

## FINAL PROJECT TOPIC

**Name:** Demonetization Software

**Domain:** Banking, Logistics, Currency exchange.

**Description:** If a country decides to demonetize its currency, then old currency notes need to be exchanged for the new one. A country can demonetize its currency owing to curb Black Money, to stop Terrorist Activities and to solve the issue of present duplicate currency circulated. The software we are making aims at making this transition of this process smooth all entities involved which include Government, Bank, Logistics (Money Transferring) and common people. By deciding to demonetize the currency there would be a great amount of planning required because of the huge population there would be chaos and long queues near the bank. The bank officials can assign a token to their customers and can ask them prior about the amount of currency they intend to exchange. This will solve the problem of the long queue and save a lot of time for calculating the currency on the spot since data can be collected before from the customer and accordingly currency can be made available. This is just one of the many use-cases. Similarly, intelligence can be applied on the basis of collected data and we will be in a position to answer questions like which city has exchanged a large number of currency or even which person has exchanged a lot of currency.

**Scope of the project:**

1. People will convey their respective Bank branches in advance about their currency exchange requirement. The Bank Manager will collect all the information of all their customers and allot them a time-slot for currency exchange. The customer is expected to go to Bank at that time only with that same amount of old currency.
2. All the Local Branches will then request the Main Federal Bank about the currency requirement.
3. The Federal Bank will know the exact amount of the currency needed for exchange. The federal bank will ask the note printing company to dispatch that much amount of money to it.
4. The Federal Bank will dispatch money to local branches and local branches, in turn, will return the new currency to the customers.

**Conclusion:** As mentioned earlier, the software aims at making the transition of currency and making currency exchange as smooth as possible for all the entities involved. Let's take an example. In India, the 500 rupee note and the 1000-rupee note was demonetised and some time is given for exchanging old currency. One can witness a huge queue outside the banks. Due to the token system, such huge queues can be prevented. Also, we can collect data beforehand from people as to how much currency is to be exchanged. This can make the processing time much faster.

**Functionalities:**

1. Sign up users with different roles
2. User authentication
3. User authorization
4. Making reservations
5. Retrieving reservations
6. Processing reservations
7. Filtering reservations
8. Viewing scheduled reservations
9. User profile editing
10. Session Management

## 11. Preventing unauthorized users

Source Code:

```
package com.demonetization.controller;

import java.util.ArrayList;

import javax.servlet.http.HttpServletRequest;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;
import com.demonetization.dao.*;
import com.demonetization.pojo.*;

/**
 * Handles requests for the application home page.
 */
@Controller
public class MainController {

    @Autowired
    @Qualifier("personDAO")
```

```
PersonDAO personDao;
```

```
@Autowired
```

```
@Qualifier("bankDAO")
```

```
BankDAO bankDao;
```

```
@Autowired
```

```
@Qualifier("notesDAO")
```

```
NotesDAO noteskDao;
```

```
@Autowired
```

```
@Qualifier("userDAO")
```

```
UserDAO userDao;
```

```
@Autowired
```

```
@Qualifier("reservationDAO")
```

```
ReservationDAO reservationDAO;
```

```
@Autowired
```

```
@Qualifier("reservationDetailsDAO")
```

```
ReservationDetailsDAO reservationDetailsDAO;
```

```
/**
```

\* Simply selects the home view to render by returning its name.

