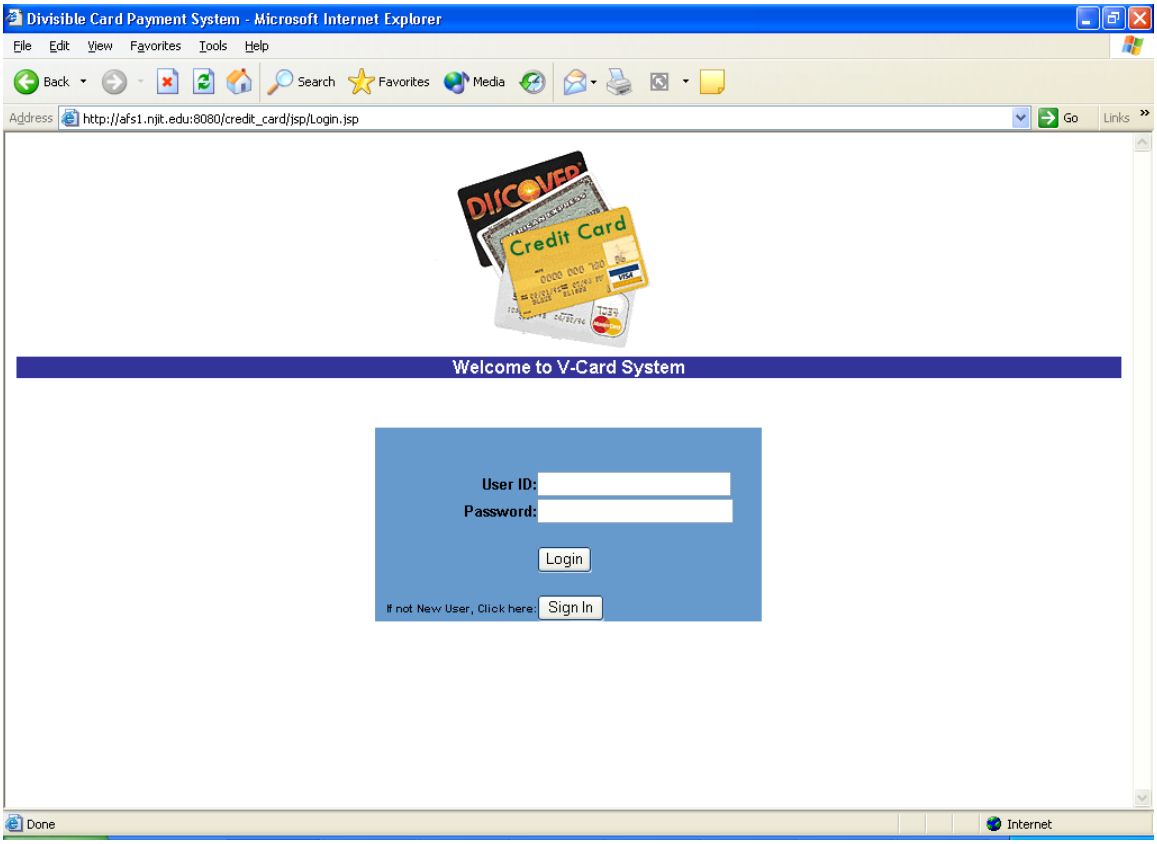
