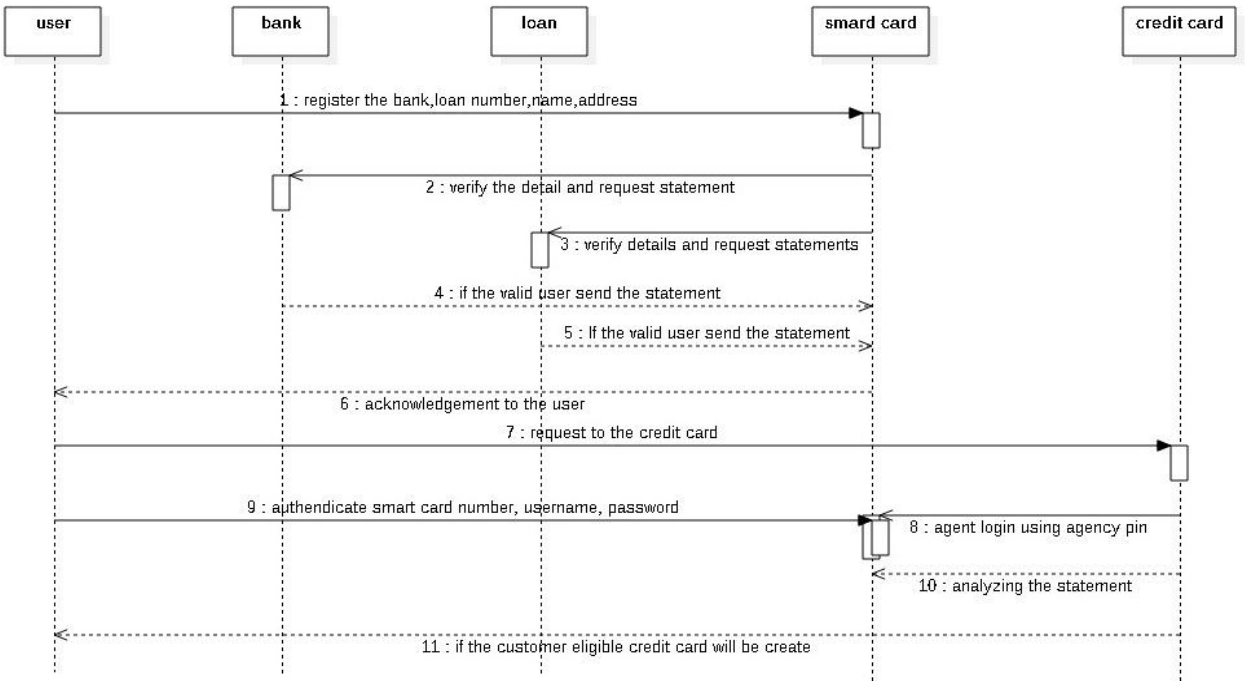


Figure 4.1.5.1 sequence diagram

4.2 MODULES

4.2.1 SMART CARD REGISTRATION

The customer's signup the account with the bank accounts, loan account, name, address and submit the registration form. The smart card admin sends the request to the banks with the respective customer's name, address, account number. If the customer's information is true, then the bank online admin sends some authorized information to the smart card admin that will be update in certain hours based on the bank server. The information's are update in the smart card.

4.2.2 CREDIT CARD PREDICTION

The credit card agency agent signs in the credit card system. From there the system the smart card sign in portal will be open enter the smart card information, username, password based on the system the smart card system returns true or false. If the system returns true, then the credit card will be generating otherwise the system update the customer is the risky customer.

4.2.3 OTHER SMART CARD INFORMATION UPDATE

The customer's other information's are update by the respective system. For example, if you want gold information update into smart card by the gold system. Like that property, relationship, medical information's are updating by the respective system.

4.3 SYSTEM SPECIFICATION

4.3.1 HARDWARE REQUIREMENT

PROCCESSOR	2.0GHz and above.
RAM	2GB and above.
MEMORY	160GB and above.

4.3.2 SOFTWARE REQUIREMENT

PLATFORM	Windows 7and above
PACKAGE	JDK 1.7 and above, eclipse mars, heidiSQL, Apache tomcat
LANGUAGES	Advance Java, HTML, CSS, JSP, MySQL, JavaScript

4.4 SOFTWARE DESCRIPTION

INTRODUCTION TO SERVLET

A Java servlet processes a Java class in Java EE that conforms to the Java Servlet API, a standard for implementing Java classes that respond to requests. Servlets could in principle communicate over any client server protocol, but they are most often used with the HTTP protocol. Thus "servlet" is often used as shorthand for "HTTP servlet". Us, a software developer may use a servlet to add dynamic content to a web server using the Java platform. The generated content is commonly HTML, but may be other data such as XML. Servlets can maintain state in session variables across many server transactions by using HTTP cookies, or URL rewriting.

To deploy and run a servlet, a web container must be used. A web container (also known as a servlet container) is essentially the component of a web server that interacts with the servlets. The web container is responsible for managing the lifecycle of servlets, mapping a URL to a particular servlet and ensuring that the URL requester has the correct access rights.

The Servlet API, contained in the Java package hierarchy javax.servlet, defines the expected interactions of the web container and a servlet.

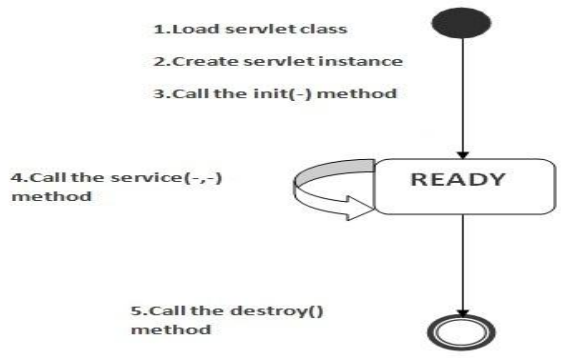
A Servlet is an object that receives a request and generates a response based on that request. The basic Servlet package defines Java objects to represent servlet requests and responses, as well as objects to reflect the servlet's configuration parameters and execution environment. The package

javax.servlet.http defines HTTP-specific subclasses of the generic servlet elements, including session management objects that track multiple requests and responses between the web server and a client. Servlets may be packaged in a WAR file as a web application.

Servlets can be generated automatically from Java Server Pages (JSP) by the Java Server Pages compiler. The difference between servlets and JSP is that servlets typically embed HTML inside Java code, while JSPs embed Java code in HTML. While the direct usage of servlets to generate HTML (as shown in the example below) has become rare, the higher level MVC web framework in Java EE (JSF) still explicitly uses the servlet technology for the low level request/response handling via the FacesServlet. A somewhat older usage is to use servlets in conjunction with JSPs in a pattern called "Model 2", which is a flavour of the model–view–controller.

Life Cycle of a Servlet

1. Servlet class is loaded.
2. Servlet instance is created.
3. init method is invoked.
4. service method is invoked.
5. destroy method is invoked.



