Team 23 Final Document

Customer Loyalty Program for Small Businesses

Clients - Jay Namboor
Adviser - Dr. Govindarasu

Members:
Christopher Waters
Van Nguyen
William Tran

Table of Contents

Abstract & Problem Statement

Businesses often utilize paper punch cards to incentivize customers to return by offering rewards after a certain number of purchases. However, these cards are easy to lose or forget about entirely. They also have difficulty attracting new customers.

We propose creating a mobile customer loyalty application to increase business for small businesses. Such an application will encourage customers to patronize local businesses by offering them reward incentives on goods and services they already purchase. It will aim to replace existing reward incentives such as punch cards and coupons by providing a more convenient system for both the consumer and business.

Our solution is to create a system for creating and obtaining and managing punch cards through a smartphone or tablet device for customers and businesses. Cards are represented with an in-app QR code that the business will scan using a device running Android or iOS and will allow them to update the customer's card with a punch.

Our primary customer will be iapps24, a smart-phone application development company based in Des Moines who have sought Iowa State University students to begin development on this application.

Concept Sketch

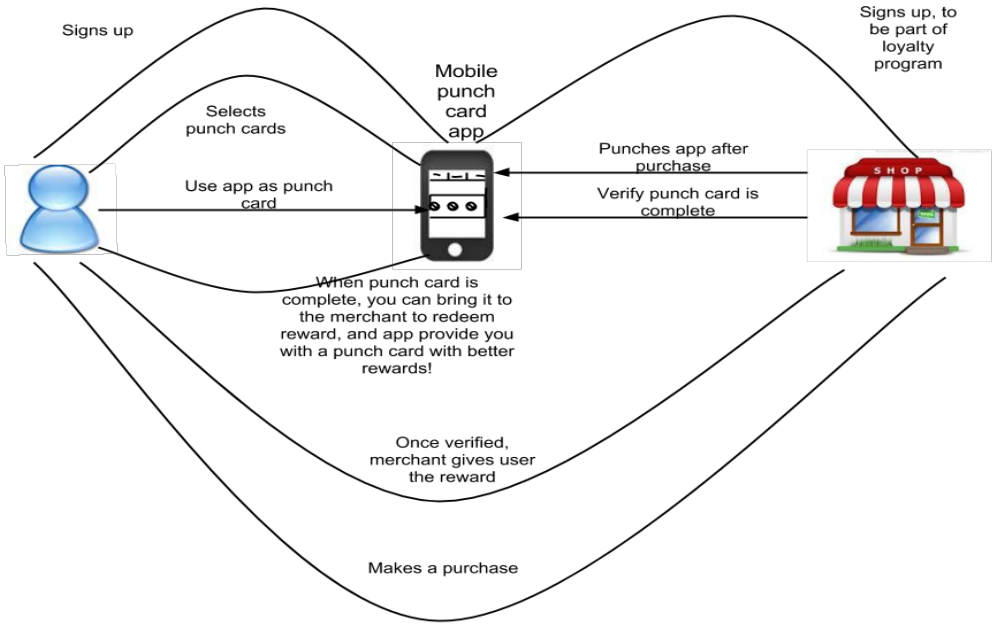


Figure 2.1. Shows how the user and the merchant can interact with our application

Figure 2.2. Shows the concept of our system storing reward cards in the phone and representing them as QR codes.

Use Cases

Use Case	Primary Actor(s)	Precondition	Summary	Outcome
UC-1 Create account	Customers and merchants	Users have to download the app	Customers/Merchants enter a username, password, and select a role that they want for a new account to be created	New account is created and it's going to be added to the system
UC-2 Login/logout	Customers and merchants	Must be a member	Customers/Merchants enter their username and password	System will redirect user according to their role.
UC-3 Create loyalty card	Merchants	Must be logged in.	Merchants can manage/create loyalty card.	New loyalty card is created/edited into system.
UC-4 Expire loyalty card	Merchants	Must be logged in.	Merchants can expire a loyalty card if they choose to.	Loyalty card is no longer valid.
UC-5 Verify punches	Merchants	Customer must have downloaded the loyalty card for that merchant.	The punch will be verified.	If it is successfully verified, user's card will get updated with punch.
UC-6 Verify reward	Merchants	Customer must have collected all required number of punches.	Merchant will be able to verify the user has collected all the punches.	Once verified, user will get the reward
UC-7 Search businesses	Customer	none	Users can search for restaurants based on the name, by location, by gps.	The system will return a list of businesses based on the search criteria.