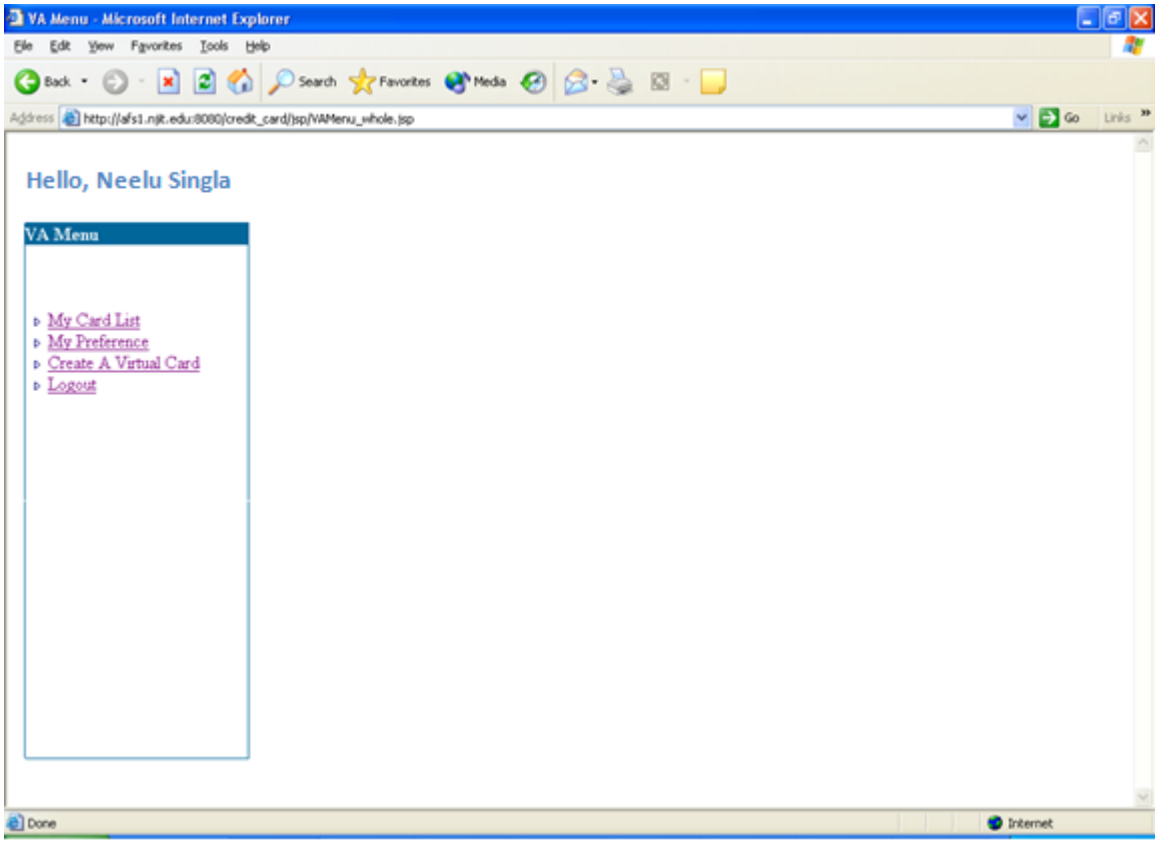