\*/

```
@RequestMapping(value = "/createReservation.htm", method =  
RequestMethod.POST)
```

```
public ModelAndView createReservation(HttpServletRequest request) {
```

```
    ModelAndView modelAndView = new ModelAndView();
```

```
    if(request.getSession().getAttribute("userId") != null){
```

```
        Bank bank =
```

```
        bankDao.retrieveBank(Integer.parseInt(request.getParameter("bankId")));
```

```
        User user =
```

```
        userDao.retrieveUser(Integer.parseInt(request.getParameter("id")));
```

```
        Notes notes5 = noteskDao.retrieveNotes(500);
```

```
        Notes notes10 = noteskDao.retrieveNotes(1000);
```

```
        int quantity5 = Integer.parseInt(request.getParameter("five"));
```

```
        int quantity10 =
```

```
        Integer.parseInt(request.getParameter("thousand"));
```

```
ReservationDetailsId reservationDetailsId5 = new  
ReservationDetailsId();
```

```
reservationDetailsId5.setNotesId(notes5.getId());
```

```
reservationDetailsId5.setQuantity(quantity5);
```

```
ReservationDetailsId reservationDetailsId10 = new  
ReservationDetailsId();
```

```
reservationDetailsId10.setNotesId(notes10.getId());
```

```
reservationDetailsId10.setQuantity(quantity10);
```

```
ReservationDetails reservationDetails5= new  
ReservationDetails();
```

```
reservationDetails5.setId(reservationDetailsId5);
```

```
ReservationDetails reservationDetails10= new  
ReservationDetails();
```

```
reservationDetails10.setId(reservationDetailsId10);
```

```
Reservation reservation = new Reservation();
```

```
reservation.setBank(bank);
```

```
reservation.setDate(request.getParameter("date"));
```

```
reservation.setTimeslot(request.getParameter("timeslot"));
```

```
reservation.setUser(user);
```

```
        reservation.setStatus("incomplete");

reservation.getReservationDetailses().add(reservationDetails5);

reservation.getReservationDetailses().add(reservationDetails10);


        user.getReservations().add(reservation);
        userDao.createUser(user);


        reservationDAO.createReservation(reservation);
        reservationDetails5.setResevation(reservation);
        reservationDetails10.setResevation(reservation);


        reservationDetailsId10.setResevationId(reservation.getId());
        reservationDetailsId5.setResevationId(reservation.getId());


reservationDetailsDAO.createReservationDetails(reservationDetails5);

reservationDetailsDAO.createReservationDetails(reservationDetails10);


        modelAndView.setViewName("result_reservation");

    }

    else{
```

```
        modelAndView.setViewName("login");

    }

    return modelAndView;
}
```

```
@RequestMapping(value = "/updateUser.htm", method =
RequestMethod.POST)

public ModelAndView updateUser(HttpServletRequest request) {

    ModelAndView modelAndView = new ModelAndView();

    if(request.getSession().getAttribute("userId") != null){

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

        Person person =
personDao.retrievePerson(Integer.parseInt(request.getParameter("ssn")));

        String result = " ";

        User user =
userDao.retrieveUser(Integer.parseInt(request.getParameter("id")));
```

```

        person.setLastName(request.getParameter("lname"));
        person.setSsn(Integer.parseInt(request.getParameter("ssn")));
        person.setFirstName(request.getParameter("fname"));

        //
        user.setBank(bankDao.retrieveBank(Integer.parseInt(request.getParameter(
("bankId")))));

        user.setEmail(request.getParameter("user_email"));
        user.setPassword(request.getParameter("psw"));
        user.setPerson(person);
        user.setRole(role);

        result = personDao.createPerson(person);
        result += " Also " + userDao.createUser(user);

        if(user.getRole().equalsIgnoreCase("customer")){

            ArrayList<Reservation> reservations =
reservationDAO.retrieveReservationAllforCustomer(user.getId());

            modelAndView.addObject("reservations",reservations);
            modelAndView.setViewName("user-dashboard");
        }

        else if(user.getRole().equalsIgnoreCase("employee")){

```



```
        ArrayList<Reservation> reservations =
reservationDAO.getReservations(user);

        modelAndView.addObject("reservations",reservations);
        modelAndView.setViewName("employee-dashboard");
    }

}

else{

    modelAndView.setViewName("login");

}


return modelAndView;
}
```

```
@RequestMapping(value = "/createUser.htm", method =
RequestMethod.POST)

public ModelAndView createUser(HttpServletRequest request) {
```

```
ModelAndView modelAndView = new ModelAndView();
```

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

```
Person person =  
personDao.retrievePerson(Integer.parseInt(request.getParameter("ssn")));
```

```
String result = " ";
```

```
if(person == null){
```

```
    User user = new User();
```

```
    person = new Person();
```

```
    person.setLastName(request.getParameter("lname"));
```

```
    person.setSsn(Integer.parseInt(request.getParameter("ssn")));
```

```
    person.setFirstName(request.getParameter("fname"));
```

```
//
```

```
    user.setBank(bankDao.retrieveBank(Integer.parseInt(request.getParameter  
("bankId"))));
```

```
    user.setEmail(request.getParameter("user_email"));
```

```
    user.setPassword(request.getParameter("psw"));
```

```
    user.setPerson(person);
```

```
    user.setRole(role);
```

```
    person.getUsers().add(user);
```

```

        result = personDao.createPerson(person);
        result += " Also " + userDao.createUser(user);
    }
    else{
        boolean userExists = false;
        for(User u:person.getUsers()){
            if(u.getRole().equalsIgnoreCase(role)){
                userExists = true;
                modelAndView.setViewName("result");
                modelAndView.addObject("result","User already
exists");

                return modelAndView;
            }
        };

        if(!userExists){
            User user = new User();

            //
            user.setBank(bankDao.retrieveBank(Integer.parseInt(request.getParameter(
"bankId"))));

            user.setEmail(request.getParameter("email"));
            user.setPassword(request.getParameter("psw"));

            user.setPerson(personDao.retrievePerson(Integer.parseInt(request.getPara
meter("personId"))));

