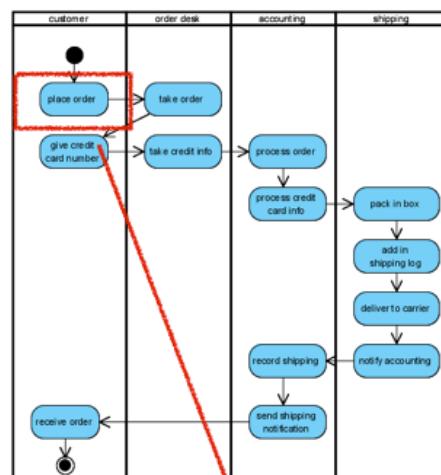
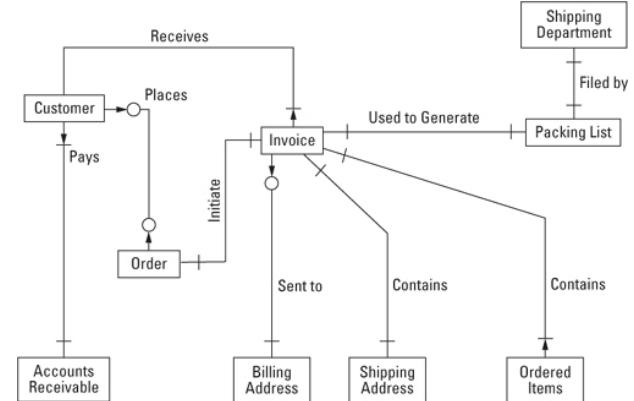
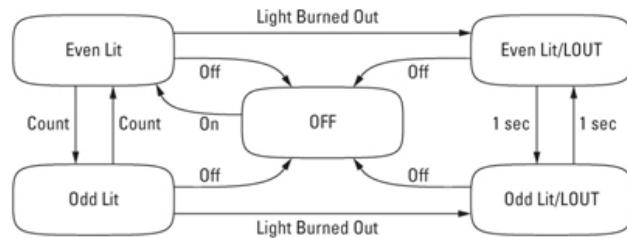
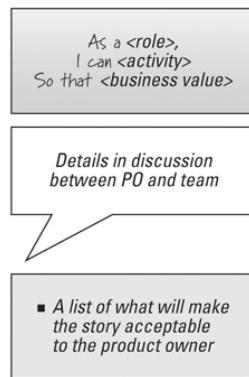


CMPE 202

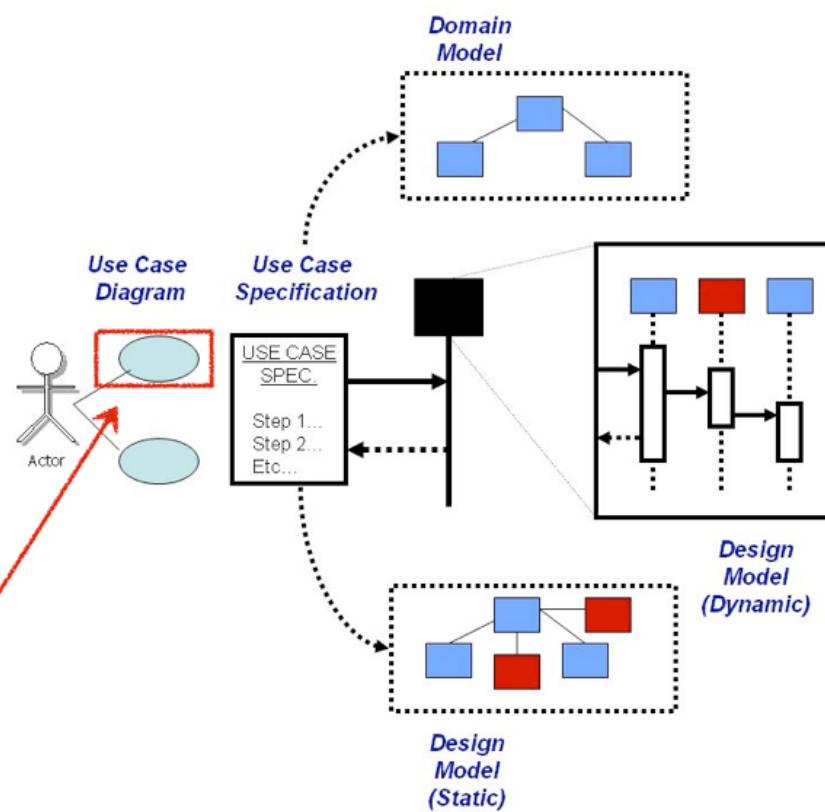
The Starbucks Mobile App Simulator

Starbucks

Modeling Workshop Examples

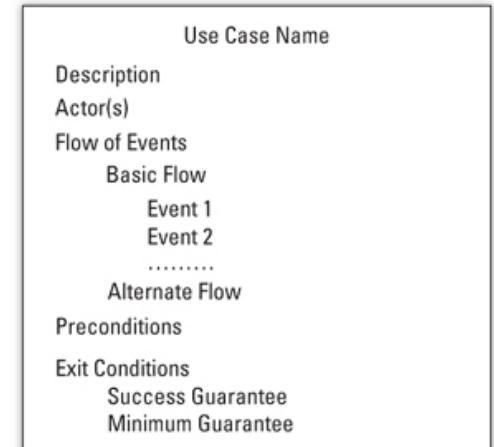


Activity --> Use Case

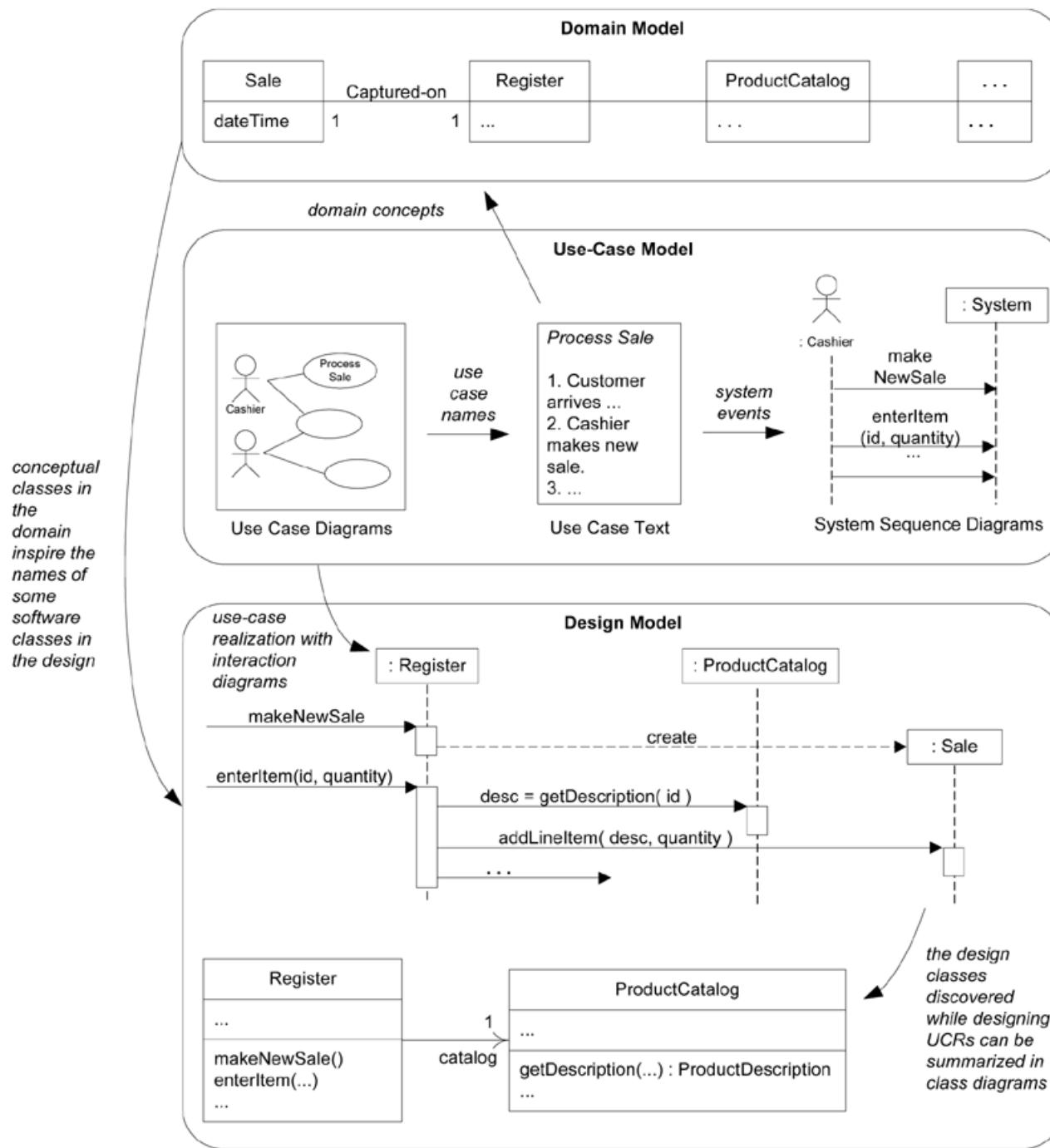


Use Case Scenario Realized
as Domain & Design Models

Set $\text{SUM}(x)=0$
FOR each customer X
IF customer purchased paid support
AND ((Current month) \geq (2 months after ship date))
AND ((Current month) \leq (14 months after ship date))
THEN $\text{Sum}(X)=\text{Sum}(X) + (\text{amount customer paid})/12$



Sample Unified Process Artifact Relationships





<http://www.starbucks.com/coffeehouse/mobile-apps>

Mobile Applications

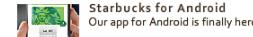
[Tweet](#) [+1](#) [Like](#) 1K



STARBUCKS CARD MOBILE APP
It's the fastest way to pay



[email](#) [share](#) [get code](#) [get transcript](#)



Starbucks for Android
Our app for Android is finally here.



Starbucks Card Mobile
It's the fastest way to pay.



Mobile Applications
Two iPhone apps from Starbucks

Starbucks Coffee

Create a little extra time in your day,

with the Starbucks card mobile app.

It's the fastest way to pay for your drink.

View and reload your Starbucks card balance,

and even check your My Starbucks Rewards Stars.

So now you'll have more time to relax.

Get to work,
walk the dog,

or just enjoy your coffee even more.

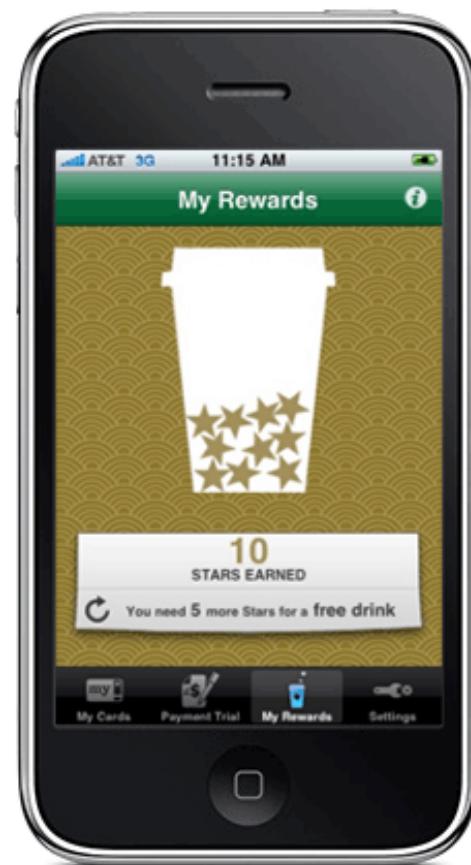
Just scan
and go.

Starbucks card mobile app,
it's the fastest way to pay.

Starbucks Coffee
Get it for iPhone or Blackberry.

Write a User Story

- For the iPhone App:
Starbucks Card Mobile



User Story - First Parse Statements

- As a Customer

- with the Starbucks card mobile app.