JDBC

Java Database Connectivity (**JDBC**) is an application programming interface (API) for the programming language Java, which defines how a client may access a database. It is Java based data access technology and used for Java database connectivity. It is part of the Java Standard Edition platform, from Oracle Corporation. It provides methods to query and update data in a database, and is oriented towards relational databases. A JDBC-to-ODBC bridge enables connections to any ODBC-accessible data source in the Java virtual machine (JVM) host environment.

JDBC Drivers Types

JDBC driver implementations vary because of the wide variety of operating systems and hardware platforms in which Java operates. Sun has divided the implementation types into four categories, Types 1, 2, 3, and 4, which is explained below −

Type 1: JDBC-ODBC Bridge Driver

In a Type 1 driver, a JDBC bridge is used to access ODBC drivers installed on each client machine. Using ODBC, requires configuring on your system a Data Source Name (DSN) that represents the target database.

When Java first came out, this was a useful driver because most databases only supported ODBC access but now this type of driver is recommended only for experimental use or when no other alternative is available.

Type 2: JDBC-Native API

In a Type 2 driver, JDBC API calls are converted into native C/C++ API calls, which are unique to the database. These drivers are typically provided by the database vendors and used in the same manner as the JDBC-ODBC Bridge. The vendor-specific driver must be installed on each client machine.

If we change the Database, we have to change the native API, as it is specific to a database and they are mostly obsolete now, but you may realize some speed increase with a Type 2 driver, because it eliminates ODBC's overhead.

Type 3: JDBC-Net pure Java

In a Type 3 driver, a three-tier approach is used to access databases. The JDBC clients use standard network sockets to communicate with a middleware application server. The socket information is then translated by the middleware application server into the call format required by the DBMS, and forwarded to the database server.

This kind of driver is extremely flexible, since it requires no code installed on the client and a single driver can actually provide access to multiple databases.

You can think of the application server as a JDBC "proxy," meaning that it makes calls for the client application. As a result, you need some knowledge of the application server's configuration in order to effectively use this driver type.

Type 4: 100% Pure Java

In a Type 4 driver, a pure Java-based driver communicates directly with the vendor's database through socket connection. This is the highest performance driver available for the database and is usually provided by the vendor itself.

This kind of driver is extremely flexible; you don't need to install special software on the client or server. Further, these drivers can be downloaded dynamically.

HTML

HTML5 is a mark-up used for structuring and presenting content on the World Wide Web. It is the fifth and current major version of the HTML standard.

It was published in October 2014 by the World Wide Web Consortium (W3C) to improve the language with support for the latest multimedia, while keeping it both easily readable by humans and consistently understood by computers and devices such as web browsers, parsers, etc. HTML5 is intended to subsume not only HTML 4, but also XHTML 1 and DOM Level 2 HTML.

HTML5 includes detailed processing models to encourage more interoperable implementations; it extends, improves and rationalizes the mark-up available for documents, and introduces mark-up and application programming interfaces(APIs) for complex web applications. For the same reasons, HTML5 is also a candidate for cross-platform mobile applications, because it includes features designed with low-powered devices in mind.

The APIs and Document Object Model (DOM) are now fundamental parts of the HTML5 specification and HTML5 also better defines the processing for any invalid documents.

SQL

Structured Query Language) is a language used in programming and designed for managing data held in a relational database management system(RDBMS), or for stream processing in a relational data stream management system (RDSMS). In comparison to older read/write APIs like ISAM or VSAM, SQL offers two main advantages: first, it introduced the concept of accessing many records with one single command; and second, it eliminates the need to specify how to reach a record, e.g. with or without an index.

Originally based upon relational algebra and tuple relational calculus, SQL consists of many types of statements, which may be informally classed as sublanguages, commonly: a data query language(DQL), a data definition

language (DDL), a data control language (DCL), and a data manipulation language (DML). The scope of SQL includes data query, data manipulation (insert, update and delete), data definition (schema creation and modification), and data access control. Although SQL is often described as, and to a great extent is, a declarative language (4GL), it also includes procedural elements.

SQL was one of the first commercial languages for Edgar F. Codd's relational model, as described in his influential 1970 paper, "A Relational Model of Data for Large Shared Data Banks". Despite not entirely adhering to the relational model as described by Codd, it became the most widely used database language.

SQL became a standard of the American National Standards Institute(ANSI) in 1986, and of the International Organization for Standardization(ISO) in 1987. Since then, the standard has been revised to include a larger set of features. Despite the existence of such standards, most SQL code is not completely portable among different database systems without adjustments.

JSP

JavaServer Pages (JSP) is a technology that helps software developers create dynamically generated web pages based on HTML, XML, or other document types. Released in 1999 by Sun Microsystems, JSP is similar to PHP and ASP, but it uses the Java programming language. To deploy and run JavaServer Pages, a compatible web server with a servlet container, such as Apache Tomcat or Jetty, is required.

JSP allows Java code and certain pre-defined actions to be interleaved with static web mark-up content, such as HTML, with the resulting page being compiled and executed on the server to deliver a document. The compiled pages, as well as any dependent Java libraries, contain Java bytecode rather than machine code. Like

any other Java program, they must be executed within a Java virtual machine (JVM) that interacts with the server's host operating system to provide an abstract, platform-neutral environment.

JSPs are usually used to deliver HTML and XML documents, but through the use of OutputStream, they can deliver other types of data as well. The Web container creates JSP implicit objects like request, response, session, application, config, page, pageContext, out and exception. JSP Engine creates these objects during translation phase.

Eclipse

Eclipse is an integrated development environment (IDE) used in computer programming, and is the most widely used Java 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 via plug-ins, including Ada, ABAP, C, C++, C#, COBOL, D, Fortran, Haskell, JavaScript, Ju lia, Lasso, Lua,NATURAL, Perl, PHP, Prolog, Python, R, Ruby (including Rub y on Rails framework), Rust, Scala, Clojure, Groovy, Scheme, and Erlang. It can also be used to develop documents with LaTeX (via a TeXlipse plug-in) and packages for the software Mathematica. Development environments include the Eclipse Java development tools (JDT) for Java and Scala, Eclipse CDT for C/C++, and Eclipse PDT for PHP, among others.The initial codebase originated from IBM VisualAge. The Eclipse software development kit (SDK), which includes the Java development tools, is meant for Java developers.

CHAPTER 5

TESTING AND MAINTENANCE

5.1 SYSTEM TESTING

Testing is the process of detecting errors. Testing plays a critical role in assuring quality and ensuring the reliability of software. The results of testing are used later on during maintenance also.

5.2 TESTING OBJECTIVES

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time.

- Testing is a process of executing a program with the intent of finding an error
- A good test case is one that has a high probability of finding error, if it exists
- The tests are inadequate to detect possibly present errors
- The software more or less confirms to the quality and reliable standards

5.3 TESTING LEVELS

System testing is stage of implementation which is aimed at ensuring that the system works accurately and efficient before live operation commences. Testing is vital the success of the system. System testing makes a logical assumption that if all the parts of the system are correct, the goal will be successfully achieved.

