

# Project eWallet

Brittney Brown  
Jenny Greenwood

# Table of Contents

<b>Introduction.....</b>	<b>4</b>
<b>Chapter 1) Business Proposal.....</b>	<b>4</b>
<b>Chapter 2) Project Requirements.....</b>	<b>6</b>
1. Introduction.....	6
1.1 Project Requirement Summary.....	6
1.2 Project Review.....	6
2. Use Case Model Overview.....	7
2.1 Credit Card Subsystem Diagram.....	7
2.1.1 Description.....	7
2.1.2 Actors.....	7
2.1.3 Use Cases.....	7
2.1.3-1 eWallet Credit Card Purchase.....	8
2.1.3-2 Add Credit Card Account to eWallet.....	9
2.1.3-3 Remove Credit Card Account from eWallet.....	10
2.2 Stored Value Subsystem Diagram.....	11
2.2.1 Description.....	11
2.2.2 Actors.....	11
2.2.3 Use Cases.....	11
2.2.3-1 eWallet Stored Value Purchase.....	12
2.2.3-2 Add Stored Value to eWallet via Internet.....	13
2.2.3-3 Add Stored Value to eWallet via Bank.....	14
2.2.3-4 View eWallet Stored Value.....	15
2.3 Identification Subsystem Diagram.....	16
2.3.1 Description.....	16
2.3.2 Actors.....	16
2.3.3 Use Cases.....	16
2.3.3-1 Enroll Initial Cardholder Identification.....	17
2.3.3-2 Enroll Cardholder's Driver's License.....	18
2.3.3-3 Verify Driver's License Information.....	19
2.3.3-4 Verify General Identity Information.....	20
2.3.3-5 Modify Identity Information.....	21
3. Summary.....	22
<b>Chapter 3) Platform Specific Architecture.....</b>	<b>23</b>

## **Chapter 4) Design**

1. Introduction.....	24
2. Subsystem Diagram.....	24
3. Subsystem Documentation.....	25
3.1 Subsystem Overview.....	25
1.1 Component Diagram.....	25
1.2 POS Component.....	25
1.2.1 Component Class Diagram.....	26
1.2.2 Class Documentation.....	26
1.2.2.1 Select Payment Screen.....	26
1.2.2.2 Credit Card Payment Screen.....	26
1.2.2.3 Transaction.....	26
1.2.2.4 Credit Card Transaction.....	26
1.2.3 Summary.....	27
1.3 Card Manager Component.....	27
1.3.1 Card Manager Class Diagram.....	27
1.3.2 Class Documentation.....	28
1.3.2.1 Login Screen.....	28
1.3.2.2 Card Option Menu.....	28
1.3.2.3 Applet Option Menu.....	28
1.3.2.4 Credit Card Applet Options.....	28
1.3.3 Summary.....	28
1.4 Card Component.....	29
1.4.1 Card Class Diagram.....	29
1.4.2 Class Documentation.....	29
1.4.2.1 Card.....	29
1.4.2.2 Applet.....	29
1.4.2.3 Credit Card Applet.....	30
1.4.2.4 Credit Card.....	30
1.4.3 Summary.....	30
1.5 Security Component.....	30
1.5.1 Security Class Diagram.....	31
1.5.2 Class Documentation.....	31
1.5.2.1 User Fingerprint.....	31
1.5.2.2 Validation.....	31
1.5.2 Summary.....	31

## **Chapter 5) Summary..... 32**

# **Introduction**

This document serves as a guide to the eWallet project and includes a business vision, project requirements, platform choices and design. With this document we will be able to implement the items contained within. A business vision gives us the underlying idea behind the project. The requirements give detailed use cases for the main functions. The platform choice describes the program development environment. The design document will provide a map for our design. All information within is protected by copyright and any infringement will be persecuted to the fullest extent of the law.

## **Chapter 1** **Business Proposal**

This project falls mostly within Embedded Systems. We are using smart cards. Each card has a small chip embedded (no pun intended) in the plastic. And that chip is an actual microprocessor with ROM and a cryptographic co-processor. The chip runs a trimmed down version of a JVM, and has the ability to store small java applets in ROM that act as runnable programs. Along with designing/implementing the applets themselves we will also implement front end applications to interact with the cards and the applets stored on them.

The project can be thought of as an eWallet of sorts. It entails issuing a smart card with multiple java applets on it. These applets will each handle separate subsystems which include: Identification, Stored Value and Credit Card Transactions. The project also involves developing applications to interact with the applet such as adding/retrieving personal data off the card, or making a purchase. All the information on the card can only be obtained when a user has authenticated them to the card. We will do this with either a pin number or fingerprint verification.

The identification applet will store the person's photograph, address, SSN, birthday, any other personal information, as well as a fingerprint. This applet can act as a driver's license, school id, or other need by which a person wants to identify them. Proof of authenticity can happen not only by comparing a photograph, but also a fingerprint.

The stored value applet will work much the same as our current U student id cards work. Users will be able to put money on their card, and then spend that money at participating vendor locations. Not a credit card transaction, but a stored value transaction. A pin number or fingerprint can be used to authenticate.

The credit card applet will hold information about various credit accounts, and rather than using a credit card, it will use the card number and account information stored on the smart chip. In order to use the credit card account the card holder must authenticate to the system with either a fingerprint or a PIN number.

Along with these applets we will design the systems to interact with them. This includes a system to initialize the card with your personal data, credit accounts, etc, a system to allow a user to prove their identity, a system to allow users to put money on their cards and make purchases with that money, and a system that lets the card handle credit card transactions. This project also includes a card tracking system to handle cards that may have been lost or stolen.

The card will be used by people who don't want to carry numerous cards in their wallet. If a regular wallet is lost several companies must be called to ensure your money and identity is protected. With the smart card only one call is necessary to deactivate the card. It will allow a convenient, one card wallet.

Our motivation for this project is to learn about smart cards and use this for our senior lab. We have worked with the cards a little in the past and feel we have enough experience to get us started. It seems like it could be a fun and interesting project that would most likely be very different from other types of projects. It would provide a convenient and safe way to store and use information you carry on multiple cards in your wallet (ie: driver's license, school id, and credit cards) all on one single card that can only be accessed by authenticating yourself.

# Chapter 2

## Project Requirements

### 1. Introduction: eWallet

Many wallets today are cluttered with several cards, cash and more. Keeping track of all these items can be difficult. The electronic wallet (eWallet) will provide all of the functions of today's wallet on one convenient smart card eliminating the need for several cards. The eWallet will also provides numerous security features not available to regular wallet carriers. Identification is required for every credit card transaction and the card is equipped with a disabling device if the card should be tampered with. These increased security measures and the convenience make this a worthwhile project. The items proposed in this chapter will give detailed descriptions of how to design this eWallet system.

### 1.1 Project Requirement Summary

This chapter provides the use case model and documentation for an electronic wallet (eWallet) with the use of smart card technology. It will provide a use case model overview that will include several subsystem diagrams, a description of each subsystem, use case models for each subsystem and actors specific to each of these subsystems. It also contains definitions for various terms for the system followed by a summary. All systems we plan to implement are defined within this chapter.

### 1.2 Project Review

A credit card function will work much like credit cards do today. The user will select which card they wish to use from a purchase interface which will forward the credit card number to the credit card machine. Refer to section 2.1 for more details.