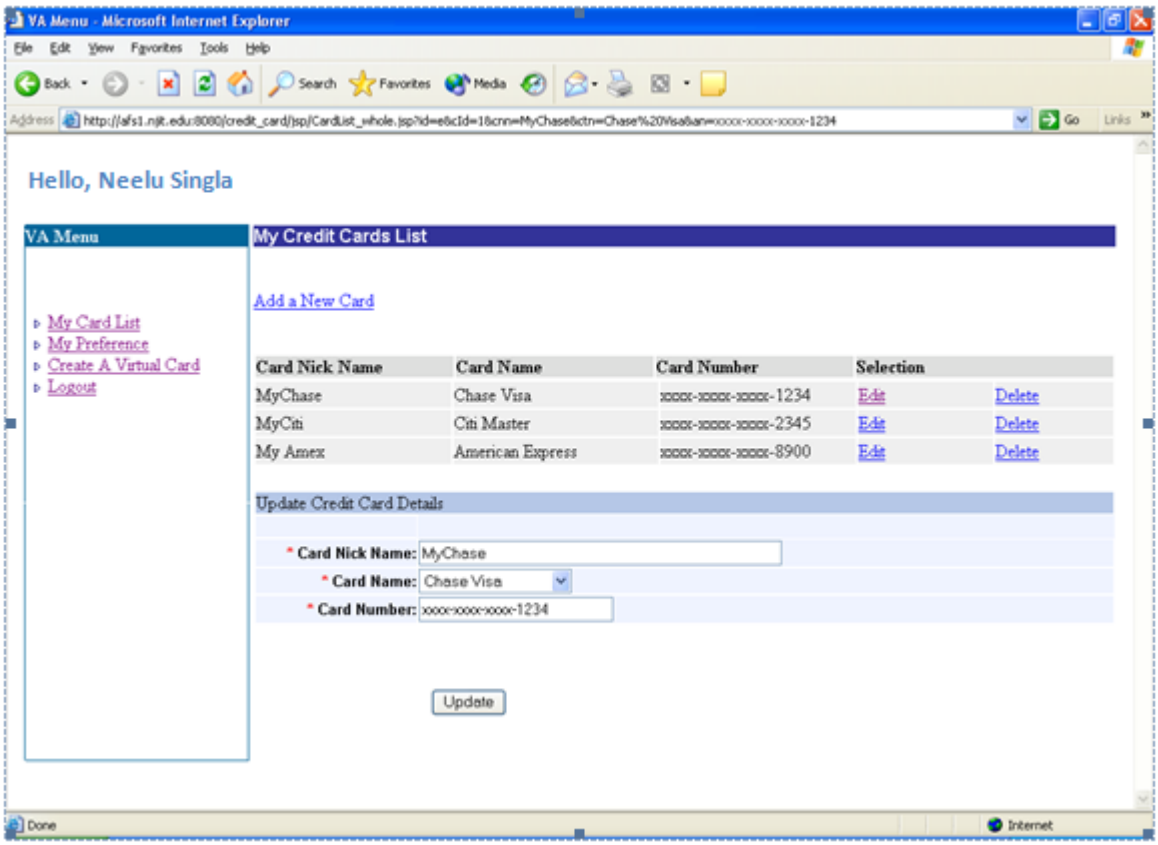
Virtual Credit Card