5.3.1 Unit Testing

In the lines of strategy, all the individual functions and modules were put to the test independently. By following this strategy all the errors in coding were identified and corrected. This method was applied in combination with the White and Black Box Testing Techniques to find the errors in each module.

5.3.2 Integration Testing

Data can be lost across the interface; one module can have an adverse effect on others. Integration testing is a systematic testing for constructing

program structure. While at the same time conducting tests to uncover errors associated within the interface. Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order sets and conducted.

The objective is to take unit tested modules and combine them test it as a whole. Thus, in the integration-testing step all the errors uncovered are corrected for the next testing steps.

5.3.3 Validation Testing

The outputs that come out of the system are as a result of the inputs that go into the system. The correct and the expected outputs that go into the system should be correct and proper. So, this testing is done to check if the inputs are correct and they are validated before it goes into the system for processing.

5.3.4 Acceptance Testing

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

- Input screen design
- Output screen design

An acceptance test has the objective of selling the user on the validity and reliability of the system. It verifies that the system's procedures operate to system specifications and that the integrity of important data is maintained. Performance of an acceptance test is actually the user's show. User motivation is very important for the successful performance of the system. After that a comprehensive report is prepared.

CHAPTER-6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONCLUSION

The system very useful for the smart city people. The government has easily identified the people and about the customer's information by the single unique identification number. That reduce the people pocket weight. The people no need for hold many card for the identification in the card contains full information. The government agency easily identifies the people information by the unique identification number. The credit card agency gets the bank details by the smart card. The people information protects by the cryptography techniques.

6.2 FUTURE ENHANCEMENT

In the future plan for the system is concentration on the medical field. If the patient admits in serious condition at this condition doctor have previous medical report that useful to the doctor further treatment suddenly. And gives the high security for the medical information for the individual people. The people information's are the maintain in the database with high firewall security over the internet. And another one plan is establishing the high security over the internet protocol then the bank transaction establishes by using biometric (Retinal scan, Fingerprint). And don't know the person details the basic information are access by any two verifications without smart card number. The smart card information's are get by the public key exchange technique. In the public key exchange technique, the people register smart card the information encrypt by the private key. The public key is share to the other systems the information will decrypt by the public key.

APPENDIX I

SAMPLE CODE

bcardreg.html

```
<!DOCTYPE html>

<html>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

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

<title>SMART CARD REGISTER</title><link
rel="icon”href="/project/card.png"type="image/png"></head><body><ul><li>
<ahref="/project/home.html">HOME</a></li><li><ahref="/project/agentreg.ht
ml">CREDIT CARD AGENT REGISTRATION</a></li><li><a
href="/project/bankupdate.html">BANK DATABASE
UPDATE</a></li><li><a href="/project/propery.html">PROPERTY</a></li>

<li><a href="/project/gold.html">GOLD</a></li><li><a
href="/project/bcardreg.html">B_CARD_INFO </a></li><li><a
href="/project/loanupdate.html">LOAN DATABASE
UPDATE</a></li><li><a href="/project/medical.html">MEDICAL</a></li>

<li><a href="/project/relation.html">RELATION</a></li></ul><form
id="regForm" action="servlet/Bcardreg" method="post"><div
class="imgcontainer">  <img src="/project/card.png" alt="card" class="card">

 </div>    <div align="center"> <b>B CARD REGISTRATION</b></div>

  <div class="tab">Name:<p><input placeholder="Enter Name"
oninput="this.className = "" name="name"></p>   User Name:    <p><input
placeholder="Enter User name" oninput="this.className = ""
```

```
name="user"></p></div><divclass="tab">Password:<p><inputtype="password
" placeholder="password...." oninput="this.className = "" name="pass"></p>

    <p><input type="password" placeholder="Confirm Password...."
oninput="this.className = "" name="pass1"></p>

  </div>  <div class="tab">Birthday:    <p><input type="date"
placeholder="date of birth" oninput="this.className = ""
name="dob"></p>Address:<p><input placeholder="Enter Address"
oninput="this.className = "" name="addr"></p>

 Mail:<p><input placeholder="Enter mail address" oninput="this.className =
"" name="mail"></p>  </div>  <div class="tab">Account details <p><input
placeholder="Enter Bank account number" oninput="this.className = ""
name="accno"></p><p><input placeholder="Enter Loan Account Number"
oninput="this.className = "" name="loanno" >(if don't have specify null)</p>

 <p><input placeholder="Enter Second Bank Account Number"
oninput="this.className = "" name="accno8"></p>  </div>  <div class="tab">
<label class="container">male<input type="radio" checked="checked"
name="gender" value="male"> </label><label class="container">female

 <input type="radio" name="gender" value="female"><span
class="checkmark"></span></label><label class="container">others<input
type="radio" name="gender" value="others"></label></div><button
type="button" id="prevBtn" onclick="nextPrev(-1)">Previous</button>
<button type="button" id="nextBtn" onclick="nextPrev(1)">Next</button>

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

</form>

</body>

</html>
```

Bcardreg.java

```
package bcard;

import java.io.*;

import java.sql.*;

import javax.servlet.*;

import com.mysql.jdbc.Connection;

public class Bcardreg extends HttpServlet {

private static final long serialVersionUID = 1L;

protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out=response.getWriter();

String name =request.getParameter("name");

String user=request.getParameter("user");

String pass=request.getParameter("pass");

String pass1=request.getParameter("pass1");

String email=request.getParameter("mail");

String dob =request.getParameter("dob");

String acc =request.getParameter("accno");

String acc1 =request.getParameter("accno8");

String lno =request.getParameter("loanno");

String gen =request.getParameter("gender");
```

```
String addr=request.getParameter("addr");

try {

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

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

if(pass.equals(pass1)){

PreparedStatement ps=con.prepareStatement("insert into
profile(name,acc_no,loan_no,address,username,paasword,dob,gender,mail,acc_
no1) values(?,?,?,?,?,?,?,?,?,?)");

ps.setString(5, user);

ps.setString(6, pass);

ps.setString(9, email);

ps.setString(7, dob);

ps.setString(1, name);

ps.setString(2, acc);

ps.setString(3, lno);

ps.setString(8, gen);

ps.setString(4, addr);

ps.setString(10, acc1);

int rs=ps.executeUpdate();

if(rs>0){
```

```
PreparedStatement ps3=con.prepareStatement("select * from profile where name=? and username=? and paasword=?");

ps3.setString(1, name);

ps3.setString(2, user);

ps3.setString(3, pass);

ResultSet rs3=ps3.executeQuery();

rs3.next();

String ccno=rs3.getString("b_card_number");

out.println(ccno);

out.println("register completed <a href='/project/bcardlogin.html'> click here</a> to login");}

else{  out.println("already exists account<a href='/project/bcardreg.html'> click here</a> to back to registration");}      }

} catch (Exception e) {

out.println("wrong entry");

out.println("<a href='/project/bcardreg.html'> click here</a> to back to registration");

e.printStackTrace();

}}}
```