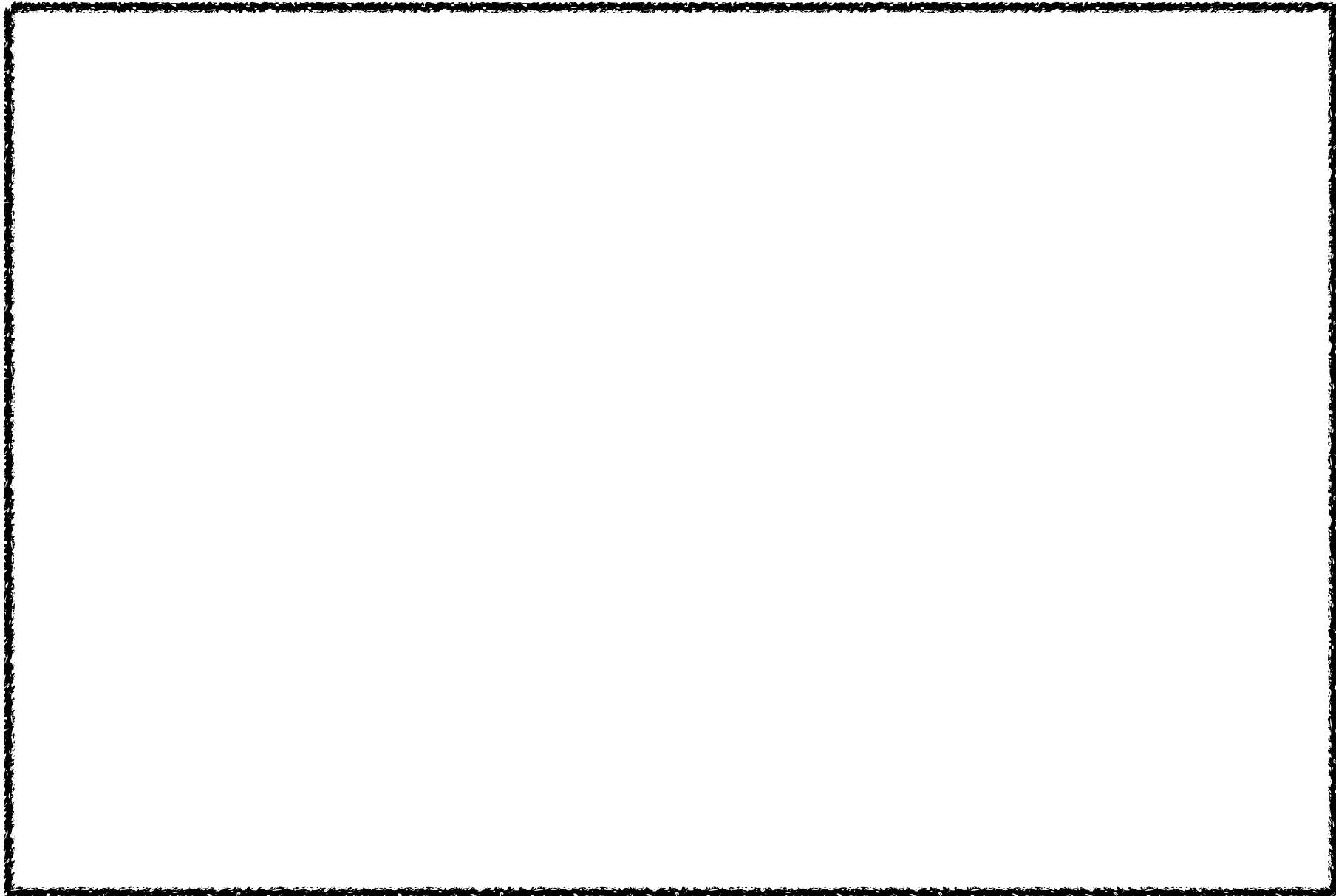
- I can ...

- View and reload your Starbucks card balance,
 - and even check your My Starbucks Rewards Stars.
 - Just scan and go.

- So that...

- Create a little extra time in your day
 - It's the fastest way to pay for your drink.
 - So now you'll have more time to relax.
 - Get to work, walk the dog,

User Story

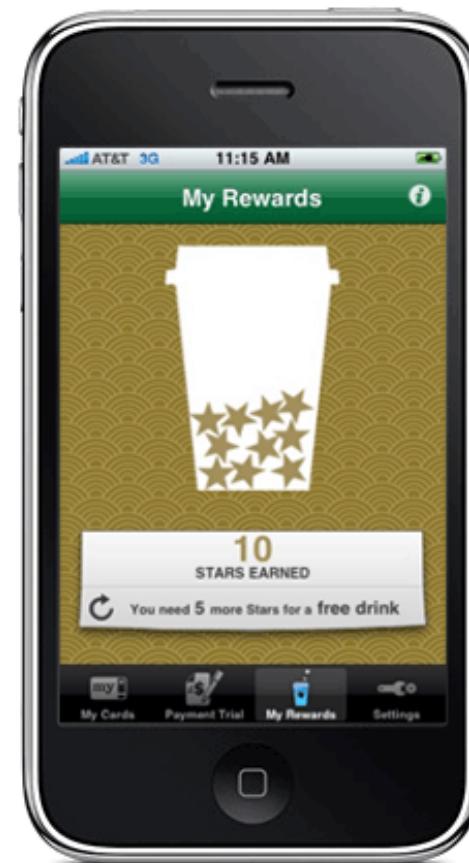


User Story

As a customer with the Starbucks card mobile app,
I can view, reload, check my rewards and pay on the go,
so that I can have a little extra time in my day to relax,
get to work faster and walk my dog.

Draw a UI Wireframe

- For the iPhone App's Pin Authentication and Payment Screen.