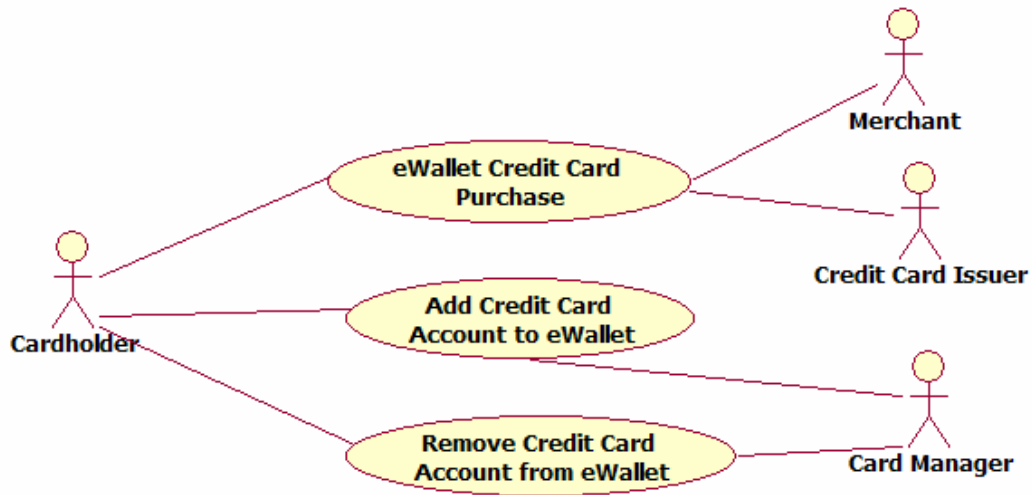
The eWallet will incorporate a "stored value" function that will work like cash, without actually needing to carry paper money and coins. This eliminates the need to carry cash. Refer to section 2.2 for more details.

An identification function will be added to the card that can replace driver's licenses and other forms of identification. Refer to section 2.3 for more details.

An applet installed on the smart card will be responsible for one functional aspect. Therefore, each subsystem is representative of one applet installed on the card.

## 2. Use Case Model Overview

### 2.1 Credit Card Subsystem Diagram



<b>2.1.1 Description</b>	The credit card payment subsystem models how credit card transaction will occur between the Merchants and Cardholders. It also describes how the Cardholder will manage credit card accounts.	
<b>2.1.2 Actors</b>	<b>Name</b>	<b>Goal</b>
	Merchant	Accepts credit card purchases with eWallet.
	Cardholder	Manages credit card accounts on eWallet. Makes credit card purchases.
	Credit Card Issuer	Issues credit card numbers and validates credit card transactions.
	POS Terminal	Guides user through the process of making a purchase
	Card Manager	Guides user through the process of adding credit cards and deleting credit cards.
<b>2.1.3 Use Cases</b>	<ul style="list-style-type: none"> <li>• eWallet Credit Card Purchase</li> <li>• Add Credit Card Account to eWallet</li> <li>• Remove Credit Card Account from eWallet</li> </ul>	

### 2.1.3-1 eWallet Credit Card Purchase

<b>Use Case UC1</b>	<b>eWallet Credit Card Purchase</b>	
<b>Description</b>	Stores credit card information on eWallet that is forwarded to credit card machine for purchase.	
<b>Primary Actors</b>	Cardholder	
<b>Stakeholders and Interests</b>	<p>-Cardholder: Wants to make a purchase using a credit card stored on their eWallet.</p> <p>-Merchant: Wants to authenticate the Cardholder's identity prior to any transaction. Want to accept credit card payment from credit card accounts stored on eWallet</p> <p>-Credit Card Issuer: Wants to verify backend credit card transaction (valid card number, exp date, CCV, etc.)</p>	
<b>Preconditions</b>	Merchant has capability to accept credit card payments and the ability to capture biometric (fingerprint) data.	
<b>Postconditions</b>		
<b>Basic Flow</b>	<b>Step</b>	<b>Action</b>
	1	Cardholder arrives at POS Terminal checkout with goods to purchase. Merchant begins new sale, totals purchase and POS Terminal presents Cardholder with total.
	2	Cardholder inserts eWallet into POS Terminal and is prompted to authenticate by fingerprint.
	3	Cardholder authenticates with fingerprint on device attached to POS Terminal.
	4	POS Terminal prompts user to select payment type.
	5	Cardholder selects "Credit Card" payment type.
	6	POS Terminal presents Cardholder with a list of valid eWallet credit card accounts and Cardholder selects one.
	7	POS Terminal sends credit card information to Credit Card Issuer in backend to verify/complete credit card transaction.
	8	Receipt is printed and POS Terminal returns eWallet to cardholder.
<b>Extensions</b>	<b>Step</b>	<b>Branch Action</b>
	3a	Cardholder fingerprint authentication fails. 3a1. Cardholder will be instructed to authenticate two more times. 3a2. Card will disable after 3 attempts.
	5a	Cardholder does not have a credit card. 5a1. POS Terminal will notify and return to step 4.
	7a	Credit Card Issuer cannot verify credit card transaction. 7a1. POS Terminal will return to step 4 and notify failure.

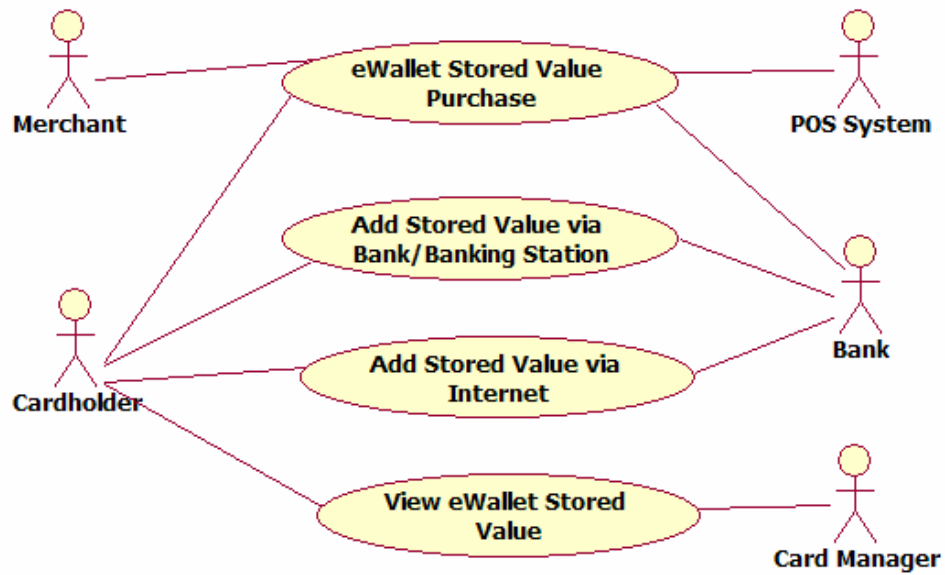
### 2.1.3-2 Add Credit Card Account to eWallet

<b>Use Case UC2</b>	<b>Add Credit Card Account to eWallet</b>	
<b>Description</b>	Allows a Cardholder to add a new credit card to eWallet.	
<b>Primary Actors</b>	Cardholder	
<b>Stakeholders and Interests</b>	<p>-Cardholder: Wants to add one or more credit card accounts to their eWallet</p> <p>-Card Manager: Wants to verify Cardholder's identity. Wants to facilitate the addition of a new eWallet credit card account.</p>	
<b>Preconditions</b>	Cardholder has a valid credit card account to add.	
<b>Postconditions</b>	All requested credit card accounts have been added to the eWallet.	
<b>Basic Flow</b>	<b>Step</b>	<b>Action</b>
	1	Cardholder inserts eWallet into Card Manager and is prompted to authenticate by fingerprint.
	2	Cardholder authenticates with fingerprint on device attached to Card Manager.
	3	Card Manager presents Cardholder with eWallet options. Cardholder chooses to add new credit card account.
	4	Card Manager prompts Cardholder for account type, card number, CCV and expiration date. Cardholder enters requested information.
	5	Card Manager prompts Cardholder to save information and Cardholder saves.
	6	Card Manager verifies account information has been successfully saved and presets cardholder with a list of all accounts on eWallet.
	7	Cardholder ends Card Manger session.
<b>Extensions</b>	<b>Step</b>	<b>Branch Action</b>
	2a	Cardholder fingerprint authentication fails. 2a1. Cardholder will be instructed to authenticate two more times. 2a2. Card will disable after 3 attempts.
	5a	Card information does not successfully save. 5a1. Card Manager notifies Cardholder of failure