UC-8 Selects a card	Customer	Must be logged in	User can select loyalty cards they want from specific businesses.	Loyalty card will be selected.
UC-9 Download card	Customer	Customers must have selected a card.	Customer wants to download the specific loyalty card they selected.	Card is downloaded and stored into the system.
UC-10 Redeem punch	Customer	Customer has to have purchased a product	Customers getting a punch after purchasing a product.	Punch is updated into card and updated in the system.
UC-11 Redeem reward	Customer	Customers must have a completed punch card	Customer has completed punch card.	Customer may let store verify, if successful, they will be given the reward. Reward redemption will be stored in the system
UC-12 Store history	Database	none	Redeemed loyalty card is stored into the system database	The system can now return the history of a loyalty card if requested.

Functional Decomposition

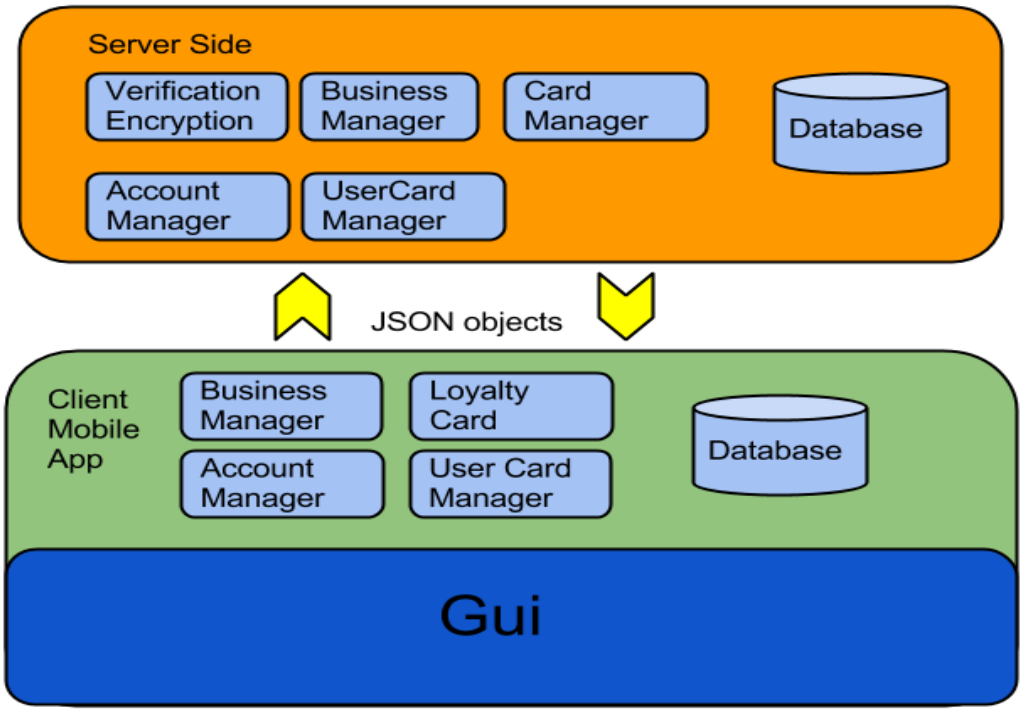


Figure 3.1. System Block Diagram

System Description

We are looking to create mobile customer loyalty application. The application will allow for merchants to create reward based incentives for their business. The rewards will be based upon a traditional punch card system, but stored electronically in the user's phone. The merchants will also be able to expire rewards and set expiration dates at any time. (Each reward card will have a disclaimer for what happens if the reward is not available at the time of redemption). Merchants will be able to verify punches and reward redemptions at the time of purchase.

Customers will be able to search the app for available punch cards at local businesses by GPS and name based searching. When a customer finds a punch card they plan on using, they can download it directly to their phone. The punch card will have information on what type of purchase will qualify for a punch. When a customer earns the required number of punches, they will be able to collect a reward.