Credit_agent.html

```
<!DOCTYPE html>

<html>

<head>
```

```
<link href="/project/css/a.css" rel="stylesheet"type="text/css">
<title>CREDIT CARD AGENT LOGIN</title>
<link rel="icon" href="/project/agent.png" type="image/png">
</head>
<body>
<ul>
<li><a href="">HOME</a></li>
<li><a href="/project/credit_agent.html">AGENT LOGIN</a></li>
<li><a href="/project/bcardlogin.html">B_CARD_DETAIL</a></li>
<li><a href="/project/user.html">REGISTRATION</a></li>
</ul>
 <form class="modal-content animate" action="servlet/Creditlogin" method="post">
<h1>AGENT LOGIN</h1>
 <label><b>USERID</b></label>
 <input type="text" placeholder="Enter Userid" name="user8" required>
 <label><b>PASSWORD</b></label>
<input type="password" placeholder="Enter Password" name="pass8" required>
<button type="submit">LOGIN</button>  </div></form>
</body>
</html>
```

Creditlogin.java

```
package credit;

import java.io.*;

import java.sql.*;

import javax.servlet.*;

import com.mysql.jdbc.Connection;

public class Creditlogin extends HttpServlet {

private static final long serialVersionUID = 1L;

protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out=response.getWriter();

String user =request.getParameter("user8");

String pass=request.getParameter("pass8");

try{

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

Connection con=(Connection)
DriverManager.getConnection("jdbc:mysql://localhost:3306/agent_db","root","r
oot");

PreparedStatement ps=con.prepareStatement("select * from profile where
user_id=? and password=?");

ps.setString(1, user);

ps.setString(2, pass);
```

```
ResultSet rs=ps.executeQuery();
 if(rs.next()){
RequestDispatcher rd=request.getRequestDispatcher("/user_validate.html");
rd.forward(request, response);
 }
 else{
  out.println("wrong user");
RequestDispatcher rd=request.getRequestDispatcher("/credit_agent.html");
rd.include(request, response);
}}
catch(Exception e){
out.println("Check your entry");
RequestDispatcher rd=request.getRequestDispatcher("/credit_agent.html");
rd.forward(request, response);
e.printStackTrace();
}}}
```

Creditreg.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
 pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
```

```
<head>
<link href="/project/css/a.css" rel="stylesheet"type="text/css">
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>CREDIT CARD REGISTRATION</title>
</head>
<body><ul>
<li><a href="">HOME</a></li>
<li><a href="/project/credit_agent.html">AGENT LOGIN</a></li>
<li><a href="/project/bcardlogin.html">B_CARD_DETAIL</a></li>
<li><a href="/project/user.html">REGISTRATION</a></li>
</ul>
<div align="center">
<fieldset id="reg">
<legend>CREDIT REGISTRATION</legend>
<h1>WELCOME</h1><%String namehe=request.getParameter("name45");
out.println(namehe.toUpperCase());
%>
<br>B_CARD_NUMBER:
<% String bcardhe=request.getParameter("bcard45");
out.println(bcardhe);%>
<form action="Creditreg" method="post" >
<table><tr>
```

```
<tr><td>NAME</td> <td>        <input type="text" name="name87" readonly
value=<% out.println(namehe);  %>></td></tr>

<tr><td>B CARD NUMBER:</td><td>         <input type="text"
name="bcard87" readonly value=<%= request.getParameter("bcard45")
%>></td></tr><tr><td>USER ID:</td> <td>        <input type="text"
name="user101" /></td></tr><tr><td>PASSWORD</td><td>         <input
type="password" name="pass101"/></td></tr><tr><td>CONFIRM
PASSWORD</td><td>   <input type="password" name="pass102"/></td></tr>

</table>

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

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

</form>

</fieldset>

</div>

</body>

</html>
```

Creditreg.java

```
package credit;

import java.io.*;

import java.sql.*;

import javax.servlet.*;

import com.mysql.jdbc.Connection;

public class Creditreg extends HttpServlet {
```

```
private static final long serialVersionUID = 1L;

protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out=response.getWriter();

String name =(String) request.getSession().getAttribute("ccnam");

String pass =request.getParameter("pass101");

String pass1 =request.getParameter("pass102");

String bcard =request.getParameter("bcard87");

String user=request.getParameter("user101");

try {

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

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

PreparedStatement ps=con.prepareStatement("select * from profile where
b_card_number=? and name=?  ");

ps.setString(2, name);

ps.setString(1, bcard);

ResultSet rs =ps.executeQuery();

rs.next();

if(pass1.equals(pass)){

String userdob=rs.getString("dob");
```

```
String useradd=rs.getString("address");

DateFormat df=new SimpleDateFormat("yyyy-MM-dd");

Calendar cal2=Calendar.getInstance();

cal2.add(Calendar.MONTH, 1);

String datepay=df.format(cal2.getTime());

Connection con1=(Connection)
DriverManager.getConnection("jdbc:mysql://localhost:3306/credit_card_db","r
oot","root");

 PreparedStatement ps1=con1.prepareStatement("insert into
profile(name,b_card_no,dob,user_id,password,address,date_to_pay)
values(?,?,?,?,?,?,?)  ");

ps1.setString(1, name);

ps1.setString(2, bcard);

ps1.setString(3, userdob);

ps1.setString(4, user);

ps1.setString(5, pass);

ps1.setString(6, useradd);

ps1.setString(7, datepay);

ps1.executeUpdate();

PreparedStatement ps2=con1.prepareStatement("select *from profile where
b_card_no=?");

ps2.setString(1, bcard);

ResultSet rs1=ps2.executeQuery();
```

```
rs1.next();
String cno=rs1.getString("c_card_no");
request.setAttribute("cno",cno);
request.setAttribute("nameofc",name);
request.setAttribute("ageofc",userdob);
request.setAttribute("useradd", useradd);
RequestDispatcher rd4=request.getRequestDispatcher("/report.jsp");
rd4.forward(request, response);
}}
catch (Exception e) {
e.printStackTrace();
out.println("already credit card user");
RequestDispatcher rd4=request.getRequestDispatcher("/credit.jsp");
rd4.include(request, response);
}}}
```

Report.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ page import = "java.io.*,java.util.*,java.sql.*"%>
<%@ page import = "javax.servlet.http.*,javax.servlet.*" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix = "c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix = "sql"%>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>report generation</title>

</head>

<body>

<fieldset>

<h1>REPORT</h1>

<table>

<tr><td>CARD NUMBER:</td><td><%=request.getAttribute("cno")
%></td><td></td></tr>

<tr><td>NAME:</td><td><%=request.getAttribute("nameofc")
%></td><td></td></tr>

<tr><td>DOB:</td><td><%=request.getAttribute("ageofc")
%></td><td></td></tr>

<tr><td>ADDRESS:</td><td><%=request.getAttribute("useradd")
%></td><td></td></tr>

</table>

</fieldset>

</body>

</html>
```