### 2.1.3-3 Remove Credit Card Account on eWallet

<b>Use Case UC3</b>	<b>Remove Credit Card Account on eWallet</b>	
<b>Description</b>	Allow Cardholder to remove and existing credit card from eWallet	
<b>Primary Actors</b>	Cardholder	
<b>Stakeholders and Interests</b>	<p>-Cardholder: Wants to remove a credit card account that is installed on their eWallet.</p> <p>-Card Manager: Wants to facilitate the removal of a credit card account on a eWallet as well as verify identity of Cardholder prior to account changes.</p>	
<b>Preconditions</b>	Cardholder has eWallet credit card account to remove.	
<b>Post conditons</b>	Credit card account is no longer stored on eWallet.	
<b>Basic Flow</b>	<b>Step</b>	<b>Action</b>
	1	Cardholder inserts eWallet into Card Manager and is prompted to authenticate by fingerprint.
	2	Cardholder authenticates with fingerprint on device attached to Card Manager.
	3	Card Manager presents Cardholder with eWallet options. Cardholder chooses to remove existing credit card account.
	4	Card Manager presets Cardholder with list of all eWallet credit card accounts. Cardholder selects which account to remove.
	5	Card Manager verifies account information has been removed and presets cardholder with a list of all accounts on eWallet.
	6	Cardholder ends card management session.
<b>Extensions</b>	<b>Step</b>	<b>Branch Action</b>
	2a	Cardholder fingerprint authentication fails. 2a1. Cardholder will be instructed to authenticate two more times. 2a2. Card will disable after 3 attempts.
	3a	eWallet has no credit cards 3a1. Card Manager will notify Cardholder that no credit cards exist.

## 2.2 Stored Value Subsystem Diagram



<b>2.2.1</b> <b>Description</b>	Stored Value acts just like carrying real cash in your wallet. It stores an integer value that represents a cash amount. This integer value can be transferred from a cardholder to a merchant like real cash. A cardholder can continually add to the integer value.	
<b>2.2.2</b> <b>Actors</b>	<b>Name</b>	<b>Goal</b>
	Merchant	Accepts stored value purchases with eWallet. Deposits stored value into own bank account.
	Cardholder	Manages stored value on eWallet. Makes stored value purchases.
	Bank	Transfers money onto eWallet.
	POS Terminal	Guides user through the process of making a purchase.
	Card Manager	Guides user through the process of adding stored value money to eWallet.
<b>2.2.3</b> <b>Use Cases</b>	<ul style="list-style-type: none"> <li>• eWallet Stored Value Purchase</li> <li>• Add Stored Value to eWallet via Internet</li> <li>• Add Stored Value to eWallet via Bank/Banking Station</li> <li>• View eWallet Stored Value</li> </ul>	

### 2.2.3-1 eWallet Stored Value Purchase

<b>Use Case UC4</b>	<b>eWallet Stored Value Purchase</b>	
<b>Description</b>	Allows for a “cash like” purchase by removing money from eWallet and depositing into Merchant account.	
<b>Primary Actors</b>	Cardholder	
<b>Stakeholders and Interests</b>	<p>-Cardholder: Wants to pay for a purchase using Stored Value (money) from their eWallet.</p> <p>-Merchant: Wants to receive payment from the Cardholder. Wants to accept stored value payments via their POS Terminal. Wants to deposit stored value into their own bank account</p> <p>-POS Terminal: Wants to facilitate eWallet transactions and authenticate the cardholder.</p>	
<b>Preconditions</b>	Merchant has capability to accept stored value payments and the ability to capture biometric (fingerprint) data	
<b>Basic Flow</b>	<b>Step</b>	<b>Action</b>
	1	Cardholder arrives at POS Terminal checkout with goods to purchase. Merchant begins new sale, totals purchase and POS Terminal presents Cardholder with total.
	2	Cardholder inserts eWallet into POS Terminal and is prompted to authenticate by fingerprint.
	3	Cardholder authenticates with fingerprint on device attached to POS Terminal.
	4	POS Terminal prompts user to select payment type.
	5	Cardholder selects “Stored Value” payment type.
	6	POS Terminal prompts for confirmation of payment and Cardholder confirms.
	7	POS Terminal successfully completes transaction by removing amount on eWallet and depositing on Merchant account.
	8	Receipt is printed and POS Terminal returns eWallet to cardholder.
	9	Merchant deposits accrued stored value into their own bank account.
<b>Extensions</b>	<b>Step</b>	<b>Branch Action</b>
	3a	Cardholder fingerprint authentication fails. 3a1. Cardholder will be instructed to authenticate two more times. 3a2. Card will disable after 3 attempts.
	5a	Cardholder does not have any stored value money. 5a1. POS Terminal will notify and return to step 4.
	6a	Cardholder cancels 6a1. POS Terminal will return to step 4.
	7a	POS Terminal fails 7a1. Notify Cardholder and returns to step 4.

### 2.2.3-2 Add Stored Value to eWallet via Internet

<b>Use Case UC5</b>	<b>Add Stored Value to eWallet via Internet</b>	
<b>Description</b>	Allows Cardholder to add stored value money onto their eWallet over the internet.	
<b>Primary Actors</b>	Cardholder	
<b>Stakeholders and Interests</b>	<p>-Cardholder: Wants to add Stored Value (money) to their eWallet so they may spend at participating vendors.</p> <p>-Card Manager: Wants to authenticate cardholder to their bank account and to their eWallet</p> <p>-Bank: Wants to securely transfer funds from the cardholder's bank account to the cardholder's eWallet.</p>	
<b>Preconditions</b>	Cardholder has Internet access and Card Manager is configured with bank account information.	
<b>Postconditions</b>	Requested amount of funds has been transferred from bank balance to stored value balance and new balances have been presented to the Cardholder.	
<b>Basic Flow</b>	<b>Step</b>	<b>Action</b>
	1	Cardholder inserts eWallet into Card Manager and is prompted to authenticate by fingerprint.
	2	Cardholder authenticates with fingerprint on device attached to Card Manager.
	3	Card Manager prompts user to select an action.
	4	Cardholder selects "Add Funds via Internet Banking"
	5	Card Manager acquires banking information from Bank and presents Cardholder with bank account information with available funds.
	6	Card Manager prompts Cardholder to enter an amount to transfer onto eWallet
	7	Cardholder enters desired amount to transfer and confirms amount.
	8	Bank transfers funds from bank account to eWallet, deducting funds from current account balance.
	9	Card Manager gives confirmation of new bank account balance and new eWallet stored value.
	10	Cardholder exits Card Manager.
<b>Extensions</b>	<b>Step</b>	<b>Branch Action</b>
	2a	Cardholder fingerprint authentication fails. 2a1. Cardholder will be instructed to authenticate two more times. 2a2. Card will disable after 3 attempts.
	5a	Connection to Bank fails 5a1. Card Manager notifies that a failure occurred
	7a	Cardholder enters amount larger than amount available. 7a1. Card Manager says that funds are not available.