History of punch cards will be stored on a server database. Merchants will be able to view information about the number of punch cards redeemed at their store, but not about the individuals who redeemed them. Customers can view their personal history of redeemed punch

cards.

Operating Environment

The system will run on Android and iOS mobile devices. The mobile devices will utilize a local SQLite database to store information about a user's current loyalty cards. There will also be a MySQL database on a server that will store information about businesses, completed card history, and loyalty card templates. Merchants will be able to create loyalty card templates from their mobile device and upload them to the server. Customers will be able to download loyalty cards to their phone and store the information in their local database. Once downloaded, the customer can collect punches on qualifying purchases. After a predetermined number of punches, the customer earns a reward.

User Interface Description

The User Interface for the system will be displayed on a mobile device with a touch screen display. The user will be presented with menu items and icons to navigate the system, including a tabbed interface to switch between user profile, card information, and business searching.

Functional Requirements

FR-1	*The system shall allow the user to login and logout.*
FR-2	*The system shall allow the user to create an account*
FR-3	*The system shall allow merchants to create loyalty cards*

FR-4	*The system shall allow merchants to expire existing loyalty cards*
FR-5	*The system shall allow merchants to verify customer punches*
FR-6	*The system shall allow merchants to verify customer rewards*
FR-7	*The system shall allow customers to search for local businesses*
FR-8	*The system shall allow customers to select loyalty cards*
FR-9	*The system shall allow customers to download loyalty cards*
FR-10	*The system shall allow customers to redeem punches*
FR-11	*The system shall allow customers to redeem rewards*
FR-12	*The system shall allow the database to store a history of completed punch cards*
FR-13	*The system shall allow the merchant to view card history for their business*
FR-14	*The system shall allow the customer to view personal card history for completed rewards*

Non Functional Requirements

NFR-1	*The system shall be developed using the Appcelerator SDK*
NFR-2	*The system shall use an encrypted SQLite database for local storage to prevent other applications from interacting with a user's card data*
NFR-3	*The system shall utilize hashing for punch a reward verification*

Market Survey

Our mobile rewards platforms rests on the cusp of two of the largest and most rapidly expanding industries- online advertising and the mobile application space. Analyst firm research2guidance estimates that the mobile application space will become a "15.65 billion [US Dollar] business in 2013." The New York Times reports that in 2009 "worldwide spending on mobile advertising … amount[s] to … 1.4 billion [US Dollars]." Together the market is worth nearly 18 billion US Dollars. The current state of the mobile phone market reflects that Apple, Android and Windows phone account for 69% of the market. However, with Blackberry use in sharp decline and windows phone undergoing steep increase in user-ship, it is clear that the top market contenders will be Apple, Android, and Windows phone. Our cross platform application and will inherently run on Android and Apple phones. The potential for a new and revolutionary product in this market space is staggering.