Smartcardlogin.html

```
<!DOCTYPE html>
<html>
<head>
<link href="/project/css/a.css" rel="stylesheet"type="text/css">
<title>SMART CARD LOGIN</title>
<link rel="icon" href="/project/user.png" type="image/png">
</head>
<body>
<ul>
<li><a href="">HOME</a></li>
<li><a href="/project/credit_agent.html">AGENT LOGIN</a></li>
<li><a href="/project/bcardlogin.html">B_CARD_DETAIL</a></li>
<li><a href="/project/user.html">REGISTRATION</a></li>
</ul>
 <form class="modal-content animate" action="Bcardlogin" method="post">
<div class="imgcontainer">
<img src="/project/user.png" alt="user" class="user">
</div>
<div align="center">
<b>BCARD LOGIN </b>
</div>
```

```
<div class="container">
<label><b>B CARD NUMBER</b></label>
<input type="text" placeholder="Enter b card number" name="bcard" required>
<label><b>USERNAME</b></label>
<input type="text" placeholder="Enter Username" name="name96" required>
<label><b>PASSWORD</b></label>
<input type="password" placeholder="Enter Password" name="pass96" required><label><b>ENTER TYPE OF DETAIL U NEED</b></label>
<select name="need">
<option value="bio">INFORMATION</option>
<option value="medical">MEDICAL</option>
<option value="bank">BANK</option>
<option value="loan">LOAN</option>
<option value="gold">GOLD</option>
<option value="property">PROPERTY</option>
<option value="relation">RELATION</option>
</select>
<button type="submit">LOGIN</button>
</div>
</form>
</body>
</html>
```

Smartcarddetail.jsp

```
<%@ page import = "java.io.*,java.util.*,java.sql.*"%>
<%@ page import = "javax.servlet.http.*,javax.servlet.*" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix = "c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix = "sql"%>
<html>
<head>
<link href="/project/css/a.css" rel="stylesheet"type="text/css">
<title>bcard data</title>
<link rel="icon" href="/project/card.png" type="image/png">
</head>
<body style="color:red;font-size:30px;">
WELCOME MR./MRS.
<%=request.getSession().getAttribute("bna").toString().toUpperCase() %>
<% String need=request.getSession().getAttribute("need").toString();%>
<%if(need.equals("bio")){ %>
<div align="center" id="basic" style="display:block;">
<h1>BASIC INFORMATION</h1>
<sql:setDataSource var = "snapshot" driver = "com.mysql.jdbc.Driver"
url = "jdbc:mysql://localhost/b_card_db"user = "root"  password = "root"/>
<sql:query dataSource = "${snapshot}" var = "result">
```

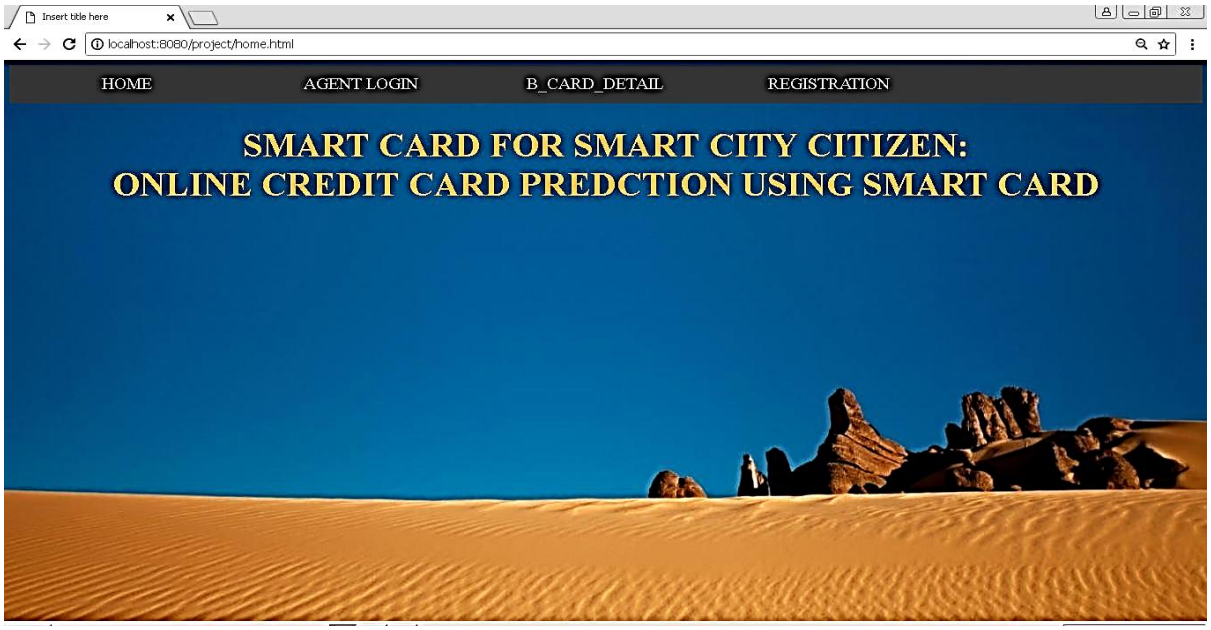
```
SELECT * from profile where
b_card_number=<%=request.getSession().getAttribute("bno").toString() %> ;

</sql:query>

<table width="50%" border = "12" style="padding:10px;border
style:groove;background-color:white;" >

<c:forEach var = "row" items = "${result.rows}">

<tr>

<td>B CARD NUMBER</td>

<td><c:out value = "${row.b_card_number}"/></td></tr>

<tr><td>NAME</td> <td><c:out value = "${row.name}"/></td></tr><tr>
<td>ADDRESS</td>  <td><c:out value = "${row.address}"/></td> </tr><tr>
<td>GENDER</td> <td><c:out value = "${row.gender}"/></td>
</tr><tr><td>DATE OF BIRTH</td>   <td><c:out value =
"${row.dob}"/></td></tr><tr>       <td>MAIL</td>        <td><c:out value =
"${row.mail}"/></td></tr>  <tr>  <td>GOLD</td> <td><c:out value =
"${row.gold}"/></td></tr> <tr> <td>PROPERTY</td>  <td><c:out value =
"${row.property}"/></td> </tr> </c:forEach>

</table>

<%} %>

<%if(need.equals("bank")){ %> <div align="center" id="bank"
style="display:block;">

<h1>BANK STATEMENT</h1>

<sql:setDataSource var = "snapshot" driver = "com.mysql.jdbc.Driver"

url = "jdbc:mysql://localhost/b_card_db"
```

```
user = "root"  password = "root"/>

<sql:query dataSource = "${snapshot}" var = "result">

SELECT * from bank_acc where
b_card_number=<%=request.getSession().getAttribute("bno").toString() %> ;

</sql:query>

<table border = "12" style="padding:10px;border-style:groove;background-
color:white;">

<c:forEach var = "row" items = "${result.rows}">

<tr> <td>ACCOUNT NUMBER</td>  <td><c:out value =
"${row.acc_no}"/></td></tr>

<tr> <td>NAME</td>    <td><c:out value = "${row.name}"/></td></tr>

<tr> <td>BALANCE</td>   <td><c:out value = "${row.balance}"/></td>

</tr> </c:forEach> </table>

</div>

<%} %>

<%if(need.equals("loan")){ %>

<div align="center" id="loan" style="display:block;">

<h1>LOAN STATEMENT</h1>

<sql:setDataSource var = "snapshot" driver = "com.mysql.jdbc.Driver"

url = "jdbc:mysql://localhost/b_card_db"user = "root"  password = "root"/>

 <sql:query dataSource = "${snapshot}" var = "result">
```