```

```

        user.setRole(role);
        person.getUsers().add(user);
        personDao.createPerson(person);
        result = userDao.createUser(user);
    }
}

modelAndView.setViewName("login");
return modelAndView;
}

```

```

@RequestMapping(value = "/login.htm", method = RequestMethod.GET)
public ModelAndView loginget(HttpServletRequest request) {
    ModelAndView modelAndView = new ModelAndView();

    int id =
Integer.parseInt(request.getSession().getAttribute("userId").toString());

    User user = userDao.retrieveUser(id);

    if(user.getRole().equalsIgnoreCase("customer")){

        ArrayList<Reservation> reservations =
reservationDAO.retrieveReservationAllforCustomer(id);

        modelAndView.addObject("reservations",reservations);
        modelAndView.setViewName("user-dashboard");
    }
}

```

```

    }

    else if(user.getRole().equalsIgnoreCase("employee")){

        ArrayList<Reservation> reservations =
reservationDAO.getReservations(user);

        modelAndView.addObject("reservations",reservations);

        modelAndView.setViewName("employee-dashboard");

    }

```

```

        return modelAndView;

    }

```

```

@RequestMapping(value = "/login.htm", method = RequestMethod.POST)
public ModelAndView login(HttpServletRequest request) {

    ModelAndView modelAndView = new ModelAndView();

    int userid = Integer.parseInt(request.getParameter("id"));

    boolean isAuthenticated = userDao.authenticateUser(userid ,
request.getParameter("password"));

    if(isAuthenticated){

```

```
request.getSession().setAttribute("userId", userid);

User user =
userDao.retrieveUser(Integer.parseInt(request.getParameter("id")));

if(user.getRole().equalsIgnoreCase("customer")){
    int id =
Integer.parseInt(request.getSession().getAttribute("userId").toString());

    ArrayList<Reservation> reservations =
reservationDAO.retrieveReservationAllforCustomer(id);

    modelAndView.addObject("reservations",reservations);
    modelAndView.setViewName("user-dashboard");
}
else if(user.getRole().equalsIgnoreCase("employee")){
    ArrayList<Reservation> reservations =
reservationDAO.getReservations(user);

    modelAndView.addObject("reservations",reservations);
    modelAndView.setViewName("employee-dashboard");
}

else if(user.getRole().equalsIgnoreCase("manager"))
    modelAndView.setViewName("manager-home");

else if(user.getRole().equalsIgnoreCase("admin"))
    modelAndView.setViewName("admin-home");
```

```
    }  
    else{  
        modelAndView.setViewName("result");  
        modelAndView.addObject("result","Unauthenticated user.");  
    }  
    return modelAndView;  
}
```

```
@RequestMapping(value = "/", method = RequestMethod.GET)  
public ModelAndView home(HttpServletRequest request) {  
    return new ModelAndView("login");  
}
```

```
@RequestMapping(value = "/logout.htm", method = RequestMethod.GET)  
public ModelAndView logout(HttpServletRequest request) {  
    ModelAndView modelAndView = new ModelAndView();  
    if(request.getSession().getAttribute("userId") != null){  
        modelAndView.setViewName("logout");  
    }  
    else{  
        modelAndView.setViewName("login");  
    }  
    return modelAndView;  
}
```

```
}
```

```
@RequestMapping(value = "/index.htm", method = RequestMethod.GET)
public ModelAndView index(HttpServletRequest request) {
    ModelAndView modelAndView = new ModelAndView();

    request.getSession().invalidate();
    modelAndView.setViewName("login");
    return modelAndView;
}
```

```
@RequestMapping(value = "/signup.htm", method = RequestMethod.GET)
public ModelAndView signup(HttpServletRequest request) {
    ModelAndView modelAndView = new ModelAndView();

    ArrayList<Bank> bankList = bankDao.getAllUsers();

    modelAndView.addObject("bankList", bankList);
    modelAndView.setViewName("signup");
    return modelAndView;
}
```



```
    @RequestMapping(value = "/user-details.htm", method =  
RequestMethod.GET)  
    public ModelAndView user_details(HttpServletRequest request) {  
        ModelAndView modelAndView = new ModelAndView();  
  
        if(request.getSession().getAttribute("userId") != null){  
  
            User user =  
userDao.retrieveUser(Integer.parseInt(request.getSession().getAttribute("userId")  
.toString()));  
//            System.out.println(user.getPerson().getId());  
            Person person = personDao.retrievePerson(user.getPerson().getId());  
            modelAndView.addObject("person",person);  
            modelAndView.addObject("user", user);  
            modelAndView.setViewName("user");  
        }  
        else{  
            modelAndView.setViewName("login");  
        }  
  
        return modelAndView;  
    }  
}
```

```
@RequestMapping(value = "/placeOrder.htm", method =
RequestMethod.GET)

public ModelAndView place_order(HttpServletRequest request) {
    ModelAndView modelAndView = new ModelAndView();
    if(request.getSession().getAttribute("userId") != null){
        ArrayList<Bank> bankList = bankDao.getAllUsers();

        modelAndView.addObject("banks",bankList);
        modelAndView.setViewName("placeOrder");
    }

    else{
        modelAndView.setViewName("login");
    }

    return modelAndView;
}
```

```

    @RequestMapping(value = "/processRequest.htm", method =
RequestMethod.POST)

    public ModelAndView processRequest(HttpServletRequest request) {

        ModelAndView modelAndView = new ModelAndView();

        if(request.getSession().getAttribute("userId") != null){

            Reservation reservation =
reservationDAO.retrieveReservation(Integer.parseInt(request.getParameter("rese
rvation"))));

            User user = reservation.getUser();

            ArrayList<ReservationDetails> reservationDetails =
reservationDetailsDAO.retrieveReservationDetails(reservation.getId());

            modelAndView.addObject("reservation1",reservationDetails.get(0));
            modelAndView.addObject("reservation2",reservationDetails.get(1));
            modelAndView.addObject("reservation",reservation);
            modelAndView.addObject("user",user);
            modelAndView.addObject("reservation",reservation);
            modelAndView.setViewName("processRequest");

        }

        else{

            modelAndView.setViewName("login");

        }

        return modelAndView;

    }