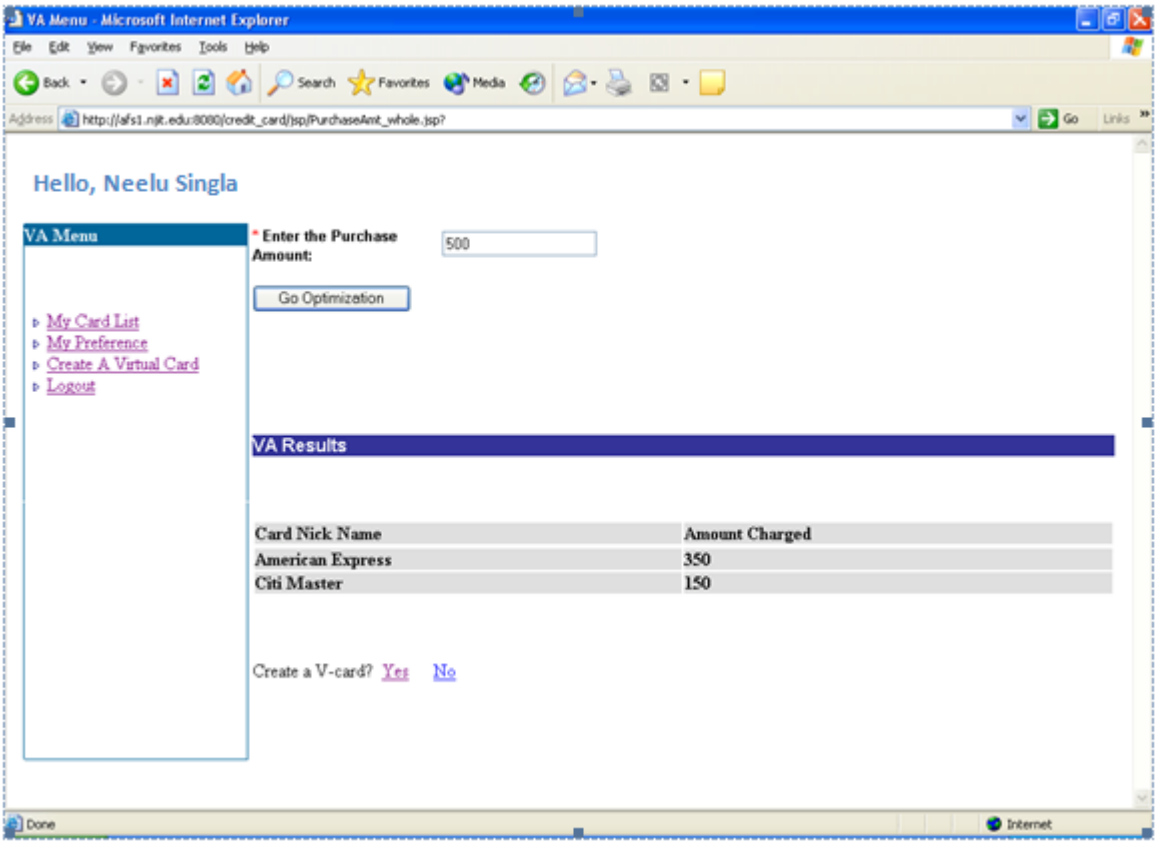
The mission of the project is to provide the customers with a better way of managing their credit cards while minimally modifying the existing infrastructure. As the customers are better off if they can use a combination of cards for a single purchase, the new infrastructure allows the customers to use a combination of different credit cards for a purchase, i.e. divisible card payment. To support the divisible card payments, two modifications are made to the existing infrastructure. First, a software agent called *Virtual Card Agent* (VA) is added to the client side. It is provided to the user with a simple GUI for ease of access. It provides user with three options: 'my card list,' 'my preference,' and 'create a V-card.'