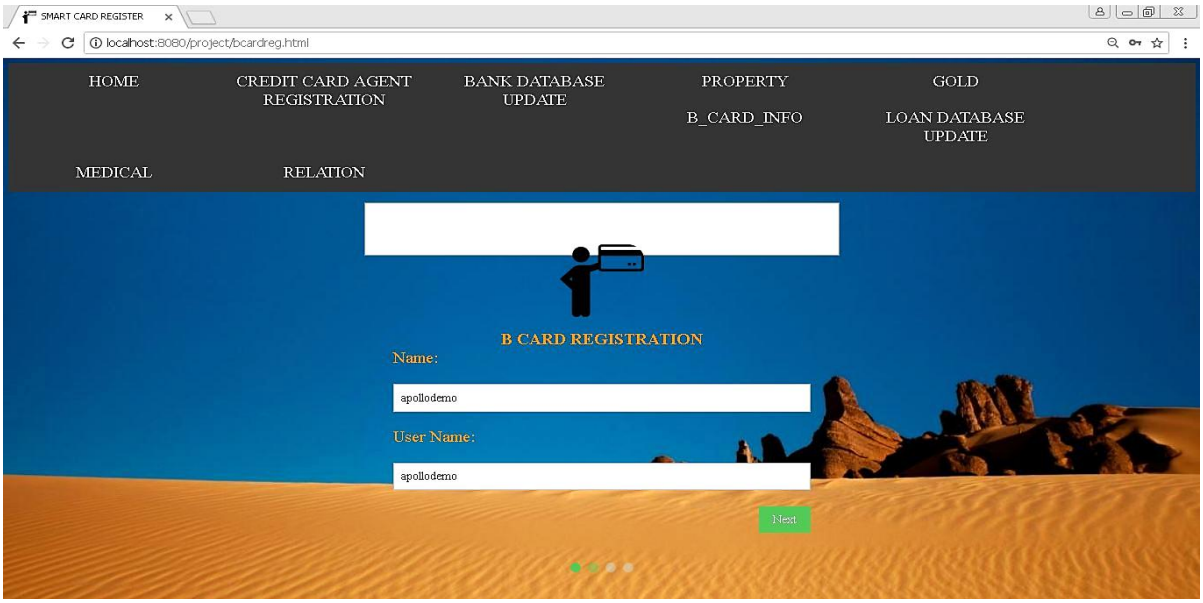
```
 SELECT * from loan_acc where
b_card_number=<%=request.getSession().getAttribute("bno").toString() %> ;
</sql:query>

<table border = "1" style="padding:10px;border-style:groove;background-
color:white;"> <tr></tr> <c:forEach var = "row" items = "${result.rows}">

<tr> <td>LOAN NUMBER</td> <td><c:out
value=${row.loan_no}"/></td></tr>

<tr> <td>NAME</td>

<td><c:out value = "${row.name}"/></td></tr>

<tr> <td>PENDING DUE</td>

<td><c:out value = "${row.pending_due}"/></td>

</tr>

</c:forEach>

</table>

</div>

<%} %>

</div><br><br>

</body>

</html>
```