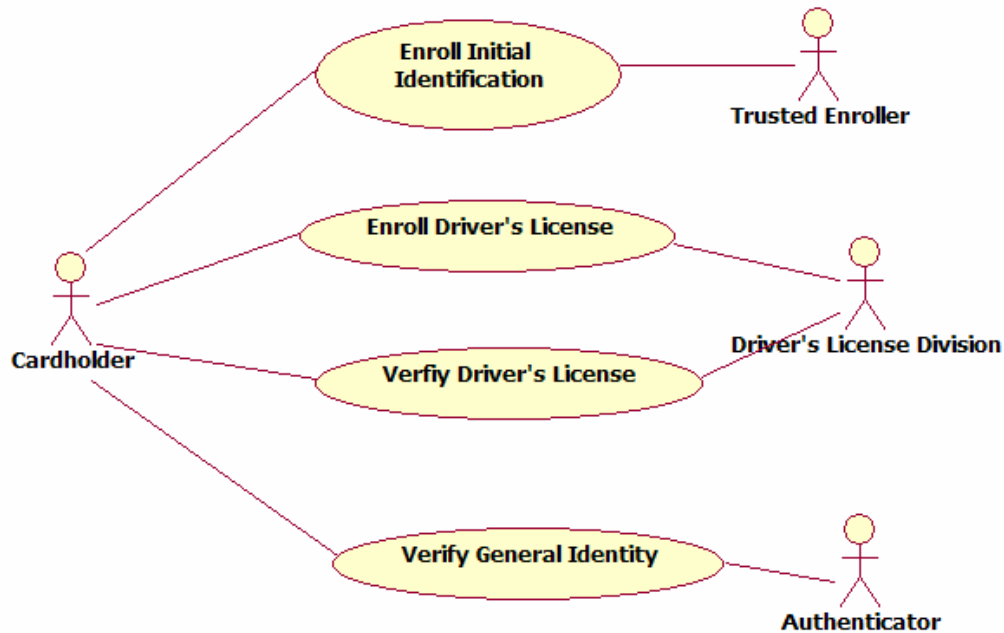
### 2.2.3-3 Add Stored Value to eWallet via Bank/Banking Station

<b>Use Case UC6</b>	<b>Add Stored Value to eWallet via Bank/Banking Station</b>	
<b>Description</b>	Allows Cardholder to add money onto the eWallet at the Bank.	
<b>Primary Actors</b>	Cardholder	
<b>Stakeholders and Interests</b>	-Cardholder: Wants to add stored value to their eWallet. -Bank: Wants to authenticate Cardholder. Wants to transfer money from cardholder's account onto their eWallet.	
<b>Preconditions</b>	Cardholder is at Bank or Banking station	
<b>Postconditions</b>	Funds were successfully and securely transferred from bank account to eWallet.	
<b>Basic Flow</b>	<b>Step</b>	<b>Action</b>
	1	Cardholder inserts eWallet into Bank and is prompted to authenticate by fingerprint.
	2	Cardholder authenticates with fingerprint.
	3	Bank prompts Cardholder to communicate an amount to transfer onto eWallet.
	4	Cardholder states desired amount to transfer.
	5	Bank transfers funds from bank account to eWallet, deducting funds from current account balance.
	6	Bank gives confirmation of new bank account balance and new eWallet stored value.
	7	Cardholder leaves bank/banking station.
<b>Extensions</b>	<b>Step</b>	<b>Branch Action</b>
	2a	Cardholder fingerprint authentication fails. 2a1. Cardholder will be instructed to authenticate two more times. 2a2. Card will disable after 3 attempts.
	4a	Cardholder states amount larger than amount available. 7a1. Bank notifies that funds are not available.

### 2.2.3-4 View eWallet Stored Value

<b>Use Case UC7</b>	<b>View eWallet Stored Value</b>	
<b>Description</b>	Allows the Cardholder to view how much money they have on their eWallet.	
<b>Primary Actors</b>	Cardholder	
<b>Stakeholders and Interests</b>	<p>-Cardholder: Wants to view stored value balance.</p> <p>-Card Manager: Wants to authenticate cardholder before allowing them to view their stored value balance. Wants to tell Cardholder their stored value balance.</p>	
<b>Preconditions</b>		
<b>Postconditions</b>	Cardholder was successfully given eWallet current stored value balance.	
<b>Basic Flow</b>	<b>Step</b>	<b>Action</b>
	1	Cardholder inserts eWallet into Card Manager and is prompted to authenticate by fingerprint.
	2	Cardholder authenticates with fingerprint on device attached to Card Manager.
	3	Card Manager presents Cardholder with eWallet options. Cardholder chooses to view current stored value balance.
	4	Card Manager reports to Cardholder the current stored value amount.
	5	Cardholder ends Card Manger session.
<b>Extensions</b>	<b>Step</b>	<b>Branch Action</b>
	2a	<p>Cardholder fingerprint authentication fails.</p> <p>2a1. Cardholder will be instructed to authenticate two more times.</p> <p>2a2. Card will disable after 3 attempts.</p>

## 2.3 Identification Subsystem Diagram



<b>2.3.1 Description</b>	The identification subsystem stores any form of identification whether it be driver's license, school id or other identification card. It allows several people to view this information upon request.	
<b>2.3.2 Actors</b>	<b>Name</b>	<b>Goal</b>
	Trusted Enroller	Verifies Cardholders credential during initial issuing and enrollment.
	Cardholder	Validates own identity.
	Authenticator	Validates cardholder identity for some general purpose.
	DMV	Enrolls and verifies drivers license on eWallet
	Policeman	Validates driver's license.
<b>2.3.3 Use Cases</b>	<ul style="list-style-type: none"> <li>• Enroll Initial Cardholder Identification</li> <li>• Enroll Cardholder's Driver's License</li> <li>• Verify Driver's License Information</li> <li>• Verify General Identity Information</li> <li>• Modify Identity Information</li> </ul>	

### 2.3.3-1 Enroll Initial Cardholder Identification

<b>Use Case UC8</b>	<b>Enroll Initial Cardholder Identification</b>	
<b>Description</b>	Allows Cardholder to initially get set up with their eWallet.	
<b>Primary Actors</b>	Cardholder	
<b>Stakeholders and Interests</b>	<p>-Cardholder: Wants to get a eWallet. Wants to add their personal information on eWallet as well as store photo and biometric data to be used for authentication with other applications.</p> <p>-Trusted Enroller: Wants to validate cardholder information as it is stored on eWallet as well as facilitate the enrollment.</p>	
<b>Preconditions</b>		
<b>Postconditions</b>	Cardholder's identification data has been successfully saved onto eWallet.	
<b>Basic Flow</b>	<b>Step</b>	<b>Action</b>
	1	Cardholder approaches Trusted Enroller.
	2	Trusted Enroller asks for Cardholder's Social Security Card, valid birth certificate and photo identification.
	3	Trusted Enroller verifies information given.
	4	Trusted Enroller takes new photo and captures fingerprint.
	5	Trusted Enroller gets and enters name, current address, phone number, ssn, birth date, hair color, eye color, weight.
	6	Trusted Enroller asks Cardholder to create a pin number to be used with eWallet.
	7	Cardholder verifies all data to be used.
	8	Trusted Enroller inserts eWallet into Trusted Enroller station and writes all identification data to the eWallet.
	9	Trusted Enroller prints photo, name, address, birth date and stamp of authenticity on eWallet.
	10	Trusted Enroller issues eWallet to cardholder and does not save any back up of data.
	11	Identification Enrollment ends.
<b>Extensions</b>	<b>Step</b>	<b>Branch Action</b>
	3a	Trusted Enroller refuses enrollment to Cardholder 3a1. Cardholder goes to step 11.
	7a	Cardholder information is incorrect and needs to be re-entered. 7a1. Trusted Enroller returns to step 4.
	8a	Writing of identification data fails. 8a1: Trusted Enroller is notified of error. 8a2: Trusted Enroller verifies integrity of card for replacement 8a3: Trusted Enroller returns to step 8.