When clicking 'My Card List' the customer sees the list of his cards. The form displays the card nickname and its card number. Clicking the card nickname reveals more detailed information about that card. The interface also provides three functions to add, edit and delete the existing card information in the list.

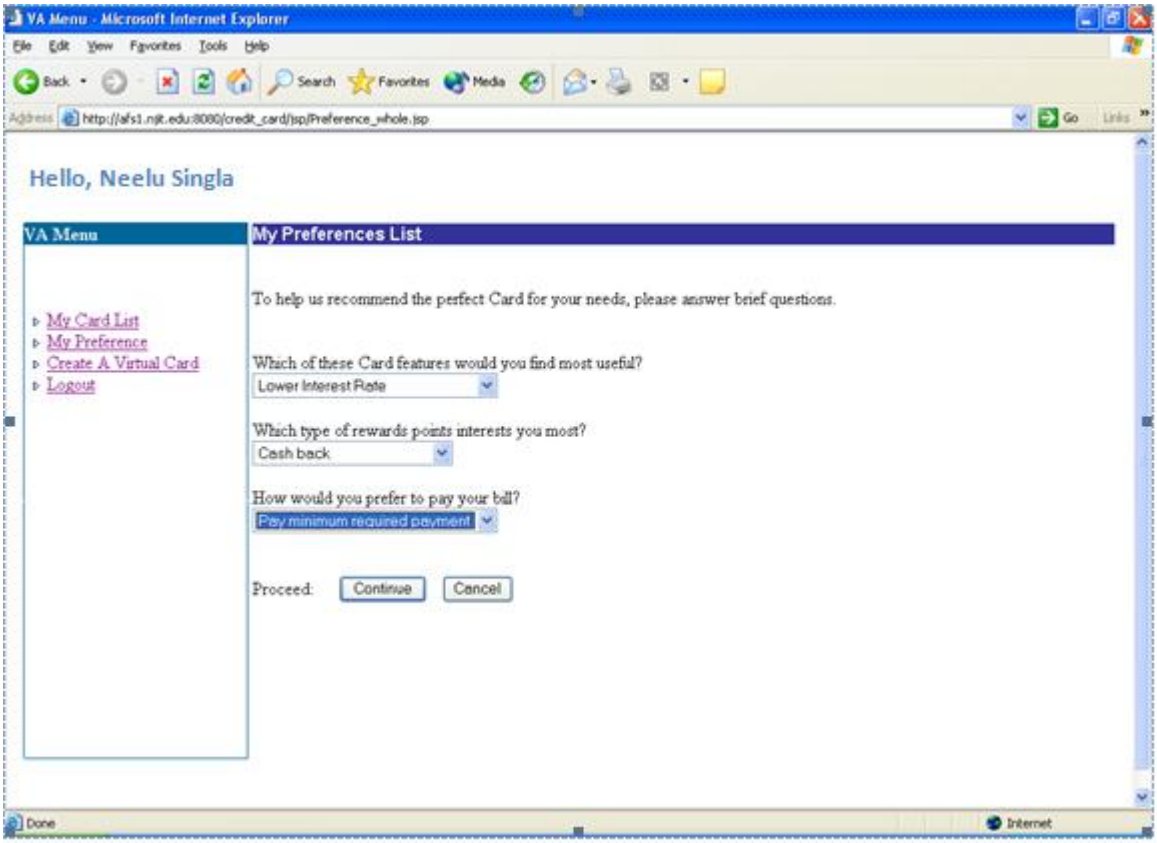


The VA recommends to the customer an optimal combination of credit cards to use. If the customer accepts its suggestion, the VA generates the *Virtual card* (*V-card* in short). As the V-card is used online, no physical card needs to be generated. Instead, the VA generates a unique card number, the amount in the card, and the divisible card billing information. The V-card number is unique to prevent double spending. The method used to generate a unique V-card number, where the V-number is based on the combination of credit card numbers used in the V-card. The V-card number is generated using the first two card numbers with the current timestamp. As each card number is unique to each individual, this simple method is sufficient to guarantee the unique V-card number. The process used to generate V-Card is:-