APPENDIX II

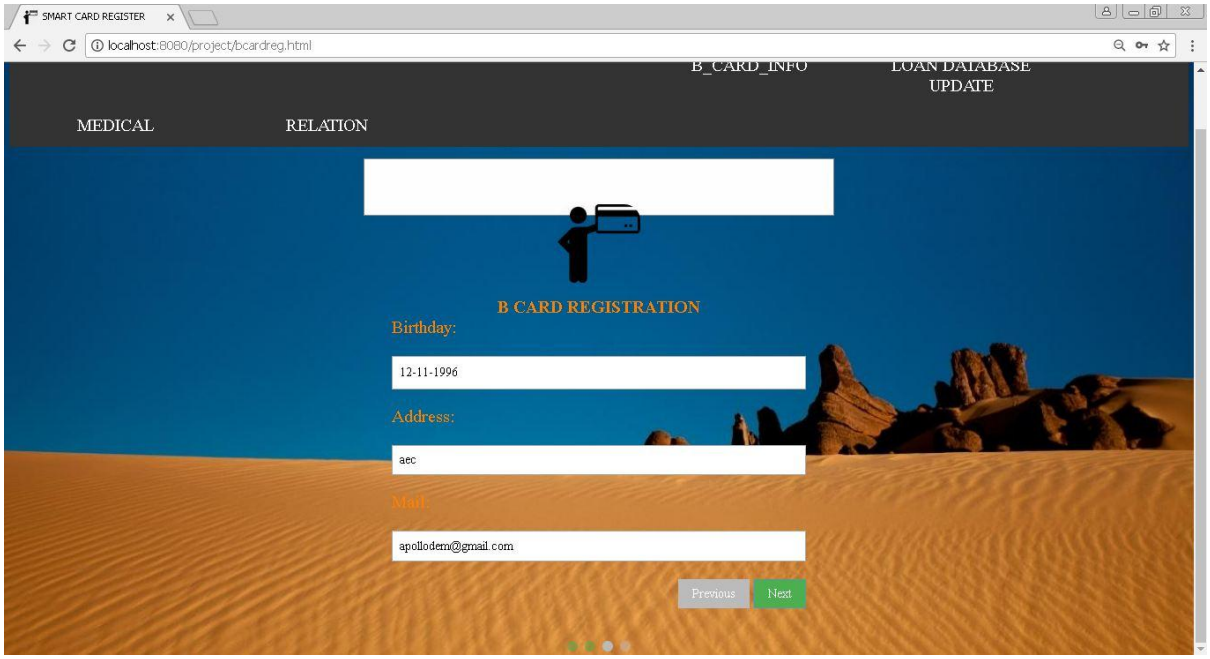
SCREEN SHOTS



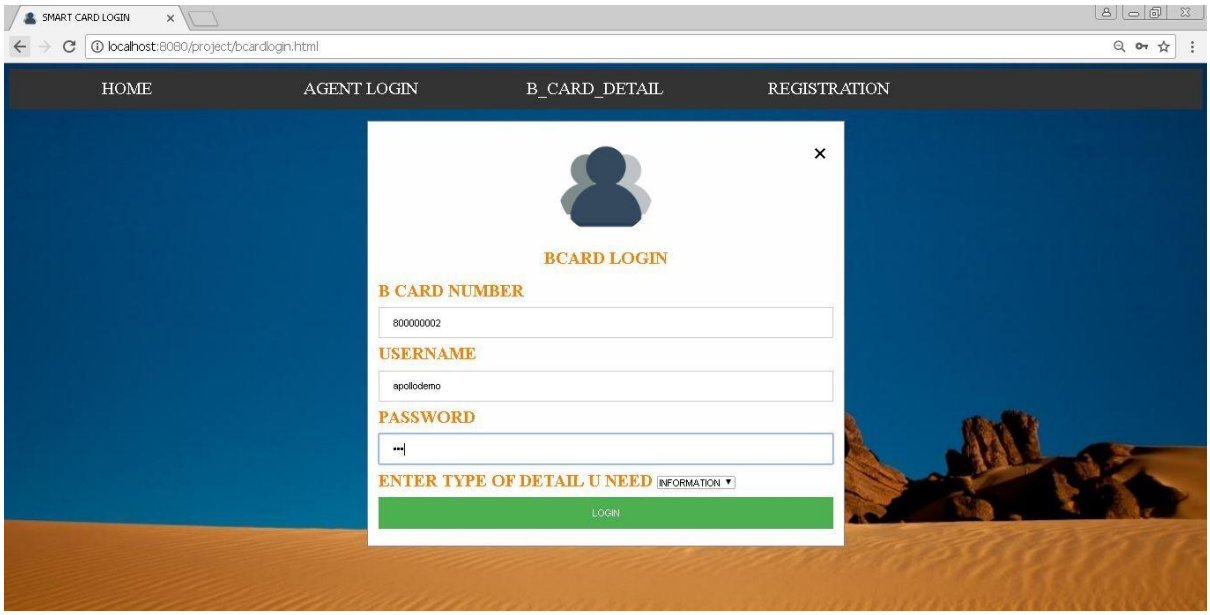
A1. home page



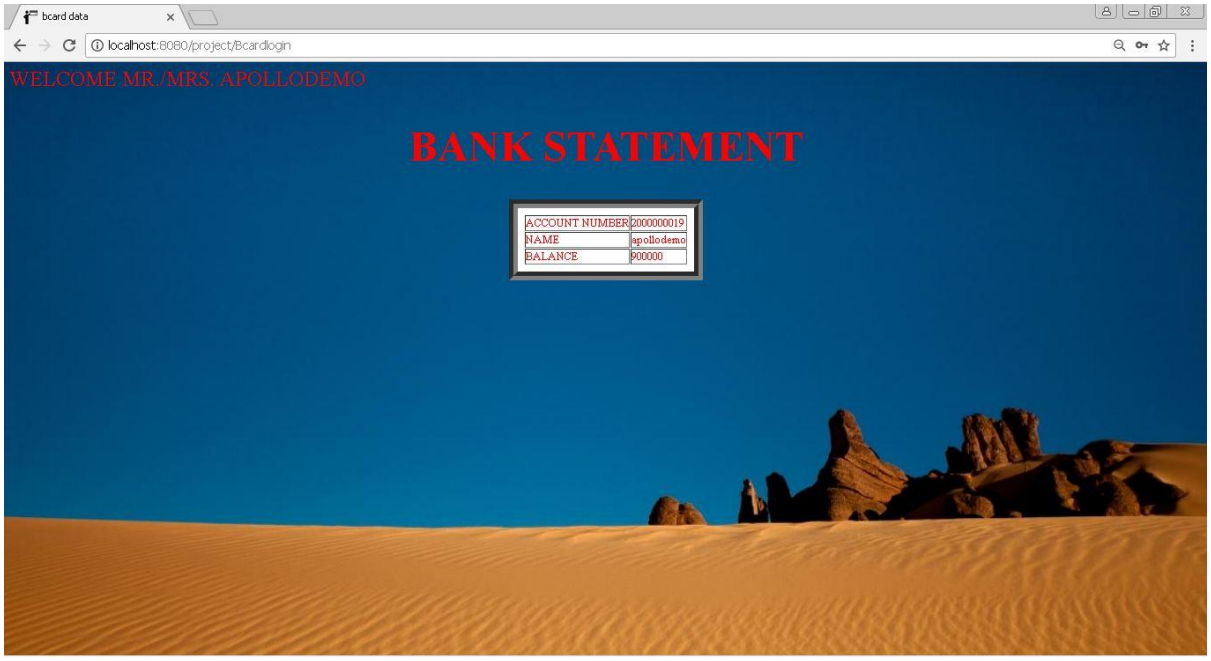
A2. Smart card name registration



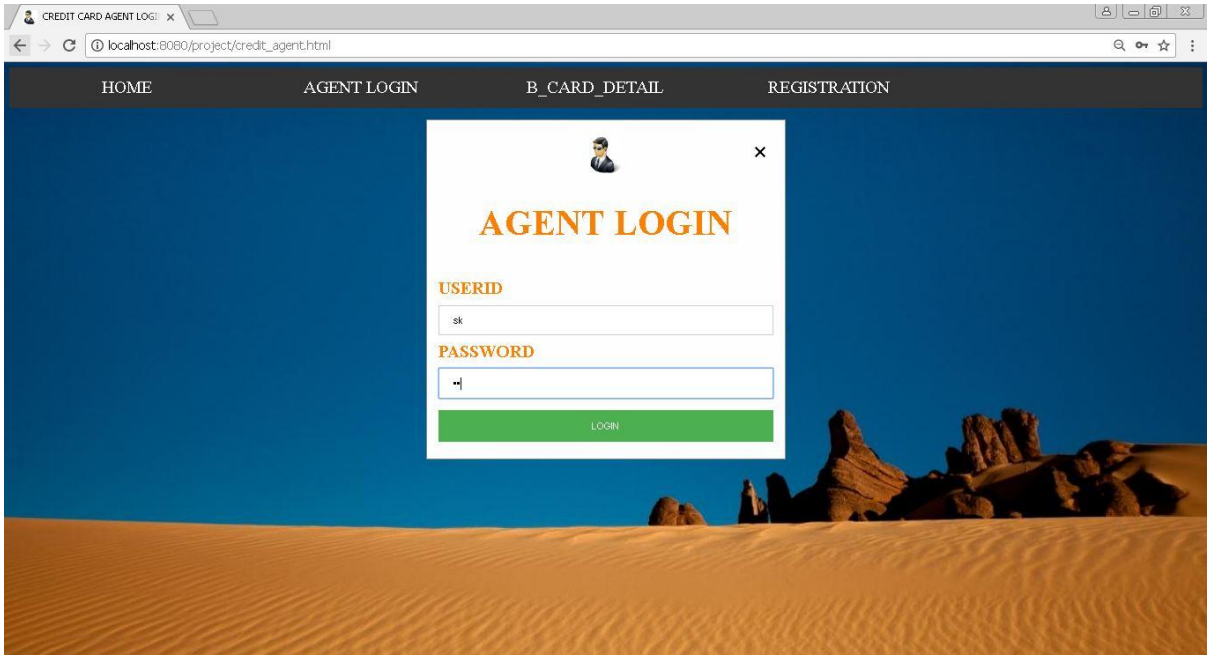
A3. Smart card details registration



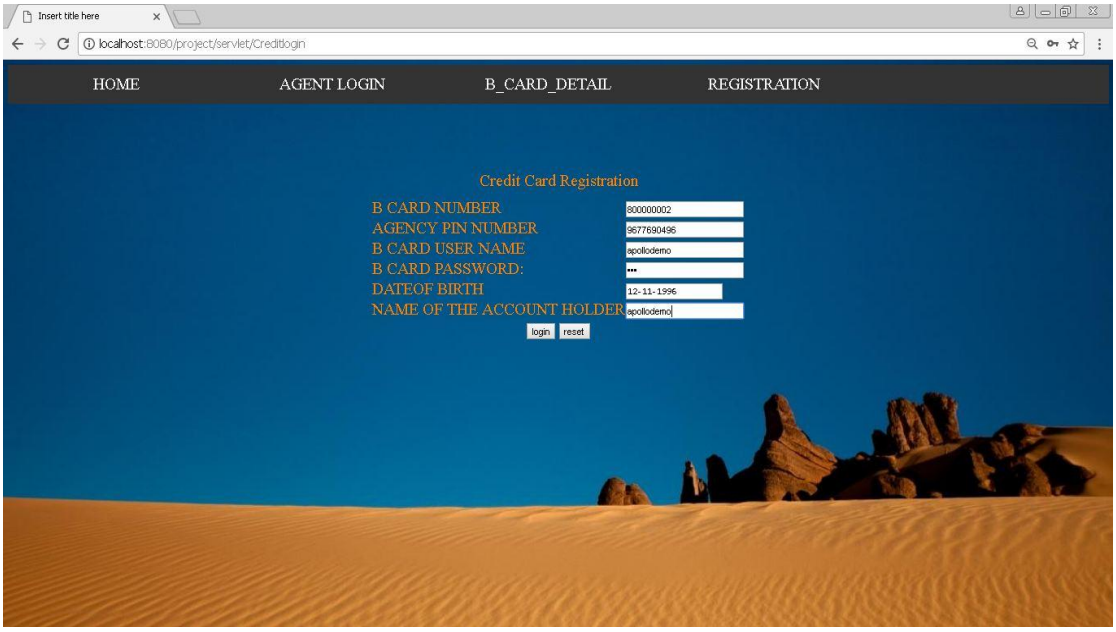
A4. Smart card login



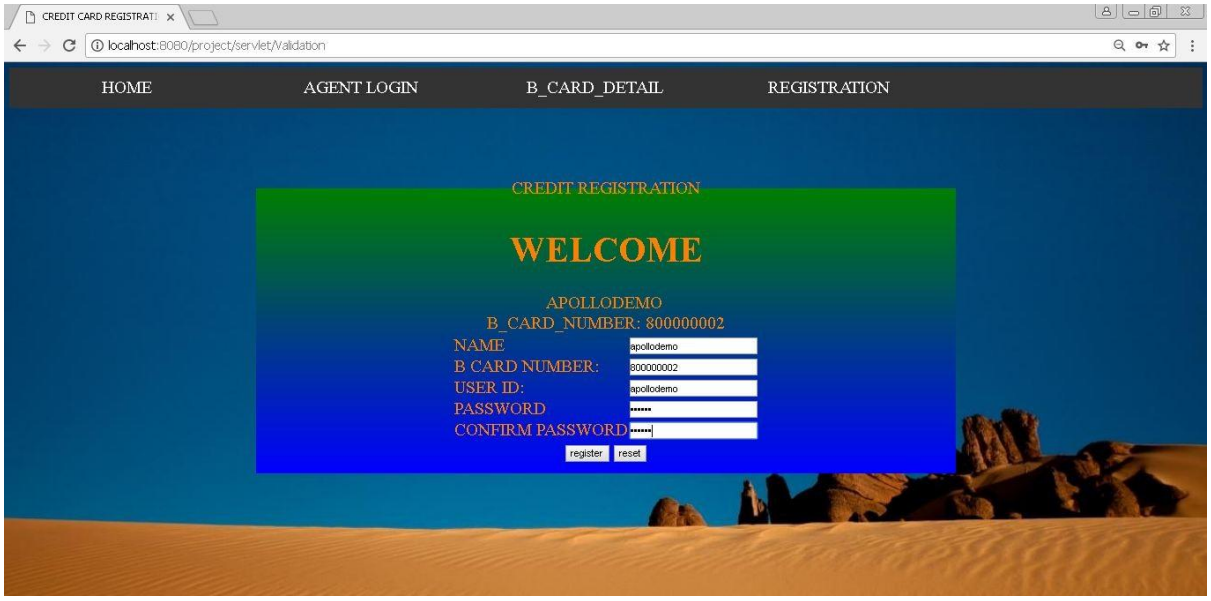
A5. Smart card user bank mini statement



A6. Credit card agent login



A7. User fill the smart card details



A8. Eligible user application

A9. credit card report

REFERENCES

[1] John O. Awoyemi, et al., “Credit card fraud detection using Machine Learning Techniques”, 978-1-5090-4642-3/17

[2] John Richard D. Kho, et al. “Credit Card Fraud Detection Based on Transaction

Behaviour”, 978-1-5090-1134-6/17

[3] Donald E. Brown, et al.” Adversarial Learning in Credit Card Fraud Detection” ,978-1-5386-1848-6/17

[4] Dr John K.C. Kingston, et al.” Representing, Reasoning and Predicting Fraud using Fraud Plans”, 978-1-5090-5476-3/17

[5] John rocker, et al., “Credit card fraud detection using prediction”, 978-1-5090-4645-3/17

[6] Richard, et al. “Credit Card Fraud Detection Based on card usage Behaviour”, 978-1-5011-1134-6/17

[7] Donald Trump, et al.” Medical treatment prediction” ,978-1-5386-1878-6/16

[8] Dr.Gupta , et al.” Diseases prediction ”, 978-1-4090-5976-3/16

[9] Dr Vignesh sharma, et al.” Low level people identification using Aadhar card” ,978-1-5399-1878-9/15

[10] Dr.M.S. Gupta, et al.” Eligible people for the Loan by the bank Statement”, 978-1-4091-5976-3/16