### 2.3.3-2 Enroll Cardholder's Driver's License

<b>Use Case UC9</b>	<b>Enroll Cardholder's Driver's License</b>	
<b>Description</b>	Allows Cardholder to store driver's license on eWallet.	
<b>Primary Actors</b>	Cardholder	
<b>Stakeholders and Interests</b>	<p>-Cardholder: Wants to have their driver's license added to their eWallet so that it will be accessible for viewing.</p> <p>-DMV: Wants to verify driver's license information and facilitate adding this information to eWallet</p>	
<b>Preconditions</b>		
<b>Postconditions</b>		
<b>Basic Flow</b>	<b>Step</b>	<b>Action</b>
	1	Cardholder approaches DMV requesting to add their driver's license to eWallet.
	2	Cardholder presents eWallet to DMV and DMV inserts eWallet into DMV device and prompts Cardholder to authenticate by fingerprint.
	3	Cardholder authenticates with fingerprint on DMV device.
	4	If available Cardholder presents current driver's license and DMV looks up current license information.
	5	DMV installs Cardholder license information onto eWallet and returns eWallet to Cardholder.
	6	Cardholder license enrollment ends.
<b>Extensions</b>	<b>Step</b>	<b>Branch Action</b>
	2a	Cardholder fingerprint authentication fails. 2a1. Cardholder will be instructed to authenticate two more times. 2a2. Card will disable after 3 attempts.
	3a	License information lookup fails. Cardholder has no current driver's license. 3a1: DMV declines Cardholder of license enrollment. 3a2: Cardholder goes to step 6.
	4a	Installation of license information fails 4a1: DMV is presented with error.

### 2.3.3-3 Verify Driver's License Information

<b>Use Case UC10</b>	<b>Verify Driver's License Information</b>	
<b>Description</b>	Allows DMV to view driver's license on eWallet.	
<b>Primary Actors</b>	Cardholder	
<b>Stakeholders and Interests</b>	<p>-Cardholder: Wants to use eWallet as a way to present their driver's license.</p> <p>-Policeman: Wants to verify driver's license.</p> <p>-DMV: Wants to verify driver's license.</p>	
<b>Preconditions</b>	DMV member has made initial contact with the Cardholder. Policeman is capable of reading eWallet.	
<b>Postconditions</b>	DMV member has successfully determined the validity of Cardholder's driver's license.	
<b>Basic Flow</b>	<b>Step</b>	<b>Action</b>
	1	Cardholder is requested to present driver's license by Policeman.
	2	Cardholder presents eWallet.
	3	Policeman inserts eWallet into DMV handheld device.
	4	Driver's license information is displayed on DMV handheld device.
	5	Policeman performs verification of drivers' license information and returns eWallet to Cardholder.
	6	Policeman completes any necessary actions such as ticketing the Cardholder.
<b>Extensions</b>	<b>Step</b>	<b>Branch Action</b>

### 2.3.3-4 Verify General Identity Information

<b>Use Case UC11</b>	<b>Verify General Identity Information</b>	
<b>Description</b>	Allows identification for general purpose.	
<b>Primary Actors</b>	Cardholder	
<b>Stakeholders and Interests</b>	-Cardholder: Wants to use eWallet as a way to present their driver's license. -Authenticator: Wants Cardholder to verify their identity by authenticating to the eWallet.	
<b>Preconditions</b>		
<b>Postconditions</b>	Cardholder has verified their identity through their fingerprint	
<b>Basic Flow</b>	<b>Step</b>	<b>Action</b>
	1	Authenticator request authentication from Cardholder.
	2	Cardholder presents eWallet to Authenticator.
	3	Authenticator inserts eWallet into Authenticator device and prompts cardholder for fingerprint.
	4	Cardholder gives fingerprint scan and Authenticator compares fingerprint on eWallet.
	5	Authenticator request further identity information from eWallet.
	6	Authenticator returns eWallet to Cardholder.
<b>Extensions</b>	<b>Step</b>	<b>Branch Action</b>
	4a	Fingerprints don't match 4a1. Authenticator notifies there is no match.

### 2.3.3-5 Modify Identity Information

<b>Use Case UC12</b>	<b>Modify Identity Information</b>	
<b>Description</b>	Allows identity information to be changed on eWallet.	
<b>Primary Actors</b>	Cardholder	
<b>Stakeholders and Interests</b>	<p>-Cardholder: Wants to use change personal information stored on eWallet such as address, fingerprint, photo, exp date.</p> <p>- Trusted Enroller: Wants to verify information change and have Cardholder prove identity and validity of information being changed.</p>	
<b>Preconditions</b>		
<b>Postconditions</b>	Cardholder identity information has been successfully changed.	
<b>Basic Flow</b>	<b>Step</b>	<b>Action</b>
	1	Cardholder requests Trusted Enroller change identification information on eWallet.
	2	Cardholder presents eWallet to Trusted Enroller and Trusted Enroller inserts eWallet into Trusted Enroller device and prompts Cardholder to authenticate by fingerprint.
	3	Cardholder authenticates with fingerprint on Trusted Enroller device.
	4	Cardholder conveys changes and Trusted Enroller verifies and captures/enters any new data.
	5	Trusted Enroller verifies changes by reporting back to Cardholder.
	6	Cardholder confirms changes and Trusted Enroller saves changes.
	7	Trusted Enroller returns eWallet to Cardholder.
<b>Extensions</b>	<b>Step</b>	<b>Branch Action</b>
	3a	Cardholder fingerprint authentication fails. 3a1. Cardholder will be instructed to authenticate two more times. 3a2. Card will disable after 3 attempts.
	6a	Card information does not successfully save 5a1. Trusted Enroller receives error message.

### **3. Summary**

The purpose of eWallet is to consolidate various cards in a wallet down to one card which requires authentication for any kind of access or transaction.

Moving all wallet type transactions on one card reduces clutter and increases security. If the eWallet is lost or stolen not only is it difficult to access by the thief but only one call instead of several is required to de-activate it. The system is designed entirely around security and any kind of modification to the card will require some kind of authentication. The subsystem diagrams, descriptions, use case models and actors specific to each of these subsystems will all be implemented as described in this chapter.

## Chapter 3

# Platform Specific Architecture

The eWallet project will use the J2EE platform. This platform provides a good environment for developing in Java. The J2EE platform has a set of frameworks for JSP/Servlets, Java Beans and various other frameworks. The client platform Java provides Swing, SWT, ActiveX and Windows Forms. Many of these features make this a good platform for our project.

The project will contain several web applications. Servlets will play an important role in our web application development. Java servlets technology provides web development with a simple, consistent mechanism for extending the functionality of a web server and for accessing existing business systems. A servlet can almost be thought of as an applet that runs on the server side. Java servlets make many web applications possible.

Client applications will be done in Swing which encompasses a group of features for building graphical user interface and adding rich graphics, functionality and interactivity to Java applications.

Java will provide many of the important components and libraries for interactions with the smart card and card readers. IBM wrote the smart card operating system and provides the Java libraries necessary to communicate with it. They also provide PCSC for the card readers. These IBM Java components and libraries are absolutely necessary for the project to be functional.

The smart card runs a Java Virtual Machine and the front end application runs applets as developed by IBM. Applets are written in the Java programming language. When you use a Java applet, the applet's code is transferred to the card reader to be executed by the card reader's Java Virtual Machine. The information on the smart card must be transferred to the Java Virtual Machine to be functional. These very important features can only work in Java.

The development environment will be done in Eclipse. Eclipse is a kind of universal tool platform. It's an open extensible IDE for anything and nothing in particular. Tools are provided as plug-ins in Eclipse that help to debug smart card applets and applications. This is a very helpful feature that will assist throughout development.

The third party components and libraries we will be using include the IBM JCOP smart card libraries, pure Java libraries for smart card readers, smart card printer drivers and digital camera drivers.