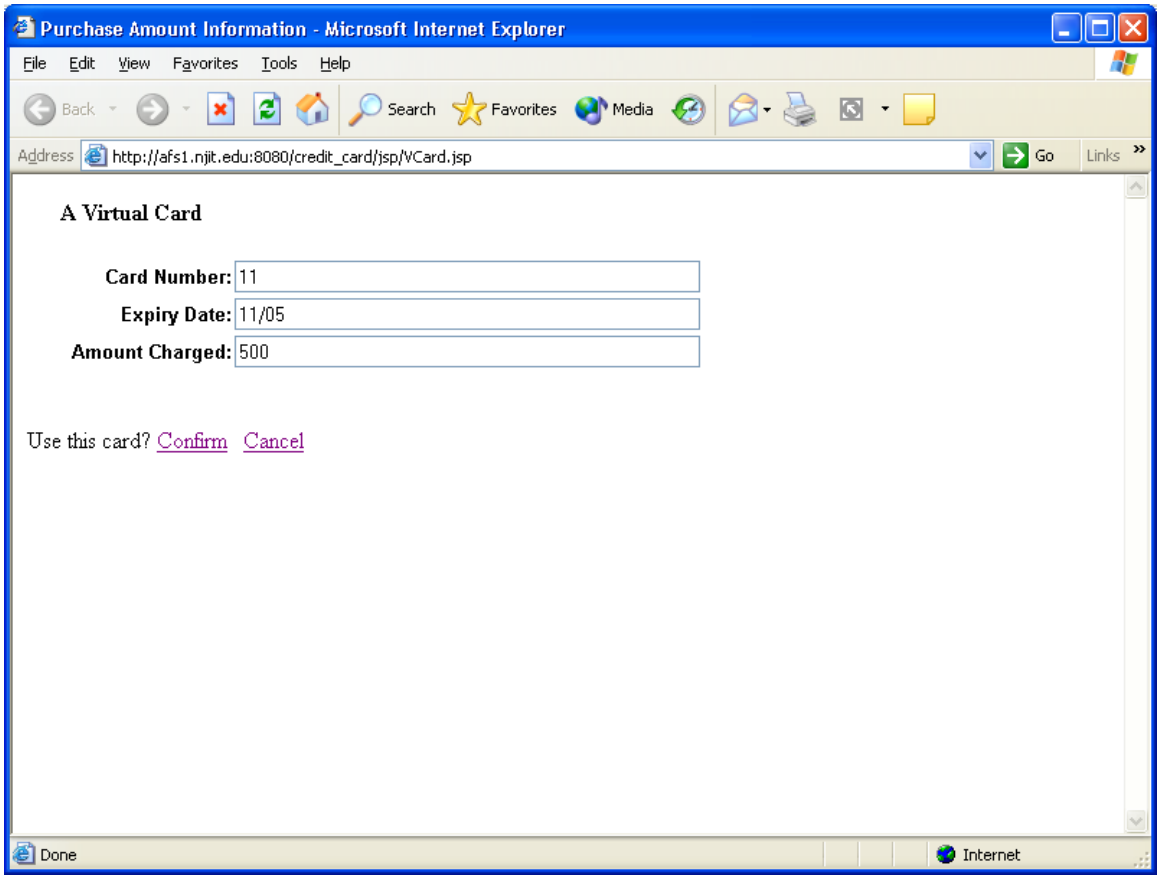


When the customer wants to purchase a product, she enters the purchase amount and pushes the 'Go Optimization' button. For example, the customer enters the purchase amount of 500. This information, in conjunction with the previously entered information about credit cards and preferences, is used to calculate the optimization result. Optimized solution shows a list of cards to be used and the charge against each card. The example shows a list of two cards (out of three cards in the VA database) with their nicknames and the amount charged on each card.

When determining the optimal combination of cards to use, the VA may consider the customer's preferences over various factors such as interest rates, annual fees, mileage bonus, cash-back bonus, on-going promotions, etc. The VA provides the GUI to the customer so that she can easily update her preference profile.



At present, the utility function is computed by approximation based on a series of simple questions.



The final step is to create a V-card. When the customer follows the suggestion (by selecting "Yes" to the question of "Create a V-Card"), the VA creates a one-time use V-card number. The expiration date is set to be the next year from today at present.

Second, special software called a Virtual Card Manager (VCM) is added at the merchant side to handle the V-card payment. When the V-card is up for approval, the VCM decrypts the divisible payment information and forwards the billing information to each card issuer involved in the V-card. Unlike the current protocol that contacts one credit card issuer for approval, the VCM needs to communicate with all the issuing banks involved in a V-card. Each card-issuing bank sends the approval code. When all the approval codes are sent back to VCM, VCM sends back the approval of the V-card to the payment gateway.

I have implemented the Virtual Agent Software and Bank Simulation using the Java Server Pages technology. I have also implemented debugging and error logs for ease of maintenance. For each JSP, the call flow will be logged in the file whose file name is same as the JSP file name. Also if an error gets generated, it will be logged into error file of that day's date for ease. The database used is Oracle to store all the tables pertaining to this application. I also implemented Client Side Validations using JavaScript which checks simple validations and mandatory fields.

I have also implemented the algorithm to generate optimal combinations of credit cards based on some of the user' preferences in any order as follows:

- Lower interest rates
- Airline miles
- Cash-back

Since we don't have access to the real bank servers during the implementation of the project, I have implemented the bank simulation server. For simplicity, a back-end database is created for each bank. For the card issuing banks, when the banks receive the transaction requests, their simulated servers will check in their databases to validate the transactions. After validations, each of the issuing banks will return either a denial message or an approval code back to VCM. For the merchant's Acquiring Bank, it behaves similar to the card issuing banks, except for sending the request to the issuing banks and then collecting the funds from each of them and updating its own database.