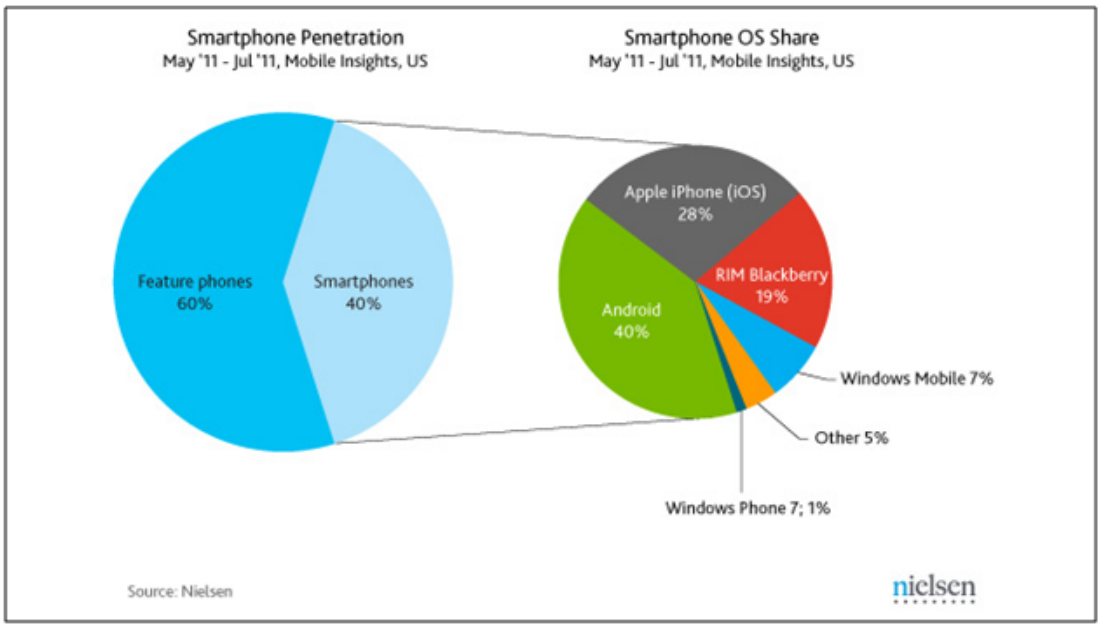


Figure 9.1. This graph demonstrates the market capitalization of the various mobile operating systems.[9] Notice smartphones have 40% of the cell phone market and continues to growth.

i) Mobile Application Space

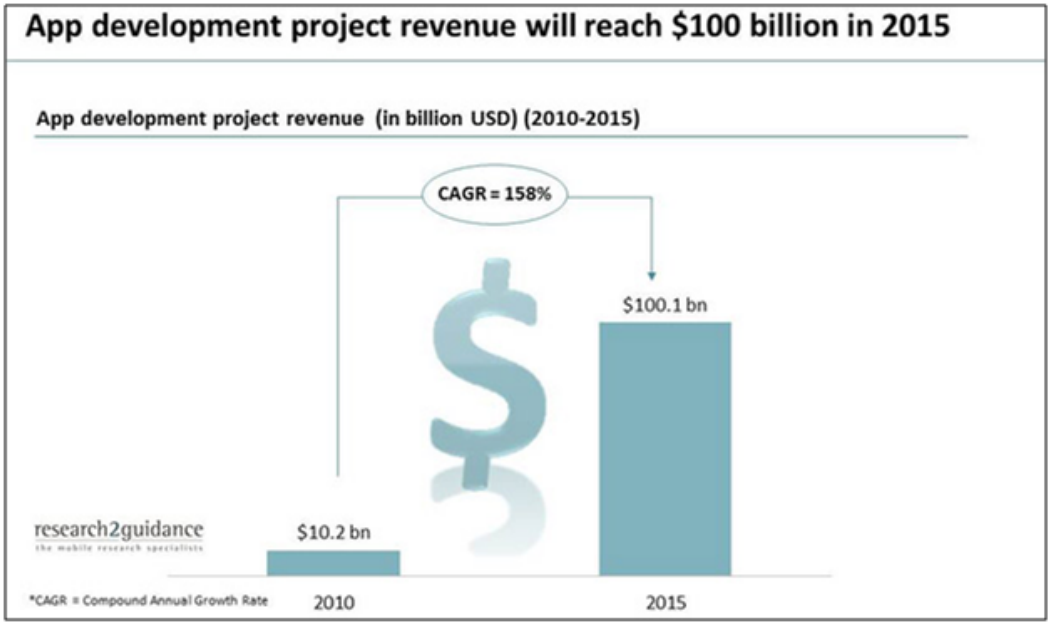


Figure 9.2. This figure illustrates the research2guidance's prediction of the enormous growth in revenue of mobile apps from 2012 to 2015 where it will reach 100.1 billion usd.[7]

Analyst firm research2guidance recently published a study of the mobile application space. The study indicates that the mobile application market will grow from a $1.94 billion US Dollar business in 2009 to a $15.65 billion business in 2013. This growth rate of nearly

807% and is a clear indicator of the current and future strength of this market.Research2guidance also indicates that the smartphone user base to grow from 100 million to 1 billion in that same time frame; this is a 1000% growth rate. This growth rate is an indication of a transition from the laptop or desktop computer to a smaller and more portable medium. Companies will have no choices but to follow this trend, increasing their exposure in the mobile advertising arena especially when we consider that “currently, only 10% of the Fortune 2000 companies are engaging their customer base with a mobile application.” (source: BRG)One can examine a clear and definitive need for an advertising platform catering to companies transitioning to the mobile space.