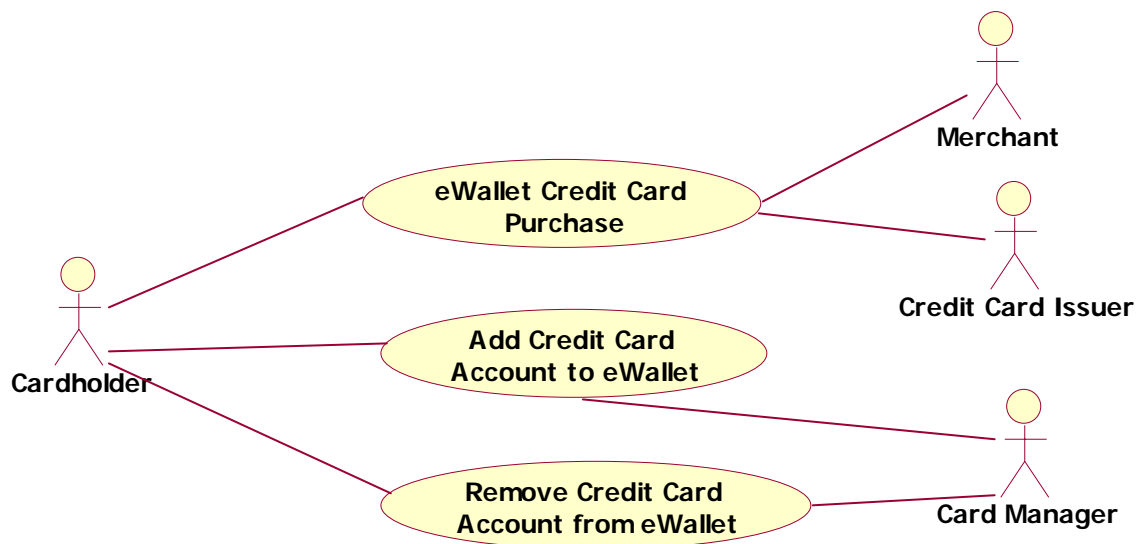
IBM's smart cards were developed to use and run Java and we are using IBM's smart cards. The only way to interact with these smart cards is to use the Java libraries provided. Using any other platform simply would not work.

# Chapter 4 Design

## 1. Introduction

This chapter contains a UML design that addresses the needs of our use cases discussed in chapter 2. It contains class diagrams and documentation for each class and component diagrams as well. Each class has a section that describes the major responsibilities of that class.

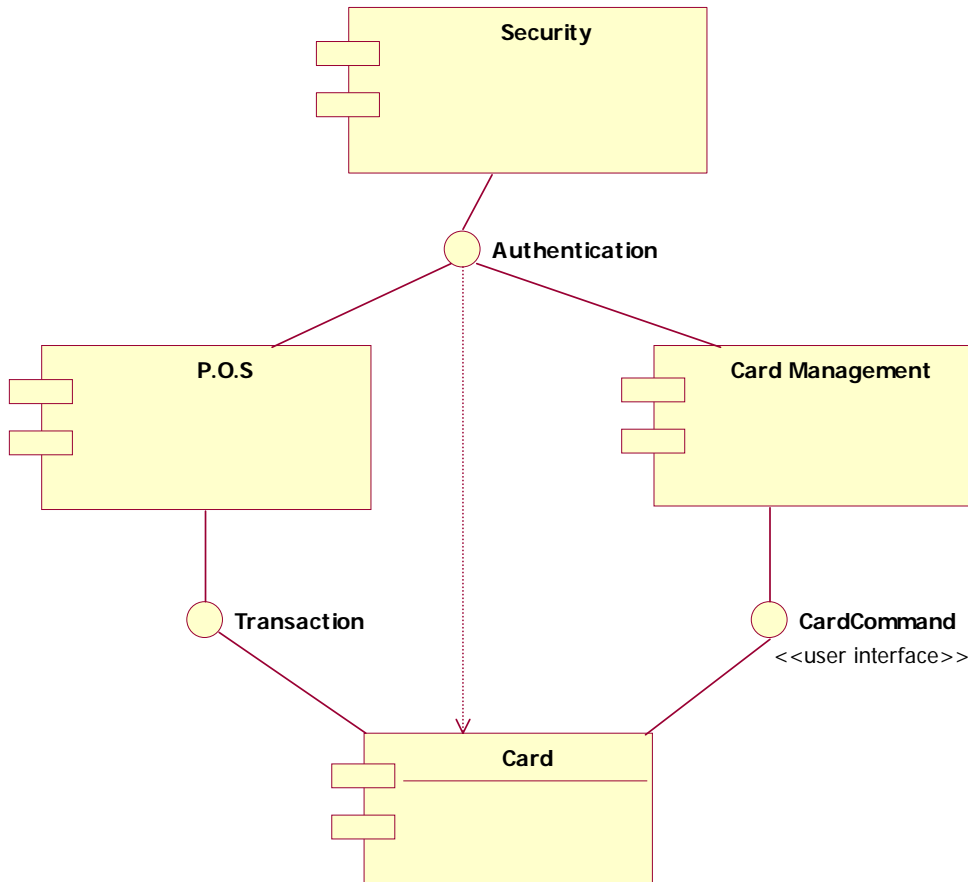
## 2. Subsystem Diagram



### 3. Subsystem Documentation

#### 1. Subsystem Overview

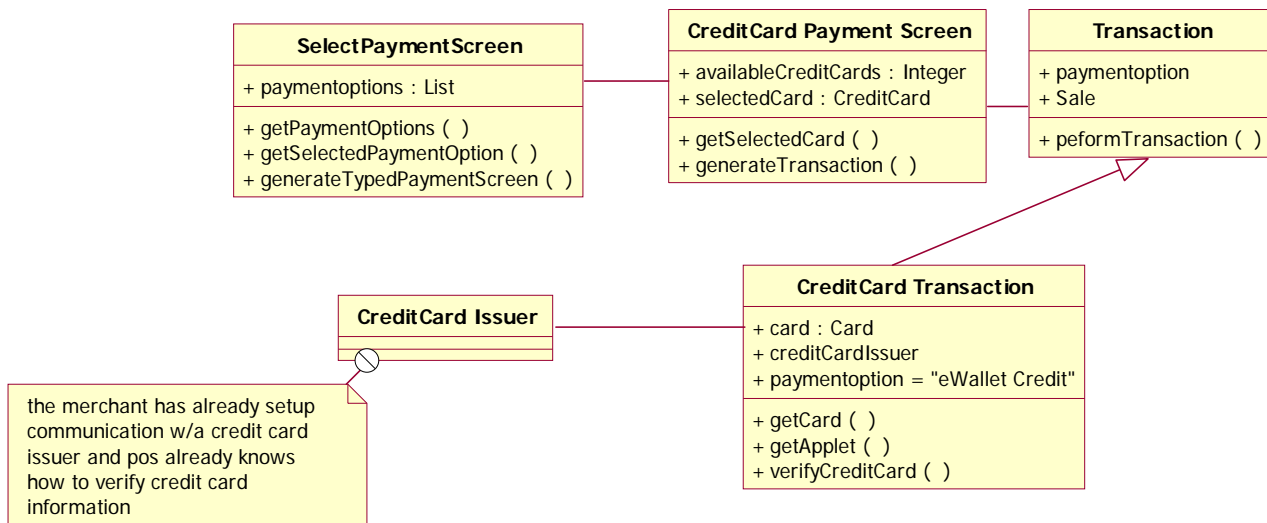
##### 1.1 Components Diagram



##### 1.2 P.O.S Component

The POS Component of the system is responsible for the merchant transaction piece of the process. The POS component interacts with the Card component to obtain necessary credit card information to complete a sale. It is essentially a set screens that drive the user and cashier through the checkout process. Along with retrieving necessary credit card account information from the card, the P.O.S component interacts with the Security component to make sure each access to the card is authenticated via a fingerprint captured from the user.

## 1.2.1 Component Class Diagram:



## 1.2.2 Class Documentation

### 1.2.2.1 Select Payment Screen

The main responsibility of the `SelectPaymentScreen` is to present the user with a list of payment options, such as cash, credit card, stored value, etc. A user arrives at the POS interface, inserts an eWallet, and is presented with a list that is produced from `getPaymentOptions`. The availability of credit card and stored value payment options is dependent upon whether or not these applets are installed on the card. The `SelectPayment` screen communicates with the Card to see which applets are installed. Once the user has made a payment selection the `SelectPayment` screen then generates the next appropriate screen to begin a transaction. (ie: when credit card is selected as the option, a credit card payment screen is instantiated)

### 1.2.2.2 Credit Card Payment Screen

The class invoked when the user chooses to make a credit card payment from their eWallet. It gets a list of available credit card accounts from the card and displays them for the user to choose from. Once the user selects the card to use this screen generates a transaction object for the sale to be completed.