UI wireframe



UI wireframe



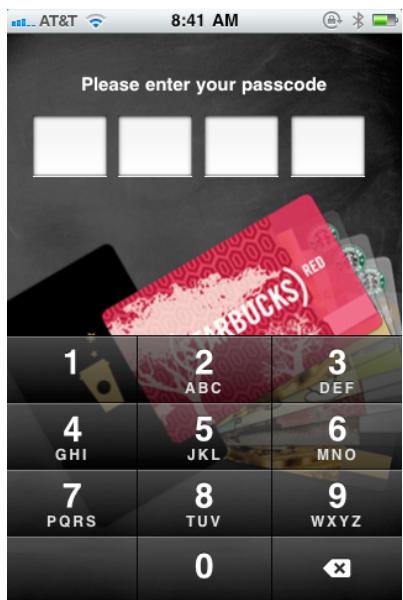
Pin Screen



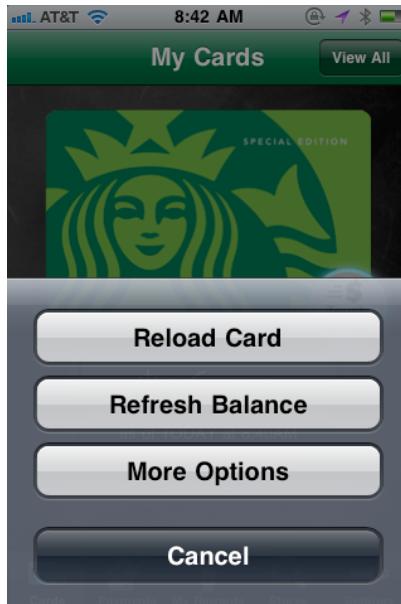
Main Screen



Payment Screen



Pin Screen



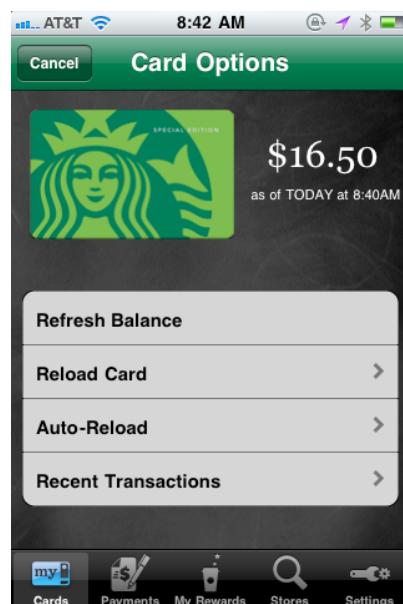
My Cards - Options



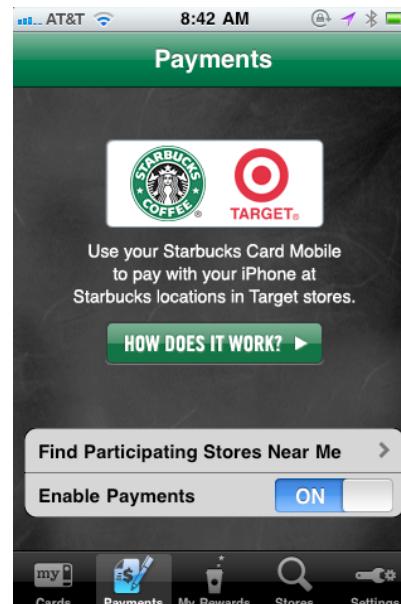
My Cards - Main



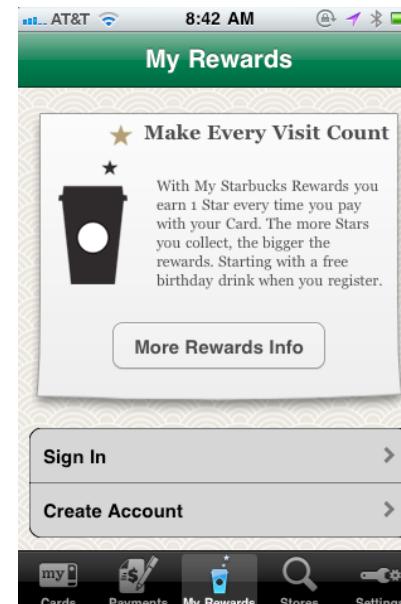
My Cards - Pay



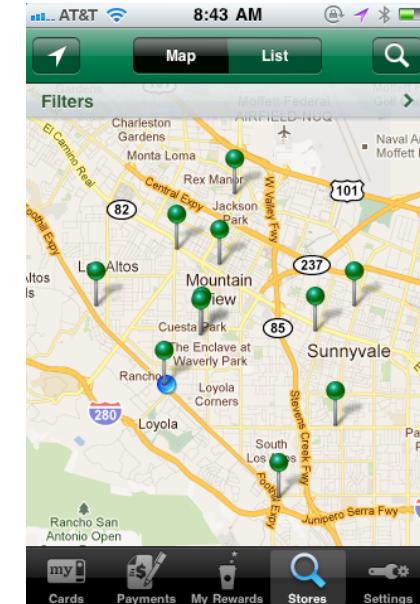
My Cards
More Options



Payment Setup



Rewards Setup

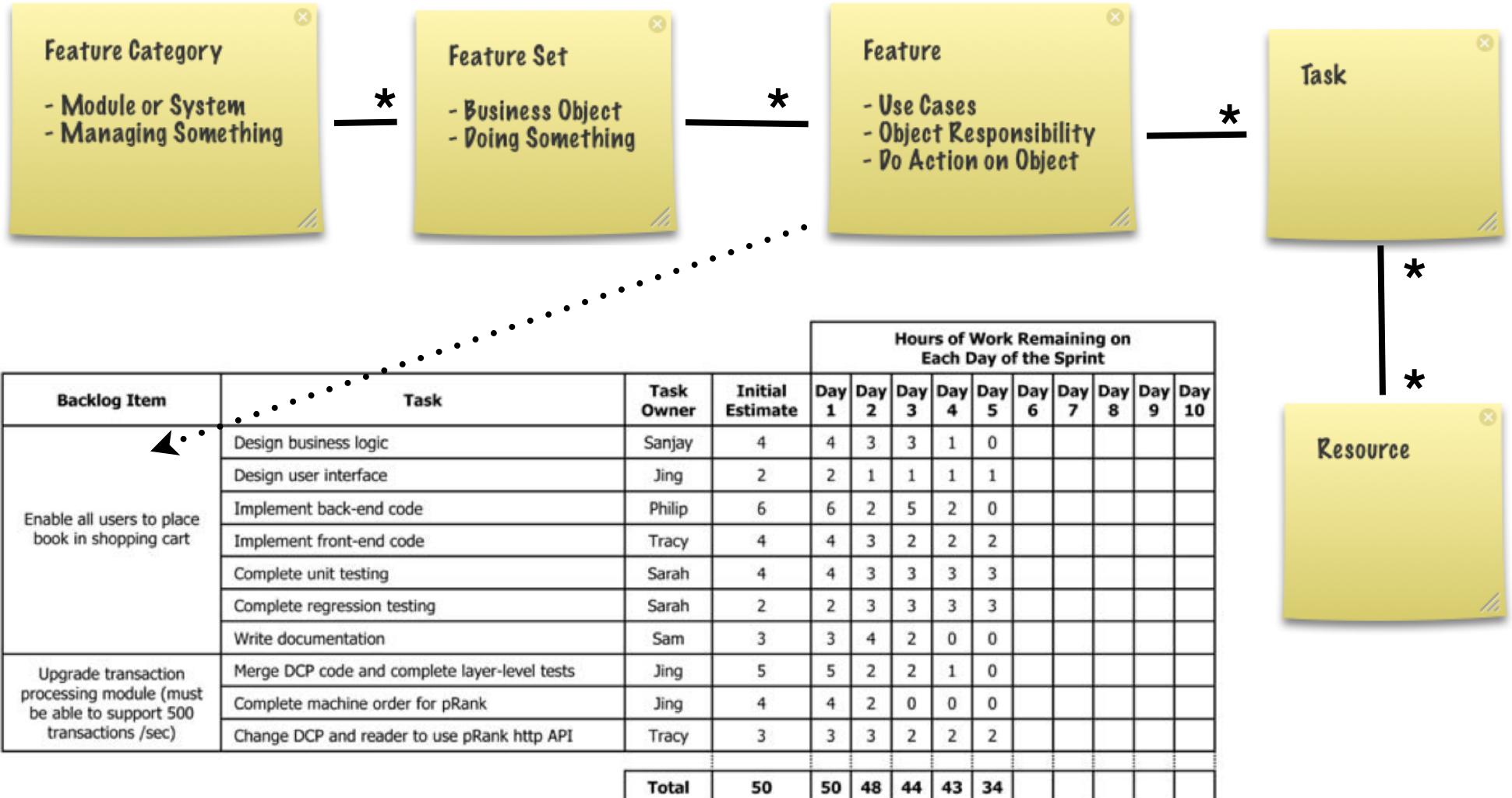


Find Starbucks

Product Backlog

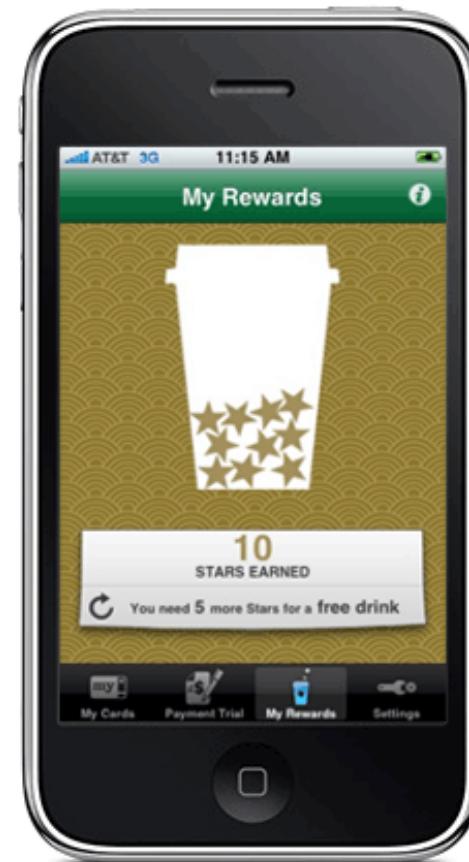
Backlog Item	Importance
Authenticate with a personal pin	100
View Current Balance on Primary Card	100
Pay with Primary Card	100
Check Balance on Card	90
Add Additional Cards	90
Reload Card	90
Enable/Disable Payments	90
Find Starbucks Store	50
Setup Rewards	80
View Recent Transactions	50

Feature Models (as backlog items)

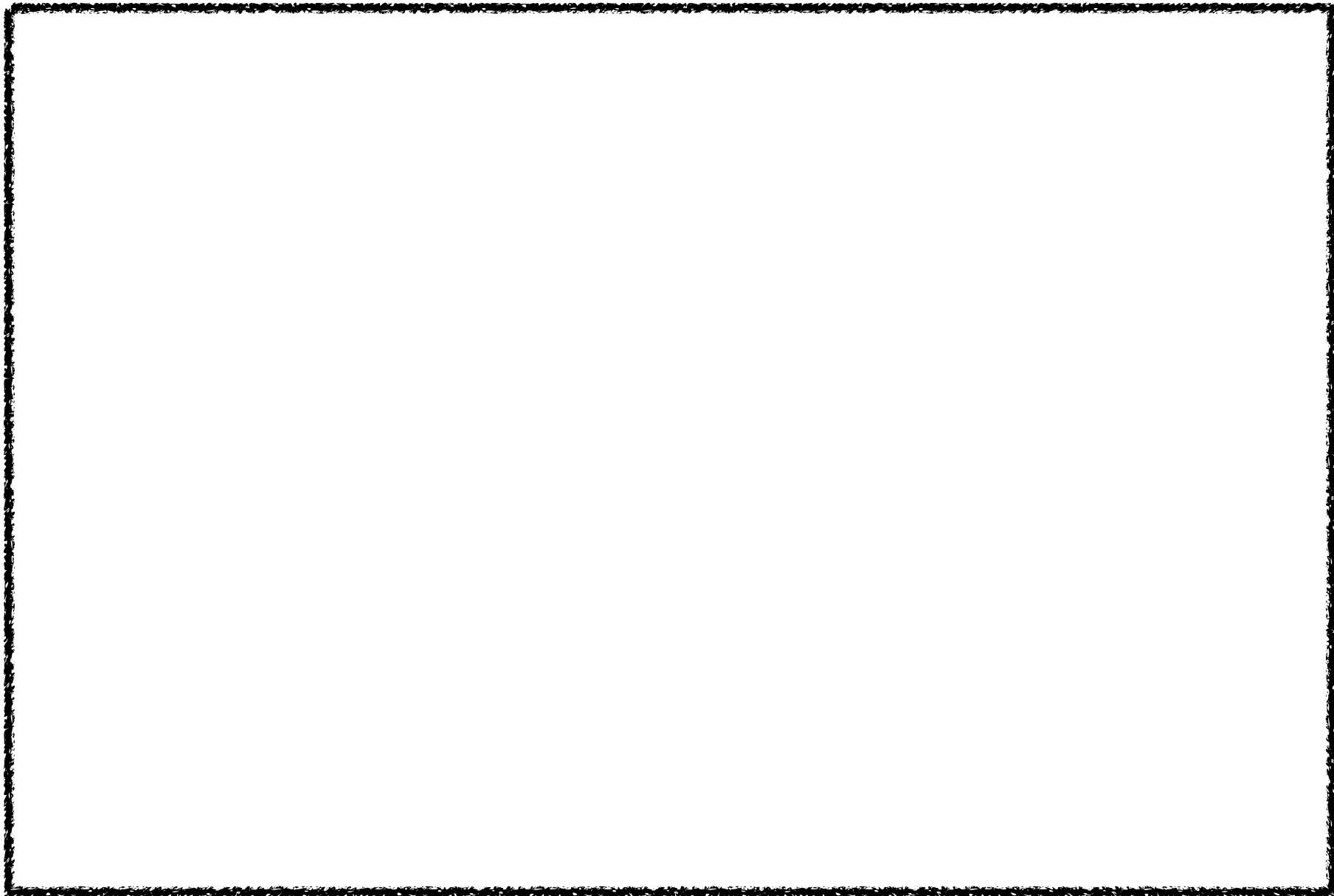


Create a Use Case Diagram

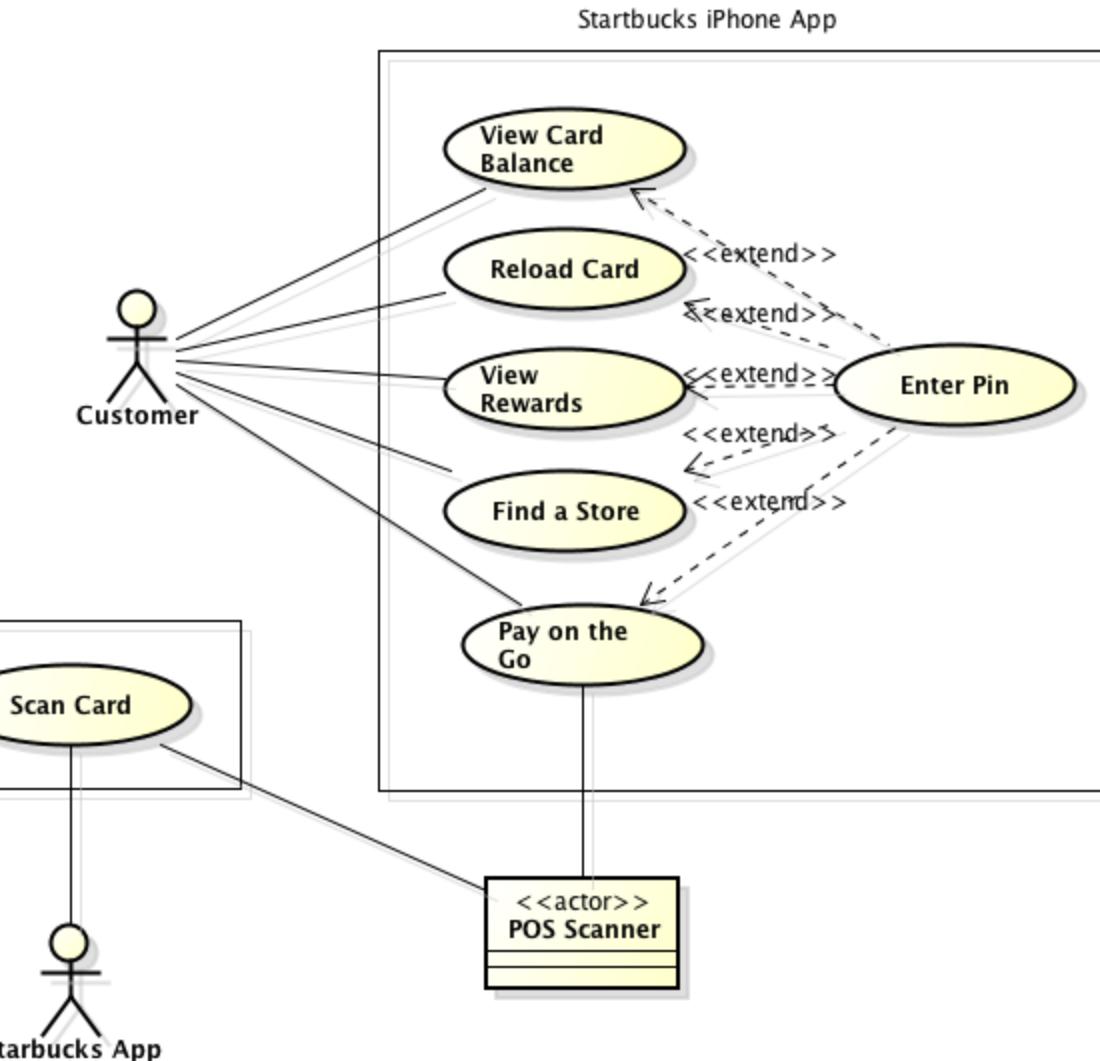
- For the iPhone App:
Starbucks Card Mobile