ii) Mobile Advertising Market

The New York Times reports that in 2009 “worldwide spending on mobile advertising … amount[s] to … 1.4 billion [US Dollars].” The number in 2010 was “$1.6 Billion generated in 2010.”(Source: IT research firm Gartner)However, research2guidance also indicates that the smartphone user base will grow from 100 million to 1 billion in that same from 2009 to 2013; this is a 1000% growth rate and as the number of internet connected user increases this will drive mobile advertising revenue into double digit gain, year afte

Deliverables

These are artifacts that will we will need to turn in.

- Project Plan 1st Revision
- Project Plan 2nd Revision
- Design Document 1st Revision
- Design Document Final Revision
- Project Plan Final Revision
- Presentation
- Customer Loyalty Application for IOS/Android

Work Plan

This section will discuss how the work is broken up into smaller modules. It will also discuss the schedule in which the work is done.

Work Breakdown

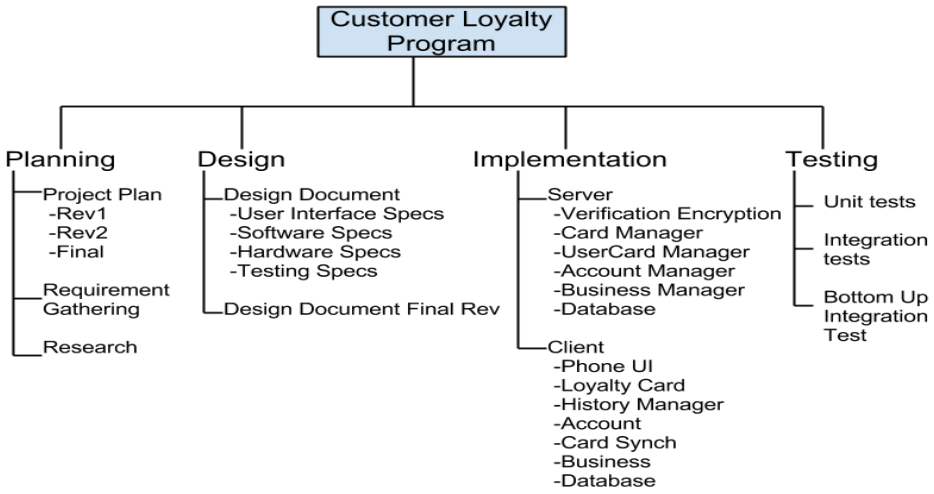


Figure 11.1 Shows our work breakdown structure.