```

```
@RequestMapping(value = "/searchOrder.htm", method =
RequestMethod.GET)

public ModelAndView searchOrder(HttpServletRequest request) {

    ModelAndView modelAndView = new ModelAndView();

    if(request.getSession().getAttribute("userId") != null){

        modelAndView.setViewName("searchOrder");
    }
    else{
        modelAndView.setViewName("login");
    }
    return modelAndView;
}
```

```
@RequestMapping(value = "/searchResult.htm", method =
RequestMethod.POST)

public ModelAndView searchResult(HttpServletRequest request) {

    ModelAndView modelAndView = new ModelAndView();

    if(request.getSession().getAttribute("userId") != null){

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

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

```
        User user =  
userDao.retrieveUser(Integer.parseInt(request.getSession().getAttribute("userId")  
.toString()));
```

```
        ArrayList<Reservation> reservations =  
reservationDAO.getReservations(date,timeslot,user);  
        modelAndView.addObject("reservations",reservations);
```

```
        modelAndView.setViewName("searchResult");
```

```
    }
```

```
    else{
```

```
        modelAndView.setViewName("login");
```

```
    }
```

```
    return modelAndView;
```

```
}
```

```
@RequestMapping(value ="/completeRequest.htm", method =  
RequestMethod.POST)
```

```
public ModelAndView processRequestSearch(HttpServletRequest request) {
```

```
    ModelAndView modelAndView = new ModelAndView();
```

```
        if(request.getSession().getAttribute("userId") != null){

            Reservation reservation =
reservationDAO.retrieveReservation(Integer.parseInt(request.getParameter("rese
rvationid")));

            reservation.setStatus("complete");

            reservationDAO.createReservation(reservation);

            User user =
userDao.retrieveUser(Integer.parseInt(request.getSession().getAttribute("userId")
.toString()));

            ArrayList<Reservation> reservations =
reservationDAO.getReservations(user);

            modelAndView.addObject("reservations",reservations);

            modelAndView.setViewName("employee-dashboard");

        }

        else{

            modelAndView.setViewName("login");

        }

        return modelAndView;

    }

}
```

## Screenshots:

1

Demonetization

Log out

DASHBOARD

USER PROFILE

NEW APPOINTMENT

Your Reservations

ID	DATE	TIME	STATUS	SELECTION
1	2017-04-28	12-1	complete	
2	2017-04-29	11-12	incomplete	
3	2017-04-26	2-3	complete	
4		7-8	incomplete	

1

Demonetization

Log out

DASHBOARD

USER PROFILE

NEW APPOINTMENT

Please fill details

USER ID (DISABLED)

NUMBER OF 500 NOTES:

NUMBER 1000 NOTES:

1

123

211

ENTER A DATE : mm/dd/yyyy

TIMESLOT: "7-8"

BANK: BOFA-1


Make Reservation

 DASHBOARD

 USER PROFILE

 SEARCH ORDER

Reservations

ID	DATE	TIME	STATUS	SELECTION
3	2017-04-26	2-3	complete	

Process