Use Case Diagram



Use Case Diagram

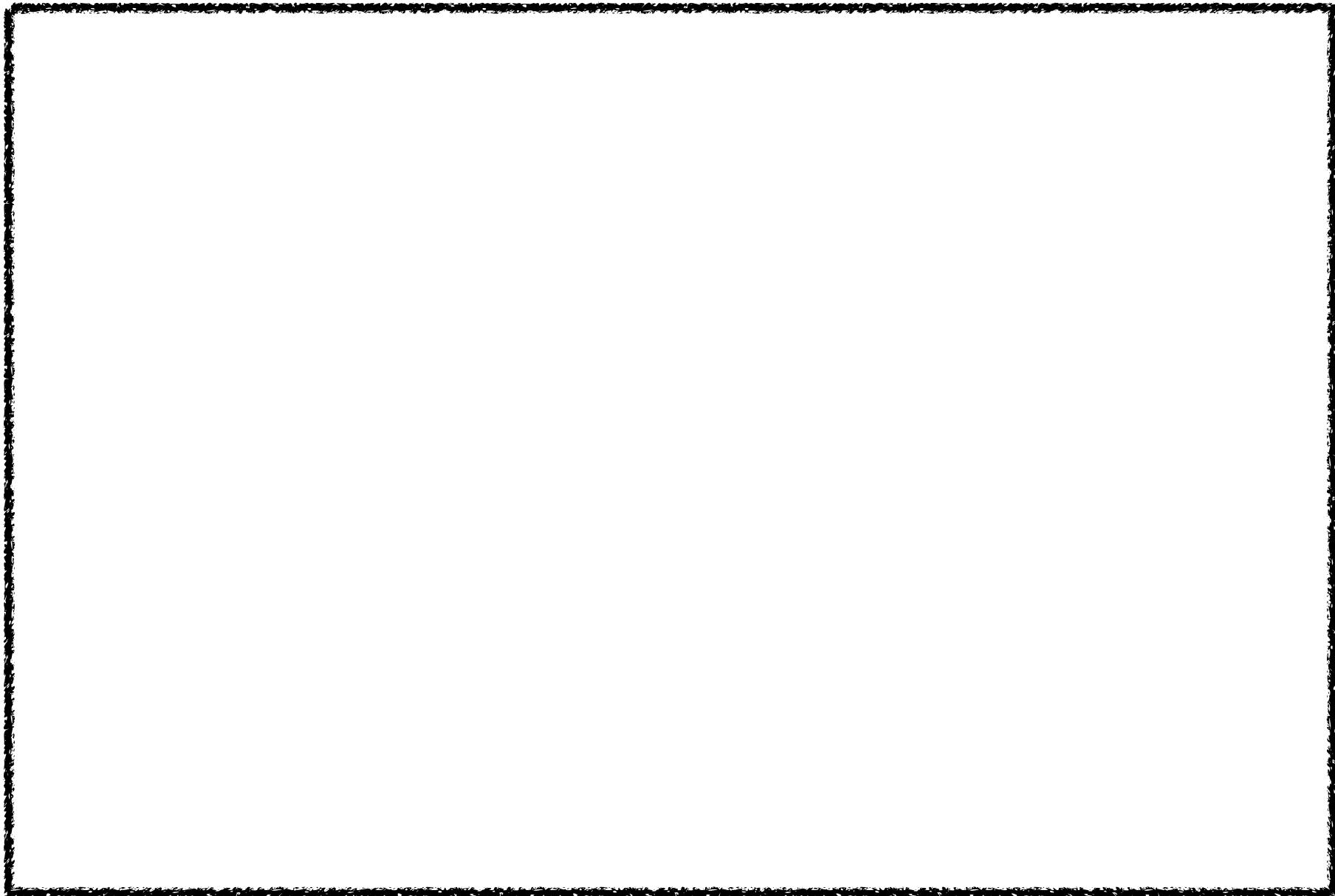


Model a UML Activity Diagram

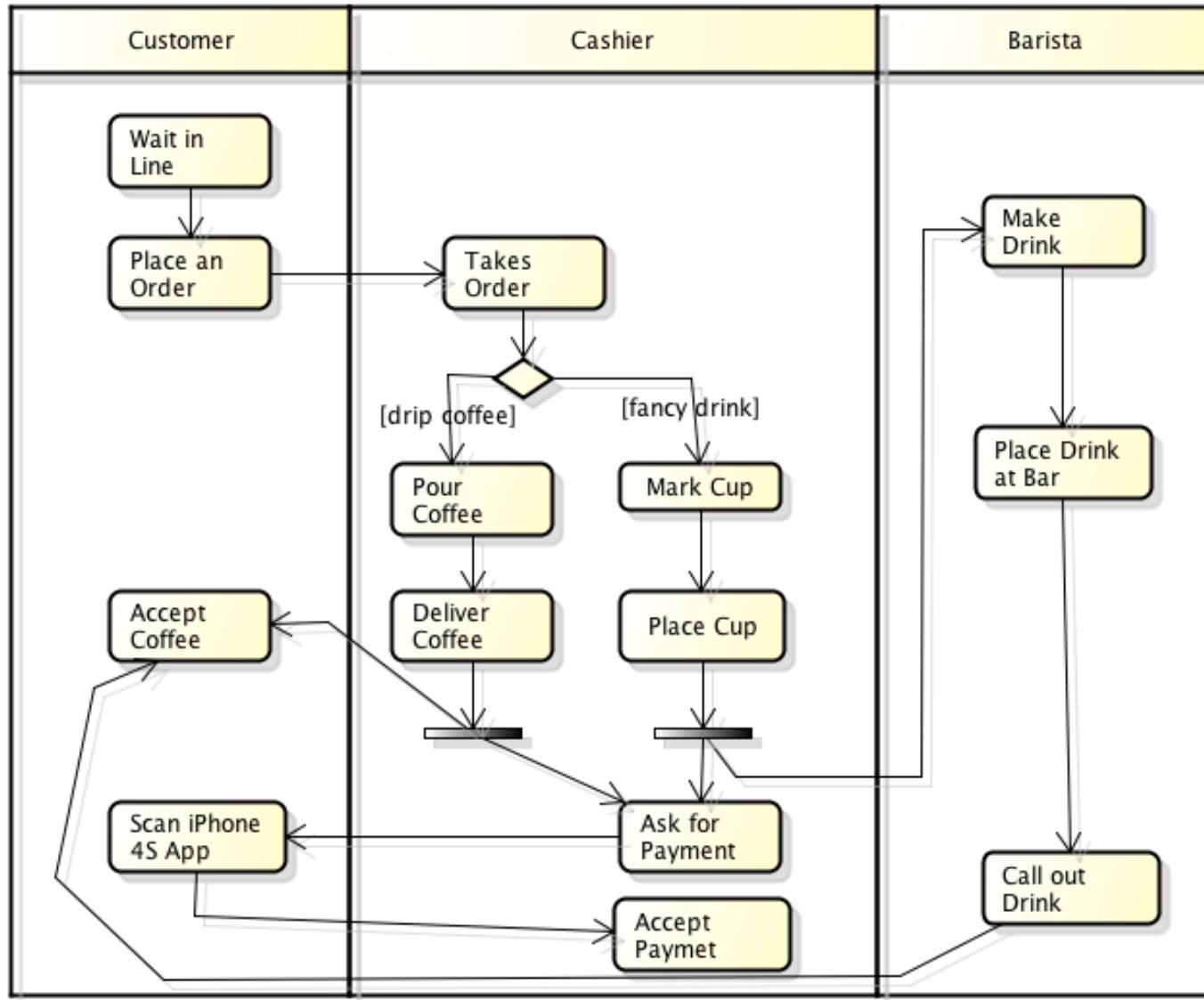
- For the Starbucks “Place Order” to “Got Coffee” Process



UML Activity Diagram

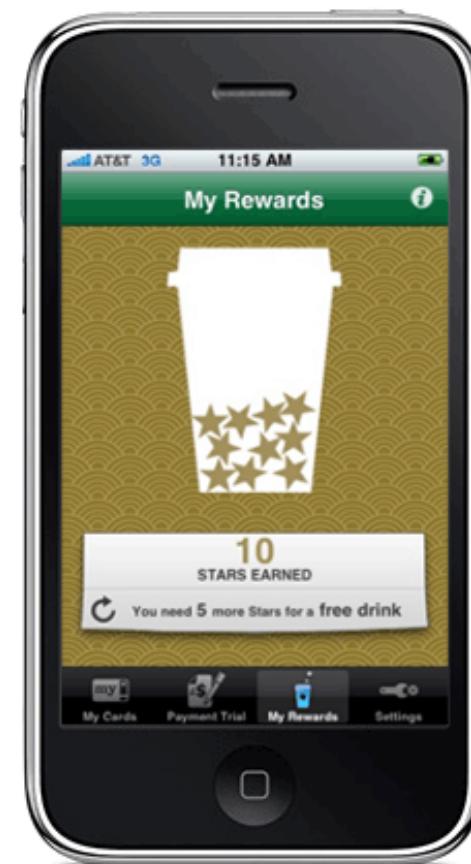


UML Activity Diagram



Draft “one” Use Case

- For the iPhone App:
Starbucks Card Mobile
- Note: One Use Case Specification + System Sequence Diagram



Use Case Name: << use case name >>

Brief Description:

<< description... >>

Actors: << list of actors >>

Basic Flow:

1. << step 1. >>
2. << step 2. >>
3. << step 3. >>
4. << step 4. >>

Alternate Flow:

4a. <<alternative to step 4. >>

Preconditions: << list preconditions >>

Success Guarantee: << list here... >>

Minimal Guarantee: << list here... >>

Use Case Name: Pay on the Go

Brief Description:

Customer wishes to use the Starbucks App to pay for a drink.

Actors: Customer

Basic Flow:

1. Customer Starts-Up the Apps and the App challenges the Customer to enter a Pin.
2. Customer enters the Pin which the App validates successfully and then shows the Customer the last card used, the current balance on the card and the option to pay with the card.
3. The Customer selects the option to pay and the App displays a bar-code for the POS scanner.
4. The Customer presents the bar-code to the scanner and completes the payment transaction.
5. The Customer then checks the remaining balance and closes the App.

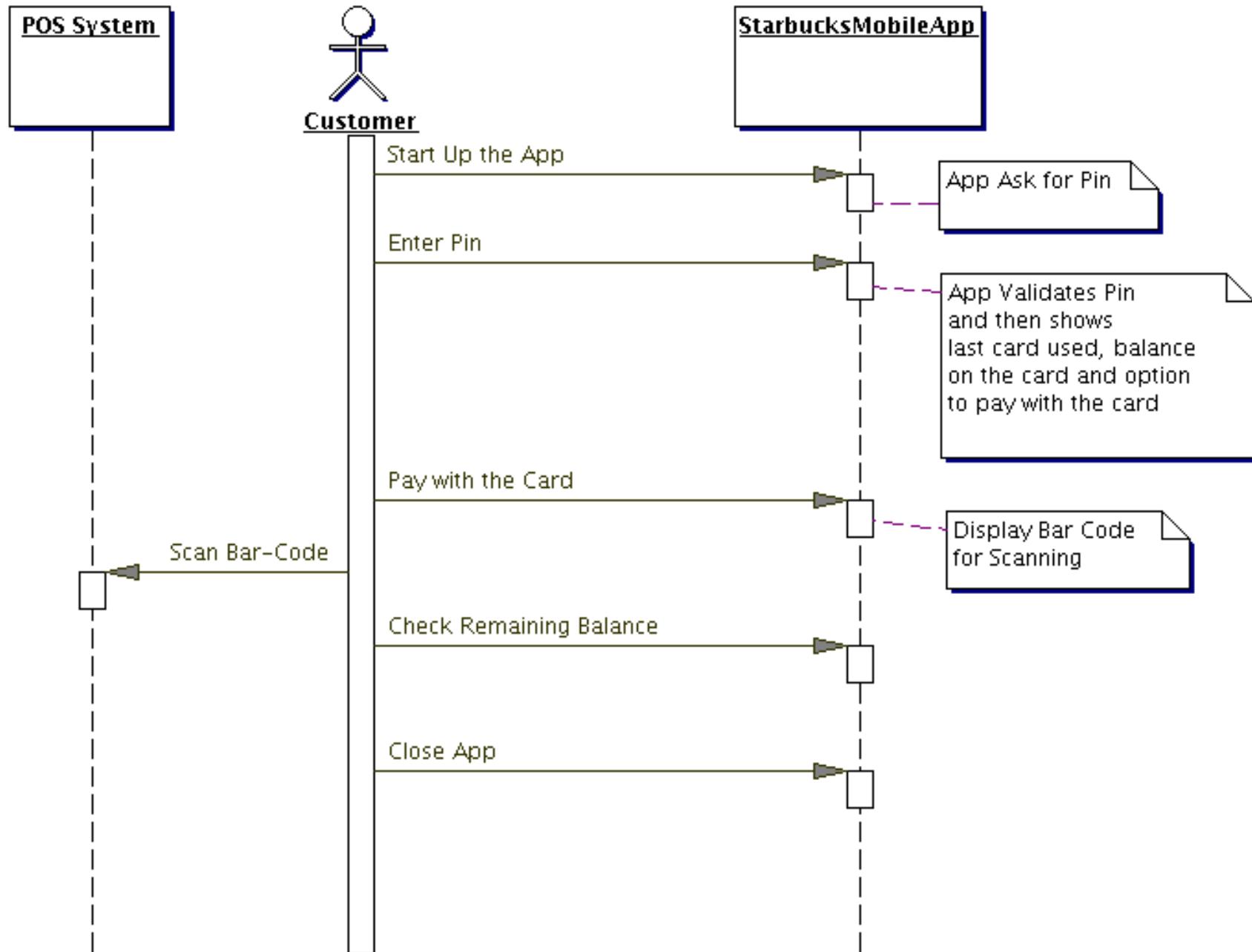
Alternate Flow:

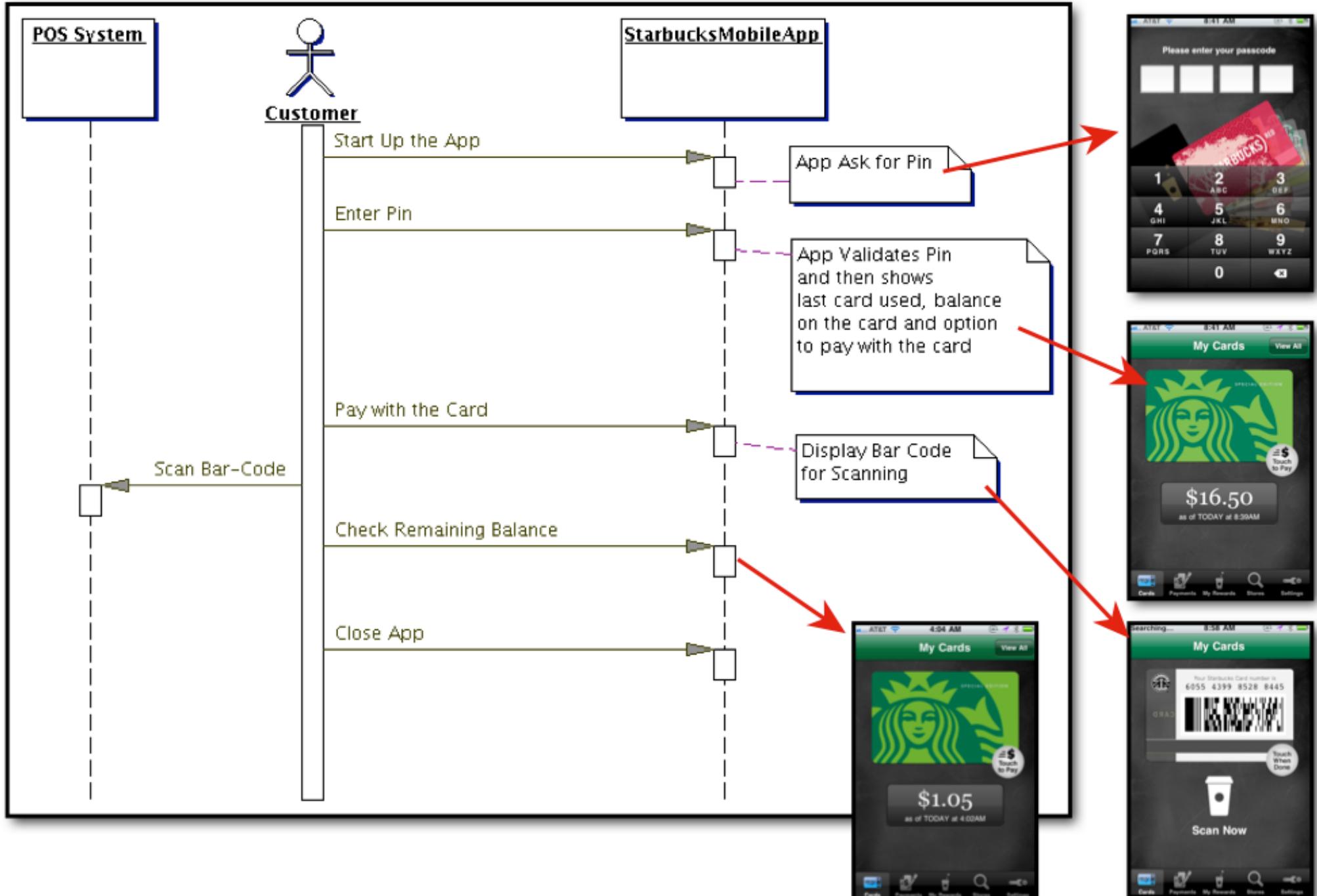
- 2a. The Customer realizes that there is not enough credit on the card and decides to chose another card or reload the current card.

Preconditions: Customer as a Card setup with the App and a Pin configured.

Success Guarantee: An electronic receipt for the transaction is created and made available to view on the App.

Minimal Guarantee: If the transaction failed, no credits on the card will be deducted and the balance on the Card will remain the same.

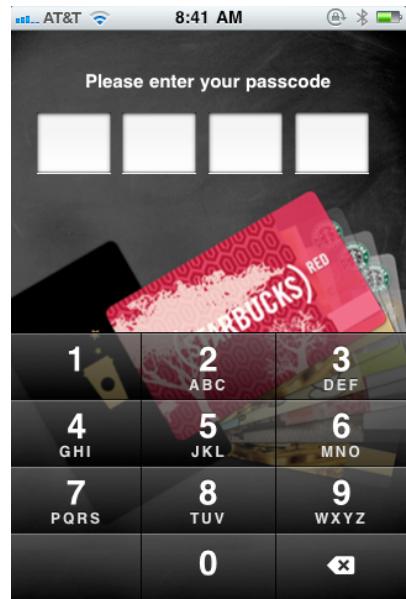




Starbucks

Sample Use Case Realization (Partial)

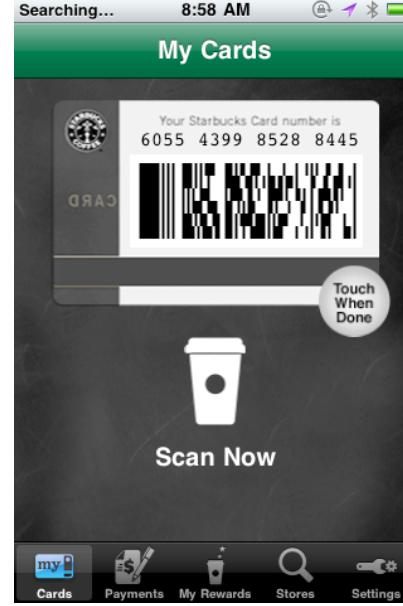
“Pay on the Go” Use Case



Pin Screen



My Cards - Main



My Cards - Pay

1. Pin Screen
2. Main Screen
3. Pay Screen

Use Case Name: Pay on the Go

Brief Description:

Customer wishes to use the Starbucks App to pay for a drink.

Actors: Customer

Basic Flow:

1. Customer Starts-Up the Apps and the App challenges the Customer to enter a Pin.
2. Customer enters the Pin which the App validates successfully and then shows the Customer the last card used, the current balance on the card and the option to pay with the card.
3. The Customer selects the option to pay and the App displays a bar-code for the POS scanner.
4. The Customer presents the bar-code to the scanner and completes the payment transaction.
5. The Customer then checks the remaining balance and closes the App.

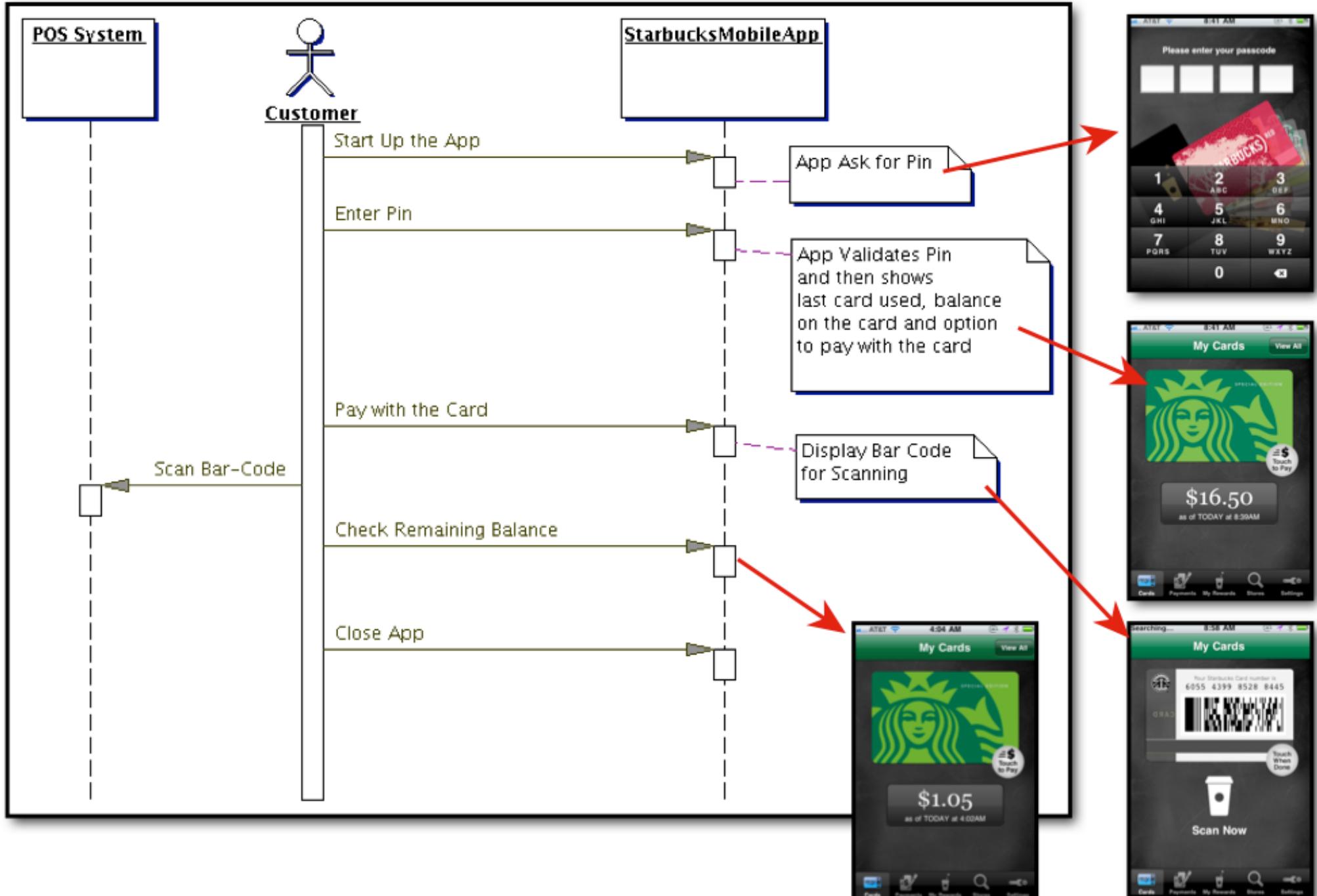
Alternate Flow:

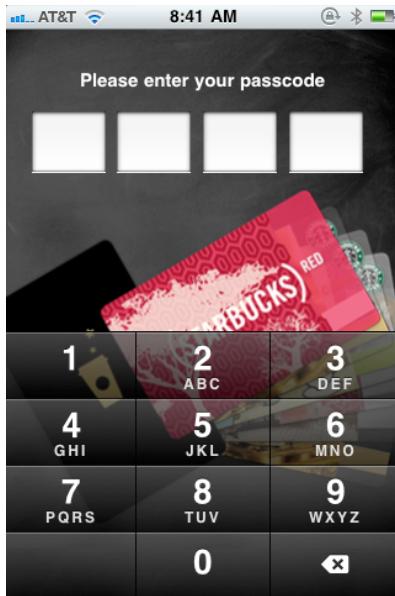
- 2a. The Customer realizes that there is not enough credit on the card and decides to chose another card or reload the current card.

Preconditions: Customer as a Card setup with the App and a Pin configured.

Success Guarantee: An electronic receipt for the transaction is created and made available to view on the App.

Minimal Guarantee: If the transaction failed, no credits on the card will be deducted and the balance on the Card will remain the same.

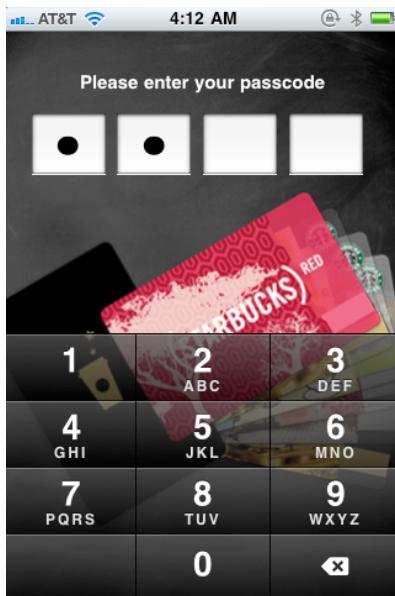




Pin Screen

For the “Pin Screen”, design a composition with:

Key Pad
Passcode Display

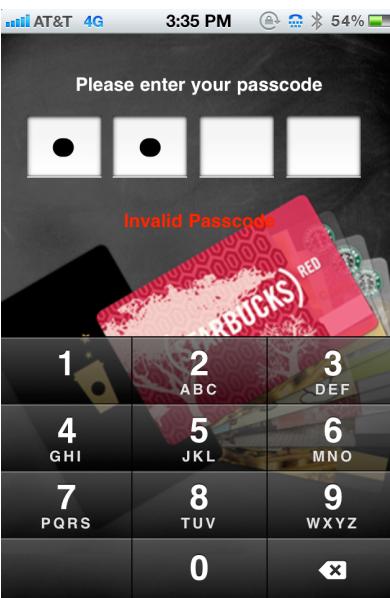


Passcode Display should echo how many pin values were entered. If the “backspace” key was pressed, one pin value is removed.