### 1.2.2.3 Transaction

The `Transaction` class is responsible for actually performing the sale transaction, whether that be paying with a credit card, stored value, or cash. Any classes that are derived from `Transaction` need to implement `performTransaction()` where all the action happens.

### 1.2.2.4 Credit Card Transaction

This class is derived from the `Transaction` class. Along with knowing the sale total and tax it is also gets a credit card account to use from the `Credit Card Payment Screen`. This transaction object is responsible for contacting the credit card issuer to verify credit card account information and complete the sale, print a receipt, etc.

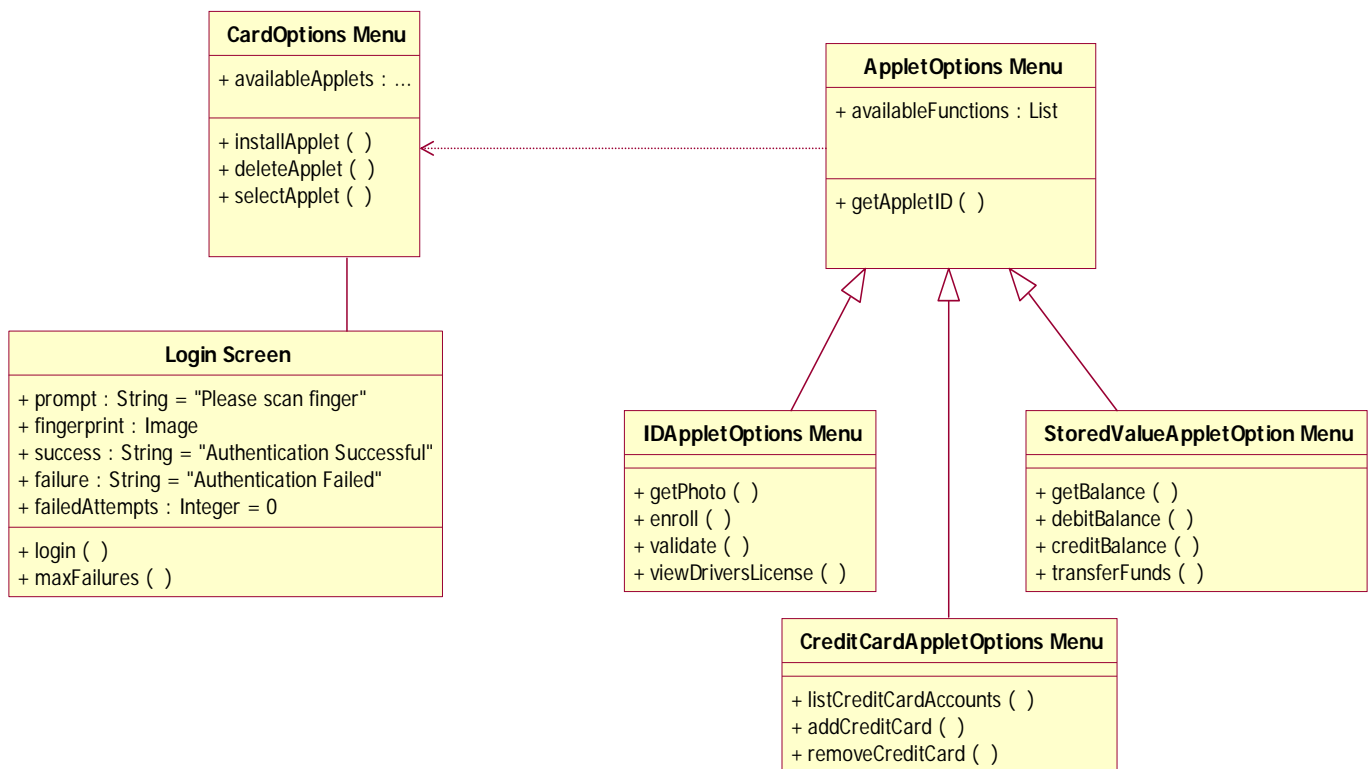
### 1.2.3 Summary

The POS component can behave as a stand alone application or plugin. If no credit card accounts exist on the eWallet or if no eWallet is used, the POS will behave as normal: accepting regular cash, check, etc. There is simply an added feature/option of accepting credit card accounts stored on the eWallet. The merchant is still responsible for verifying the credit card account information with the credit card issuer and completing the credit card sale as normal.

## 1.3 Card Manager Component

Card Manager Component: this component provides the user with an interface to modify applets installed on the eWallet smartcard. It is a menu driven GUI that requires the user to login before they can modify or view any data on the card. The Card Manager component interacts with the Card component – essentially providing a GUI wrapper to the card interfaces that is also secure.

### 1.3.1 Card Manager Classes Diagram:



## ***1.3.2 Class Documentation***

### **1.3.2.1 Login Screen**

The Login screen is responsible for authenticating the user before allowing any access to data stored on the card. It simply prompts for a fingerprint scan, and then hands that scan off to the Security component for verification. The Security component then notifies the Login screen as to whether or not authentication was successful. The Login screen keeps track of the number of invalid login attempts and after a certain number of unsuccessful attempts will disable login.

### **1.3.2.2 Card Option Menu**

The main responsibility of the Card Options menu is to provide an interface for managing applets on the card. The Card Options menu presents the user with options to install or delete applets as well as select an available applet to work with. This menu interacts with the Card component to obtain information about installed applets, as well as installing and removing them.

### **1.3.2.3 Applet Options Menu**

This is a generic class that provides information about an applet installed on the card. The main method it provides is `getAppletID`. Any subclass of this class needs this method, as the AppletID is the way to uniquely identify applets installed.

### **1.3.2.4 Credit Card Applet Options**

As a subclass of Applet Options menu this class allows the applet id to be queried but it also provides methods specific to dealing with credit cards. One of the responsibilities this class has is to interact with the Card component and provide the user with a list of available credit card accounts. It also facilitates the user's need to add credit card accounts to their eWallet, as well as remove them.