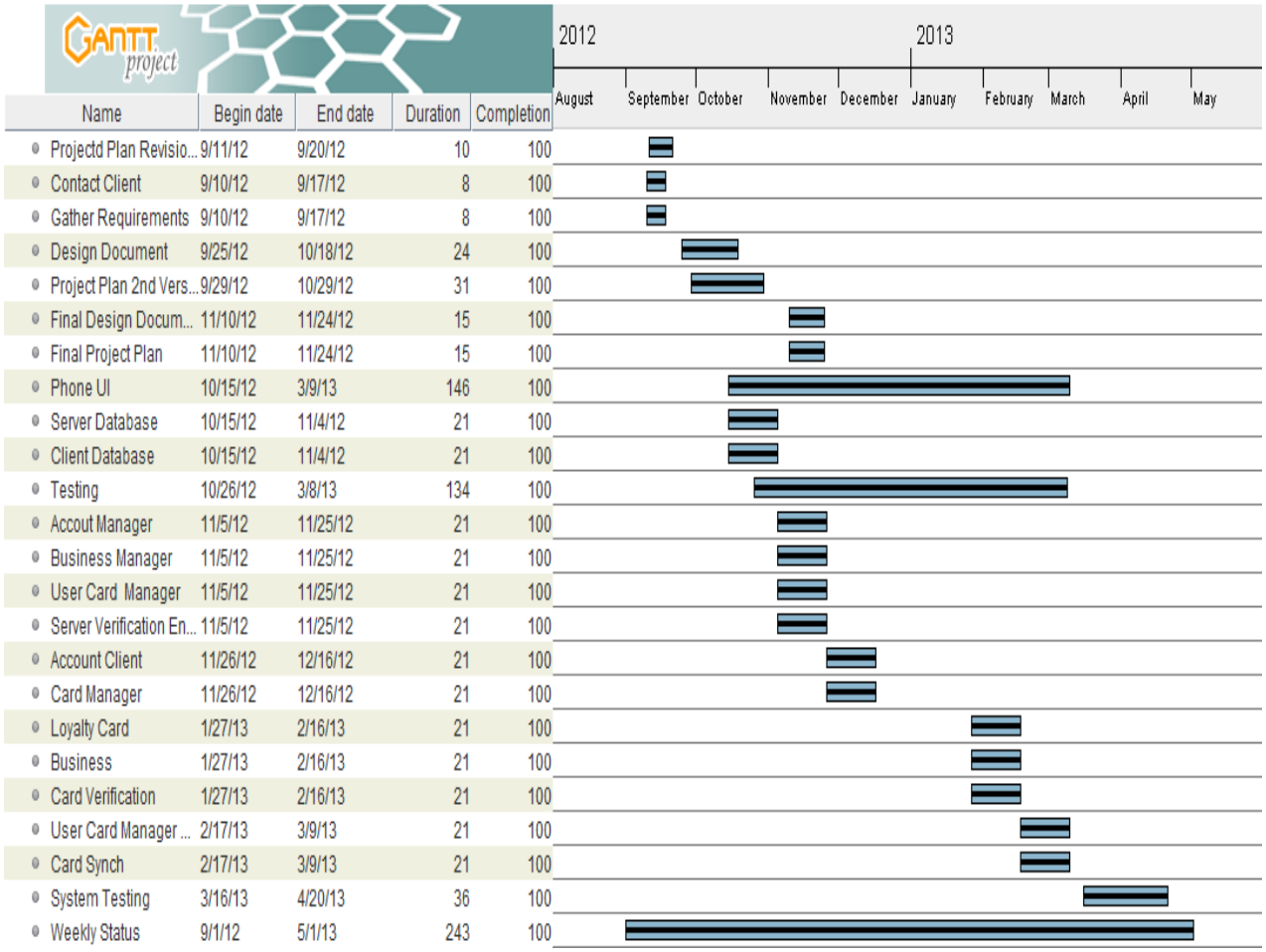
Task Breakdown

Task	Summary	Time Frame	Task Order	Duration
Project Plan Rev1		9/14 - 9/28		
Project Plan Rev2		9/29 - 11/9		
Project Plan Final		11/10 - 11/29		
Requirement		9/14 - 9/28		

Gathering				
Research	Learn technologies required to complete project	Continuing Process		
Design Document		9/29 - 10/27		
Design Document Final Rev		10/28 - 11/29		
Implementation **Server Side**				
Database	Design and creation of database	10/15/12 - 11/04/12	1	3 weeks
Verification Encryption	Handles the redemption of punches	11/26/12 - 12/16/12	4	3 weeks
Card Manager	Manages and card requests	11/26/12 - 12/16/12	3	3 weeks
User Card Manager	Handles storing and retrieving previous completed cards	11/05/12 - 11/25/12	2	3 weeks
Account Manager	Handles anything that involves accounts	11/05/12 - 11/25/12	2	3 weeks
Business Manager	Handles anything that involves businesses.	11/05/12 - 11/25/12	2	3 weeks
Client Side				
Phone UI	Entire UI	10/15/12 - 3/15/13	1	15 weeks
Punch Card/Reward Verification	Module handles verification of punches	12/17/12 - 01/06/13	4	3 weeks

Loyalty Card		01/07/13 - 01/27/13	5	3 weeks
History Viewer	Lets users view previous cards	01/28/13 - 02/17/13	7	3 weeks
Account	Handles the login/logout of user and creation of accounts.	11/05/12 - 11/25/12	2	3 weeks
Business	Handles the searching of businesses	12/17/12 - 1/6/13	6	3 weeks
Database	Stores data such as previous cards.	10/15/12 - 11/04/12	1	3 weeks
Card Synch	Keeps card data synced with server	1/28/13 - 2/17/13	6	3 weeks
Testing				
Unit Testing	Each Module will be tested	Unit testing included during implementation		
System Testing	As Modules are integrated, they will be tested as a system. This could mean modifying any module.	3/16 - 4/20	7	19 weeks