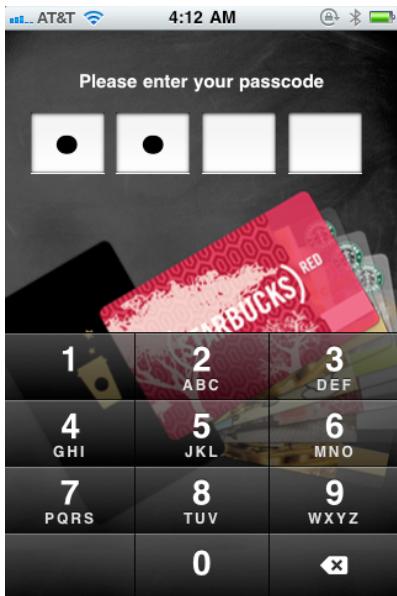
Pin Screen
(with error message)

If an invalid pin was entered, the pin authentication starts again.



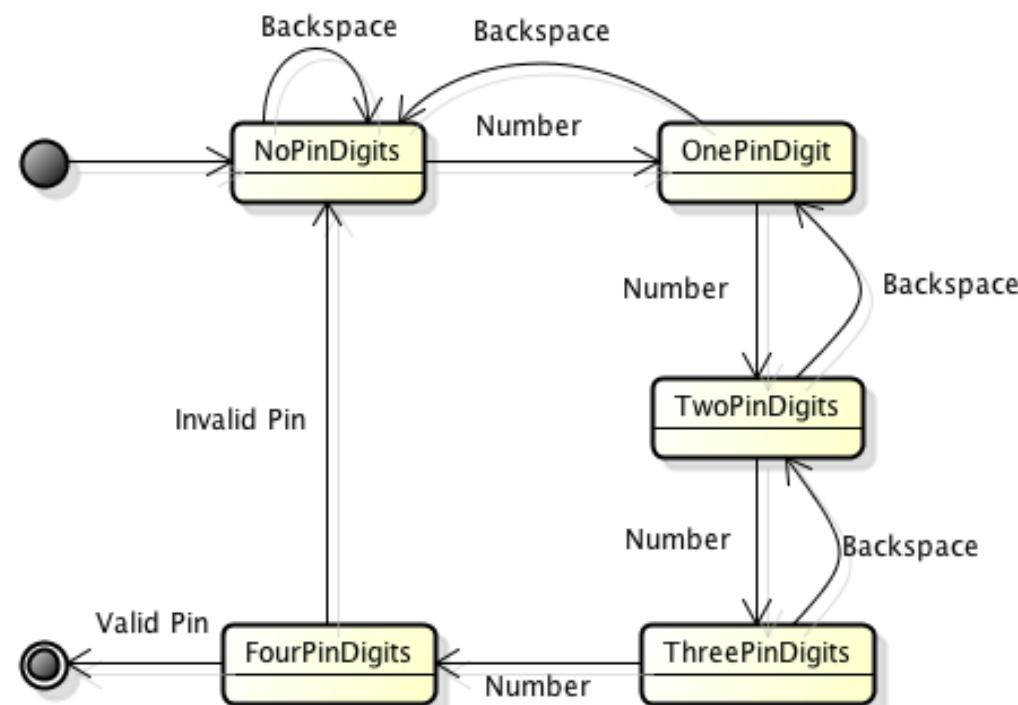
User has to enter another pin
(i.e. try again)

Pin Screen

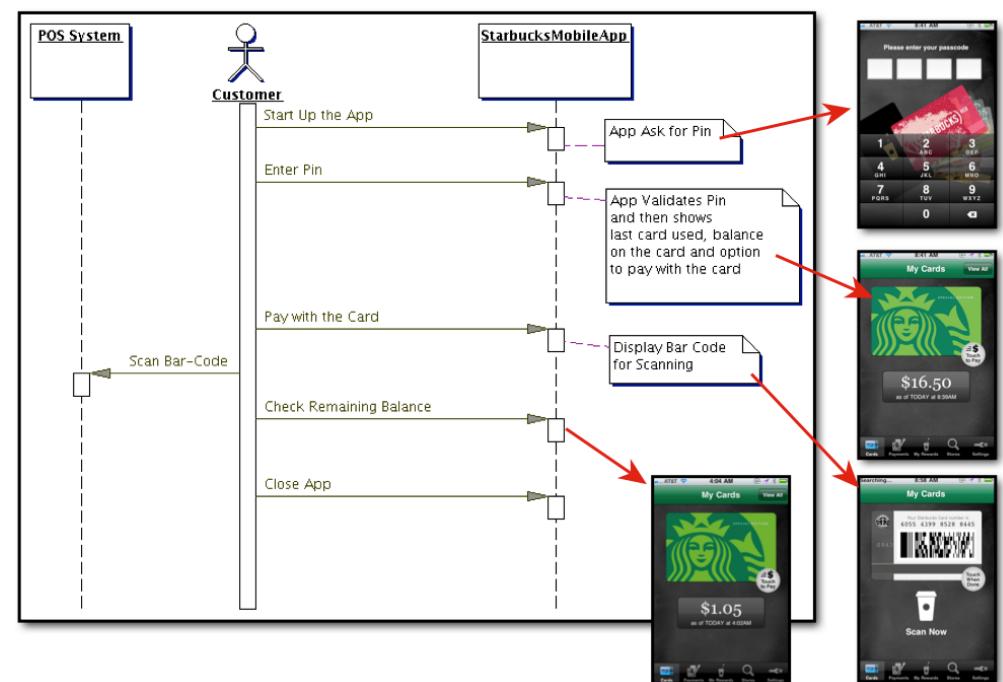
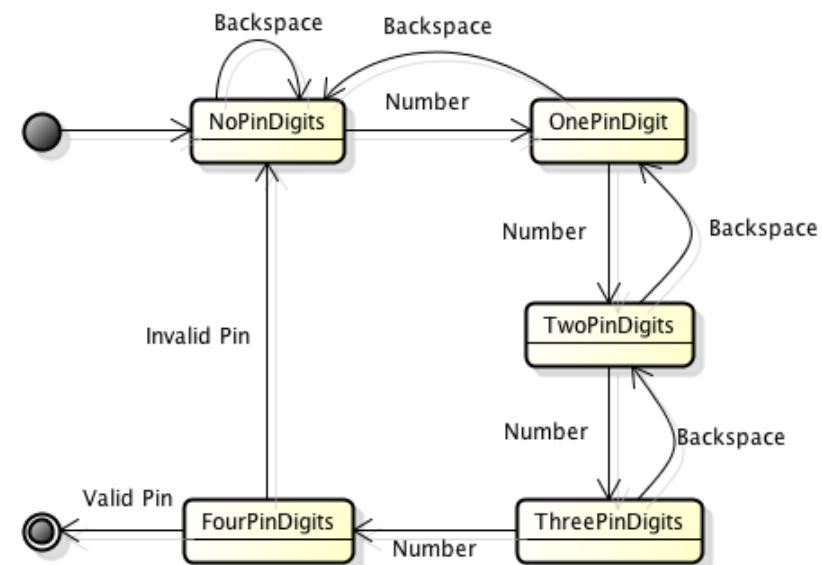
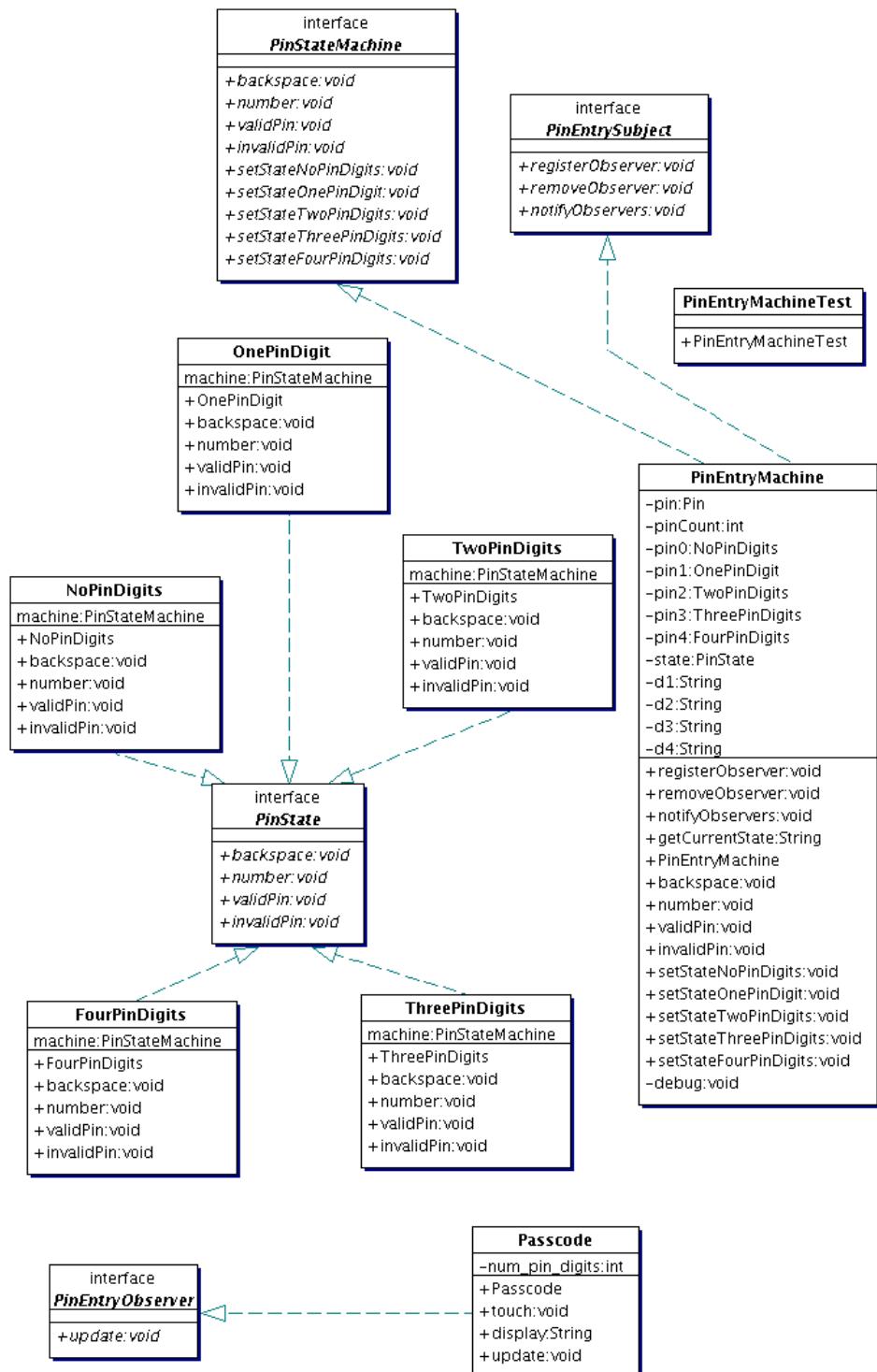


Pin Screen

Model the Pin Screen Behavior as a “State Machine”.

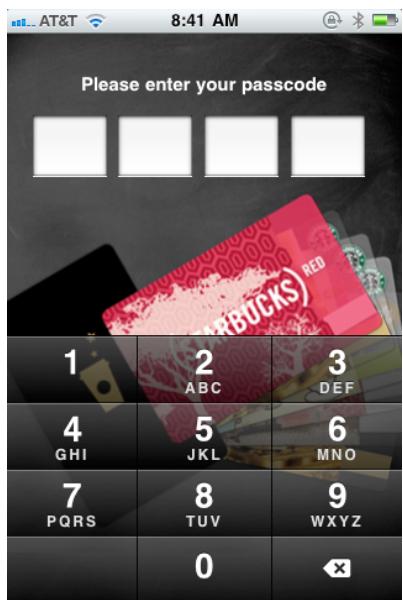


Use a default PIN of “1234” for testing.