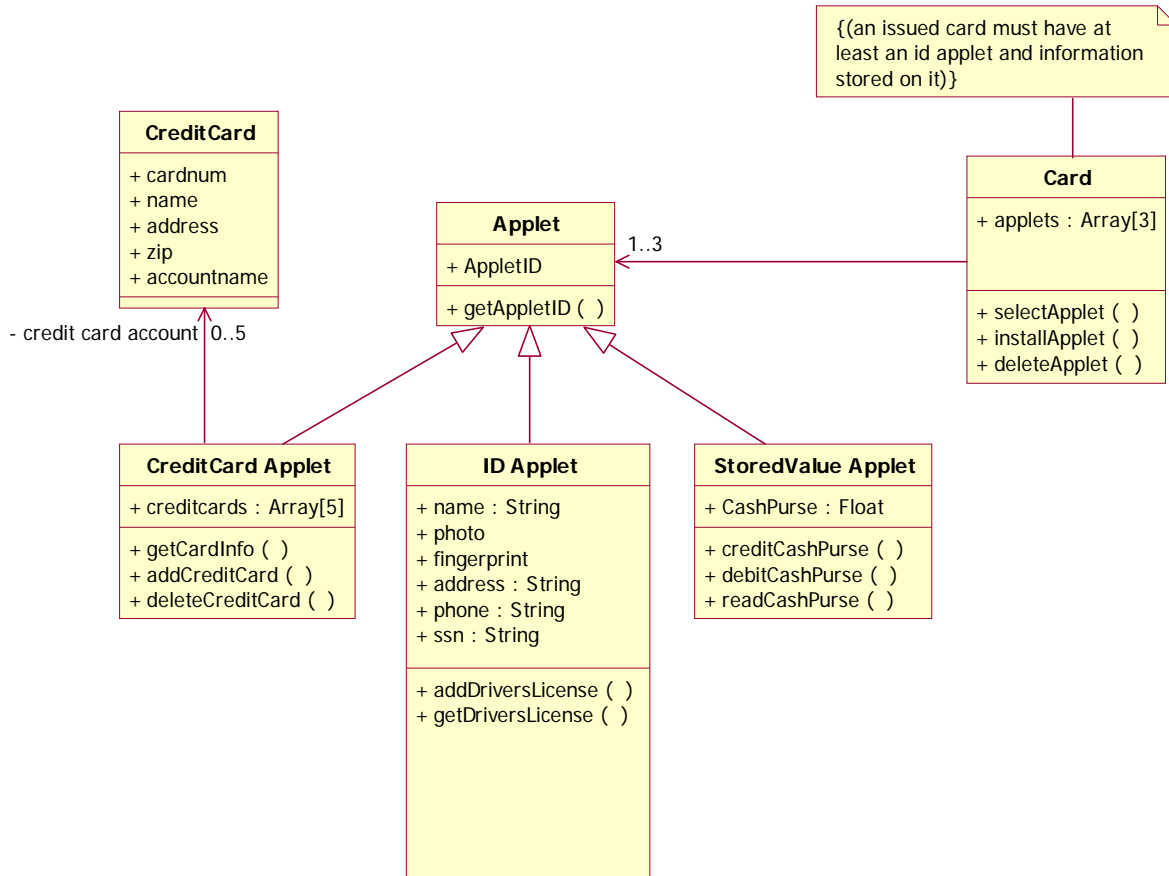
## ***1.3.3 Summary***

The Card Manager is the component the user uses to interact with their card. It provides a simple GUI interface for the user to add/delete or otherwise modify data on their card. It is secure in the fact that it requires fingerprint authentication before allowing any access. Most of the classes are screens driven through by the user and menu options are determined by which applet is being accessed, or what kind of applet maintenance is being performed.

## **1.4 Card Component**

The Card component is the component that any other system component uses to actually access data on the card. It is comprised of objects representing the card system itself, the applets stored in it, and the information those applets can contain.

### 1.4.1 Card Classes Diagram:



### 1.4.2 Class Documentation

#### 1.4.2.1 Card:

The Card class represents the eWallet card as an applet container. It provides methods to install and delete applets (such as installing the credit card applet) as well as selecting a certain applet that needs to be active so data can be read or modified. The Card class provides an interface to the card file system and high-level applet management, and currently the card can have up to 3 applets installed (ID, Credit Card, and Stored Value). A constraint is imposed in that a card that is officially issued must at least have some id information installed on it => a minimal ID applet installation.

#### 1.4.2.2 Applet:

This is a general class to describe all applets. While different applets have different functions, each applet has a unique AppletID by which it is distinguished among other applets that may reside on the card. The AppletID is used when selecting an

applet for use or deleting applets off the card. Therefore all classes derived from Applet must have a unique AppletID and be able to retrieve this applet id when needed.

#### 1.4.2.3 Credit Card Applet:

The Credit Card Applet is the Applet subclass used to manage credit card accounts stored on eWallet. The credit card accounts themselves are represented by Credit Card objects, and the Credit Card Applet simply stores a List of these Credit Cards objects. The Credit Card Applet object interacts with a POS or Card Manager to provide those components with information about the card applet. The main responsibility of this class is to act as a container for CreditCard objects and provide an interface to retrieve information about those objects as well as add and remove them. For now the maximum number of credit card accounts the applet can have is 6.

#### 1.4.2.4 Credit Card:

This class is representative of a credit card account. It is what is stored in a list in the credit card applet. It provides a means of storing and accessing credit card account data (name, card number, card type, exp date, etc)

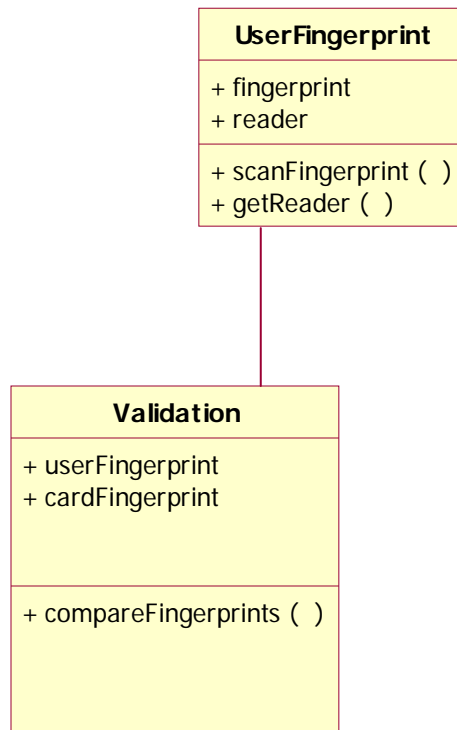
### ***1.4.3 Summary***

The card component is the basis for eWallet. It represents the eWallet itself and the applets and information stored on it. This component will interact with most every other component in the system, especially the security component because of requirements for authentication. Along with generic a generic applet class and card class it also contains classes that represent specific attributes of the applets that manager credit cards, stored value, etc.

## **1.5 Security Component**

The security component is responsible for any and all authentication against the smart card. Any time data access is attempted, the user of the card must authenticate his or herself against the fingerprint scan stored on the card. For this reason the security component interacts with both the POS and Card Manager components because the functionality of these components requires access to information stored on the card. The security component depends upon the Card Component because no authentication can take place without access to the fingerprint image that is stored on the card and used for comparison.

### 1.5.1 Security Classes Diagram:



### 1.5.2 Class Documentation

#### 1.5.2.1 User Fingerprint:

This class is responsible for managing the user fingerprint to use as authentication against a fingerprint stored on a card. Along with just a fingerprint image, this class also contains a reader object so that the class can obtain a fingerprint by scanning from a fingerprint reader. This class is then used to validate against another print.

#### 1.5.2.2 Validation

The Validation class does the actual comparison between two fingerprints. The main method `compareFingerprints` method compares the prints of the two fingerprint members and then returns a Boolean value representing whether or not the comparison was successful.

#### 1.5.2 Summary

While simple and small, it seems necessary to separate out the security from the rest of the components because it is more of a stand alone piece of implementation that interacts with almost every other component in the system. Any kind of access to the card, whether it be for applet management or for manipulating / reading data stored in the actual applets, must be authenticated. By separating this out it makes the model of the system clearer as well as the role of security in the greater scheme of the system.

## Summary

Next semester we plan to implement our eWallet project and hopefully someone will be so impressed that we will all get good jobs making big bucks. Smart cards can prove to be a tough sell because of the cost of change to infrastructure and the idea of new unfamiliar technology. While this project will begin as our Senior Project we also hope that we can familiarize people with the technology and end up with some business opportunities as well as a flashy project. We definitely feel we are much better off to actually start the senior project. Undoubtedly this level of design would've never happened had we not taken the Design class. It allowed us to make a lot of important design and even implementation decisions early enough in the project that it should save us a lot of time and rework when we finally being implementation. The business proposal assignment was good because you can't make big bucks if you can't propose something to a business. The use case assignment was very helpful. It's nice to bring programmers back up to the level of a user. Walking step-by-step as a user should be where the project begins. We didn't see how it was relevant that we have an assignment to discuss what platform we were using. We also think design is good but are a little unsure how useful it's going to be when it comes time to write code. We are not so sure we did the best job and still need practice in the area.

### **Glossary**

- POS (point of sale): a business or place where a product or service can be purchased
- Smart Card: a plastic card containing a microprocessor that enables the holder to perform operations requiring data that is stored in the microprocessor; typically used to perform financial transactions.
- Stored Value: electronic cash stored on the smart card, having been transferred from the cardholder's bank account directly to the card. Stored Value can be redeemed by Merchants who accept it at their stores.