Starbucks

Sample Problem Requirements



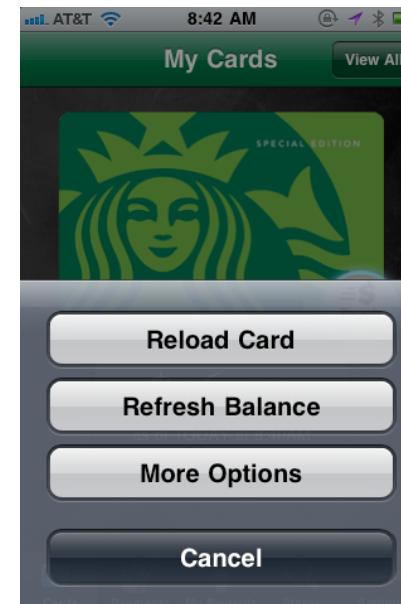
Pin Screen



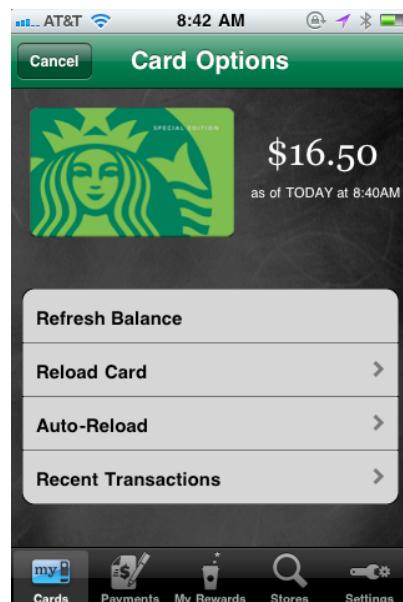
My Cards - Main



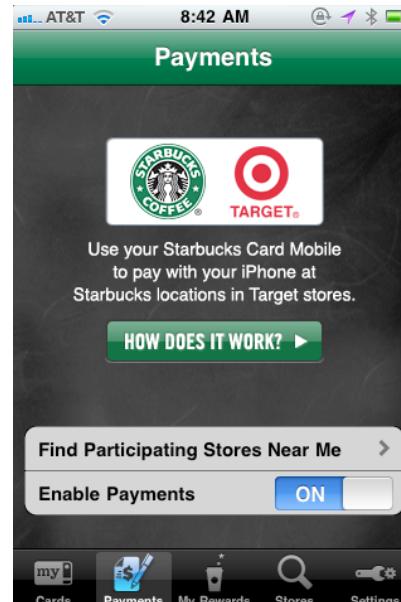
My Cards - Pay



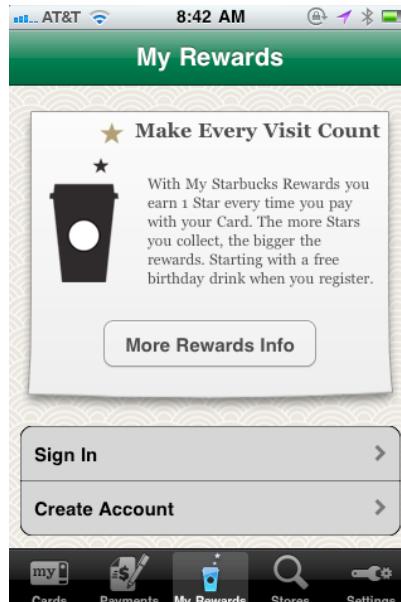
My Cards - Options



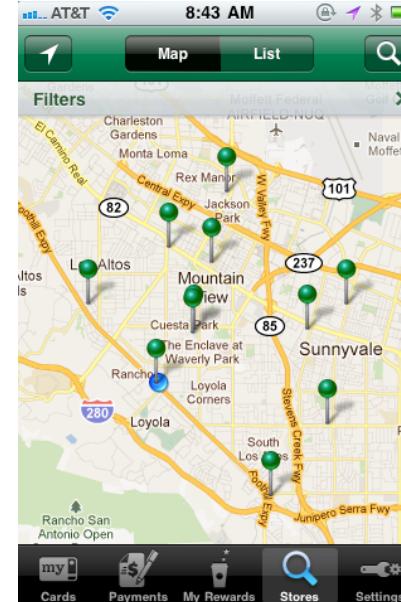
My Cards
More Options



Payment Setup

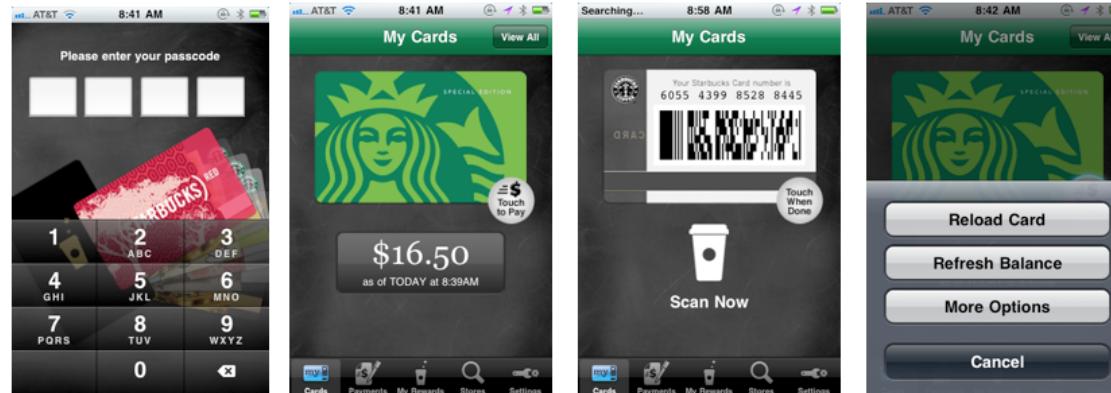


Rewards Setup



Find Starbucks

Solution should implement an “App Controller” class which should contain a “display()” method to display the current ***Screen Name*** as well as well as call the “*display()*” method of the current Screen each time the Screen changes

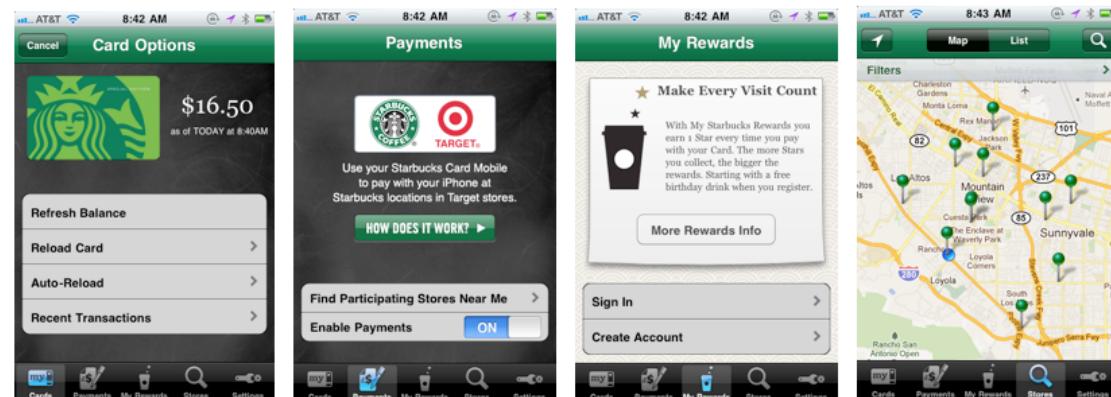


Pin Screen

My Cards - Main

My Cards - Pay

My Cards - Options



My Cards
More Options

Payment Setup

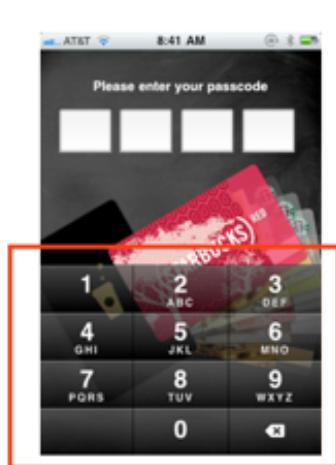
Rewards Setup

Find Starbucks

All Screens should implement a common interface which includes “*display()*” and “***touch(x,y)***” methods.

The “***touch(x,y)***” method accepts the coordinates below. Screens can contain components (such as the “keypad”) which have their own relative coordinates.

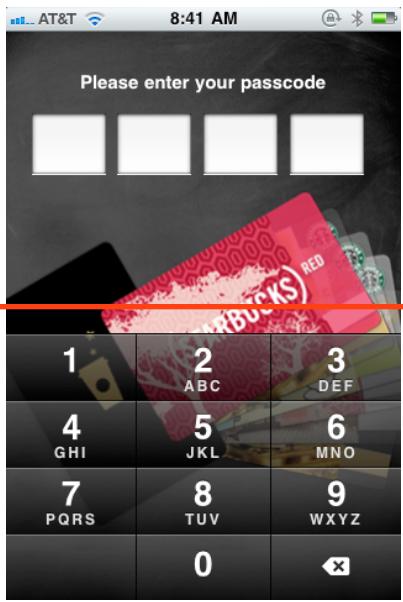
Touch events can be mapped from the Screen coordinates into the widget’s relative coordinates. For example, given the following layout of the PinScreen, a “***touch(1,5)***” would map into a “***keyPress(1,1)***” on the KeyPad.



(1,1)	(2,1)	(3,1)
(1,2)	(2,2)	(3,2)
(1,3)	(2,3)	(3,3)
(1,4)	(2,4)	(3,4)
(1,5)	(2,5)	(3,5)
(1,6)	(2,6)	(3,6)
(1,7)	(2,7)	(3,7)
(1,8)	(2,8)	(3,8)

PinScreen Coordinates (x,y)

Pin Screen



(1,1)	(2,1)	(3,1)
(1,2)	(2,2)	(3,2)
(1,3)	(2,3)	(3,3)
(1,4)	(2,4)	(3,4)
(1,5)	(2,5)	(3,5)
(1,6)	(2,6)	(3,6)
(1,7)	(2,7)	(3,7)
(1,8)	(2,8)	(3,8)

(1,1) = 1	(2,1) = 2	(3,1) = 3
(1,2) = 4	(2,2) = 5	(3,2) = 6
(1,3) = 7	(2,3) = 8	(3,3) = 9
(1,4) = _	(2,4) = 0	(3,4) = X

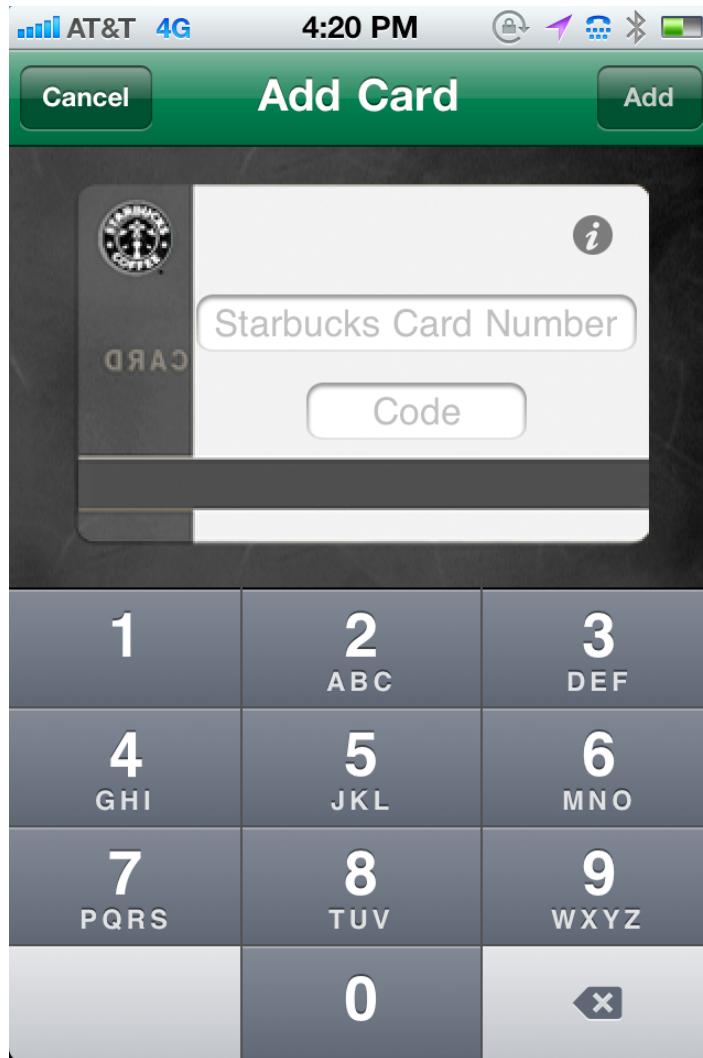
Coordinates: (x,y) = Key

KeyPad
inside PinScreen

Sample Touch Event Mapping:

=>Touch (1,5)
=>KeyPress(1,1)
=>Key Number 1

App Controller should implement “**Top Left Command**” and “**Top Right Command**” which would be mapped to certain operations in a particular screen (if any at all). For example, the **Add Card Screen maps “Top Left” to “Cancel” and “Top Right” to “Add”**.



App Controller also must implement a **Menu Bar** (typically position at the bottom of an iPhone Screen).

This should be specified in your design as an **Interface** that App Controller implements and should contain call signatures for **menu1()**, **menu2()**, **menu3()**, **menu4()** and **menu5()** to represent each of the five menu options an application may have.

For example, menu1() maps to “Cards”, menu2() maps to “Payments”, etc...



AppScreen and MenuBar Interfaces

AppScreen Interface:

```
touch( x, y )
display()
topLeftCmd()
topRightCmd()
```

MenuBar Interface:

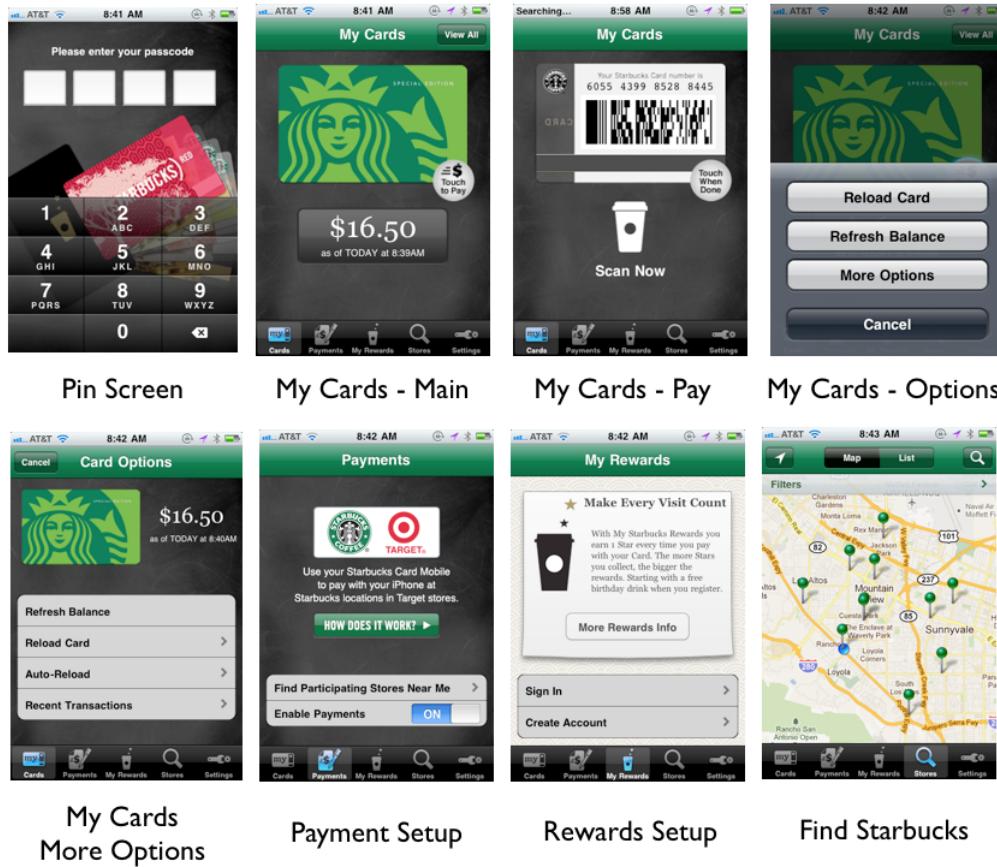
```
menu1()
menu2()
menu3()
menu4()
menu5()
```

All Screens must implement the “AppScreen” Interface.

AppController must implement the “AppScreen” and “MenuBar” Interfaces.

NOTE: The MenuBar should not be active until after the user has successfully authenticated with a valid Pin from the Pin Screen.

Screen Flows



Implement the following Screen Flows and Touch Behaviors:

1. After a “Successful” Pin Validation, the “My Cards – Main” Screen appears
2. A **touch(3,3)** on the “My Cards – Main” screen switches to the “My Cards – Pay” Screen
3. A **touch(2,4)** on the “My Cards – Main” screen switches to the “My Cards – Options” Screen
4. A **touch(1,7), touch(2,7) or touch(3,7)** on the “My Cards – Options” Screen
switches to the “My Cards – More Options” screen
5. A **touch(3,3)** on the “My Cards – Pay” screen switches to “My Cards – Main” Screen.

Display Requirements



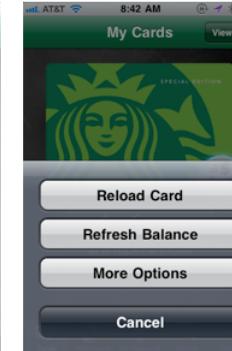
Pin Screen



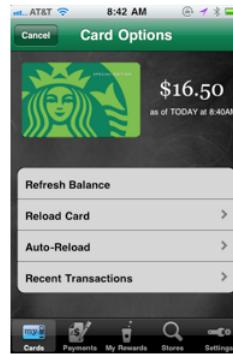
My Cards - Main



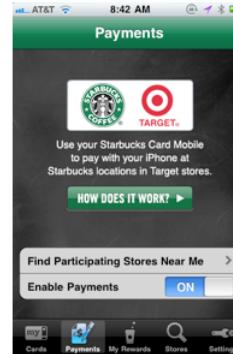
My Cards - Pay



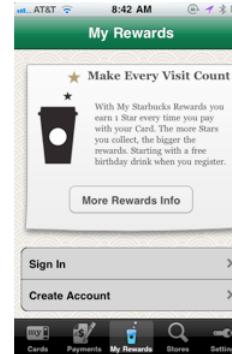
My Cards - Options



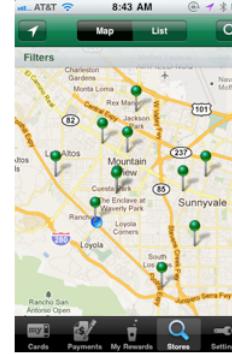
My Cards
More Options



Payment Setup



Rewards Setup



Find Starbucks

Implement the following Display Output for each screen:

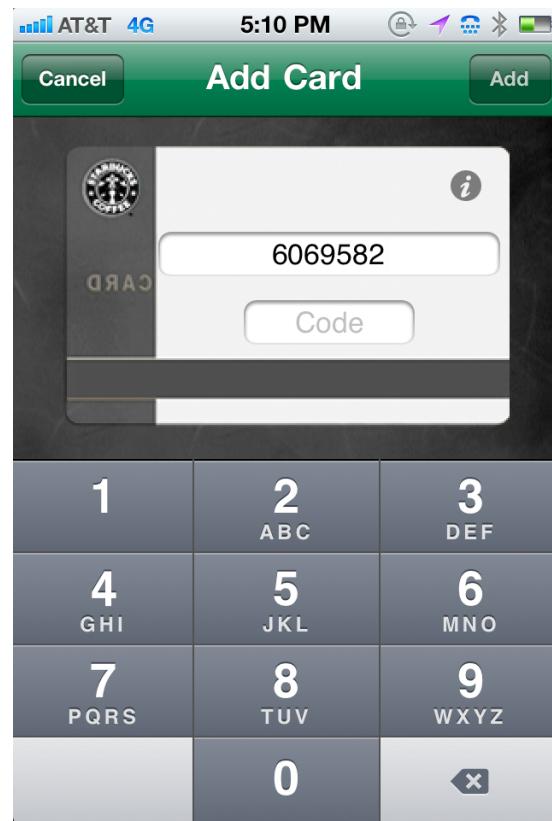
1. Calling "display()" on the "My Cards - Main" Screen should **show the balance of the current active card**.
2. Calling "display()" on the "My Cards - Pay" Screen should print-out the words "**Scan Now**"
3. Calling "display()" on the "My Cards - Options" Screen should print-out the words "**Reload, Refresh Balance, or More Options**".
4. Calling "display()" on the "My Cards - More Options" Screen should print-out the words "**Refresh, Reload or View Recent Transactions**"
5. Calling "display()" on the "Payment Setup" Screen should print-out the words "**Enable Payments?**"
6. Calling "display()" on the "Rewards Setup" Screen should print-out the words "**Make Every Visit Count**".
7. Calling "display()" on the "Find Starbucks" Screen should print-out "**Google Map of Local Starbucks**"

Add Card Screen

Add the following “**Add Card Screen**” which reuses the KeyPad widget. In the “Add Card Screen”, 16 digits can be entered for the Card ID and 8 digits can be entered for the Card Code.

A touch() with coordinates of (1,3), (2,3) or (3,3) should set the Card ID entry as the current focus. Likewise, touch (2,4) will switch the focus to the Card Code. With the focus of the cursor, new digits “touched” should be appended to the current “Card ID” or “Card Code” numbers.

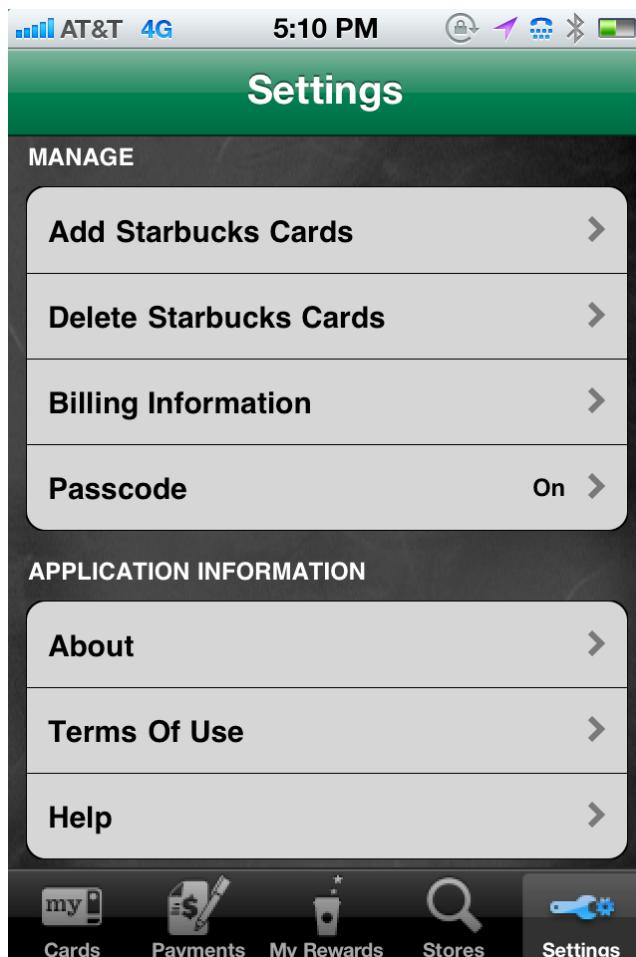
The “**Top Right Cmd**” on the Add Card Screen maps to “**Submit the Add New Card Request**” with the digits entered and the “**Top Left Cmd**” maps to “**Cancel Card Entry**” and return to the previous screen.



Settings Screen

To add a card, a user would touch the “**Settings**” menu and then the “**Add Starbucks Cards**” option. This option has the following touch coordinates: (1,2), (2,2) and (3,2). Design a **Detailed Sequence Diagram** for this workflow, which should result in creating a new Card object making it the new “Default” card for payment.

Additionally, the “**display()**” method for **Add Card Screen** should print “**Enter a new Card**”; likewise, the “**display()**” method for **Settings Screen** should print “**Manage Card, Billing, Passcode. Show About & Terms.**”



NOTE: Add all new cards with a default initial balance of \$16.50.

Making Payments

For Payments, “hard code” and **charge of \$1.50 for each transaction** made with the current card.

The following touch(x,y) coordinates should trigger the payment.

`touch(1, 2), touch (2,2), or touch (3,2)`



Unit Testing

Create Unit Tests for:

1. ***Pin Entry State Machine.*** Your unit tests should cover all state transitions and must validate the correct outcome of each transition from state-to-state.
2. Menu Selections for ***App Controller.*** Your unit tests should cover all “Commands” and should verify that the correct Screen is set as the current screen.

Patterns

1. In addition to the “touch(x,y)” and “display()” methods, the **App Controller** should also implement the following methods for the menu ribbon: menu1(), menu2(), menu3(), menu4() and menu5(). Using the **Command Pattern**, wire these menus to switch the current screen to: My Cards - Main, Payments, My Rewards, Find Starbucks and Settings, respectively. For example, calling menu1() on the App Controller should result in the display of the “My Cards – Main” screen.
2. Implement the **State Pattern** for your Pin Entry process. The State Machine should track the number of pins entered and automatically try to validate the pin upon the fourth touch. Additionally, implement the **Observer Pattern** for the Passcode Display assigning the Observer responsibility to the Passcode Display widget, which echos the number of digits entered so far upon being notified by the change by a Subject object.
3. Implement the **Singleton Pattern** for the App Controller.
4. Implement a **Chain of Responsibility** for Widgets composed within the **Pin Screen** and **Add Card Screen**. The objects within the chain should determine if the “touch(x,y)” event is interesting to it and may have to **Adapt (i.e. Adapter Pattern)** the event to the target widget for processing. For example, the Pin Screen may have two objects in the event chain, one what “wraps” Passcode Display widget and another that wraps the Key Pad widget. The Passcode Display receiver ignores all touch events since it is an output only widget. The Key Pad receiver will convert touch(x,y) events into keypad events if it determines that the touch is over the KeyPad region.