Project Schedule



Name	Begin date	End date	Duration	Completion
Projectd Plan Revisio...	9/11/12	9/20/12	10	100
Contact Client	9/10/12	9/17/12	8	100
Gather Requirements	9/10/12	9/17/12	8	100
Design Document	9/25/12	10/18/12	24	100
Project Plan 2nd Vers...	9/29/12	10/29/12	31	100
Final Design Docum...	11/10/12	11/24/12	15	100
Final Project Plan	11/10/12	11/24/12	15	100
Phone UI	10/15/12	3/9/13	146	100
Server Database	10/15/12	11/4/12	21	100
Client Database	10/15/12	11/4/12	21	100
Testing	10/26/12	3/8/13	134	100
Accout Manager	11/5/12	11/25/12	21	100
Business Manager	11/5/12	11/25/12	21	100
User Card Manager	11/5/12	11/25/12	21	100
Server Verification En...	11/5/12	11/25/12	21	100
Account Client	11/26/12	12/16/12	21	100
Card Manager	11/26/12	12/16/12	21	100
Loyalty Card	1/27/13	2/16/13	21	100
Business	1/27/13	2/16/13	21	100
Card Verification	1/27/13	2/16/13	21	100
User Card Manager ...	2/17/13	3/9/13	21	100
Card Synch	2/17/13	3/9/13	21	100
System Testing	3/16/13	4/20/13	36	100
Weekly Status	9/1/12	5/1/13	243	100

Figure 11.1 Project Schedule - Please note: Shaded black means percentage done. Also, some modules are done before the scheduled starting date,meaning we are ahead of schedule. We will be pulling in modules as needed if they are finished early.

Project Team Information

Project manager: Christopher Walters
He is in charge of setting up weekly meeting with the advisor and the client. He also managing communication between client, advisor, and team members.

Project planner: William Tran
He is in charge of weekly report, team weekly meeting, and divided up tasks among team members.

Website master: Van Nguyen
She is in charge of creating, designing, and update team website.

Developer: Van Nguyen, Christopher Walters, and William Tran
All of us are in charge of creating the project plan, design documents, gathering requirements, and implementing the application.

Communication Management

Team Communication:

Communication Method	Time	Summary
Weekly meeting	Sunday @ 5:30-6:30, Wednesday @ 7-8	Meet on sunday to go over our to do list and distribute works among team members. Meet on Wednesday for works progression update and add more works.
Email	Anytime	Use email for questions or rescheduling purpose

Advisor and Team Communication:

Communication Method	Time	Summary
Weekly meeting	Wednesday @1-2	Meet on Wednesday to provide the advisor our work progress and future plan.
Email	Anytime	Use email for questions

Client, Advisor, and Team Communication:

Communication Method	Time	Summary
Weekly meeting		Meet once a week to provide the client our work progress, future plan, and ask client questions or answer their questions.
Email	Anytime	Use email for questions

Risks and Risk Management

Risk	**Mitigation**
R1. Losing a team member	We will contact our advisor.
R2. Limited experience of Appcelerator	We will begin learning the Appcelerator framework as early as possible
R3. None of us have a mac to do IOS development.	We will use macs on campus for development.
R4. The schedule planned might not be realistic, because of R2.	We will begin implementation as modules are designed.

System Analysis

While our system is mainly a client-server architecture, it is however, somewhat unique in that both the server and client side will house some of its own data. The reason for this is to provide some offline functionality. The offline functionality is needed when the user is “punching” the card. If we were to depend on an internet connection, that may slow the checkout process for

the customer and the merchant. We do not want the application to cause the business to lose customers because it is trying to connect to the internet.

That is why cards can be downloaded on to the client locally. The client itself will be responsible for keeping track of the punch cards. It will handle everything from punching the card to redeeming the reward.

Because of the offline capability, we proposed a cloud synchronization solution. If the user were to ever lose their phone, they would not lose all of their punch cards. The Card sync module will interact with the History Manager on the server side to ensure data on both the client and the server are up to date.

Server Side

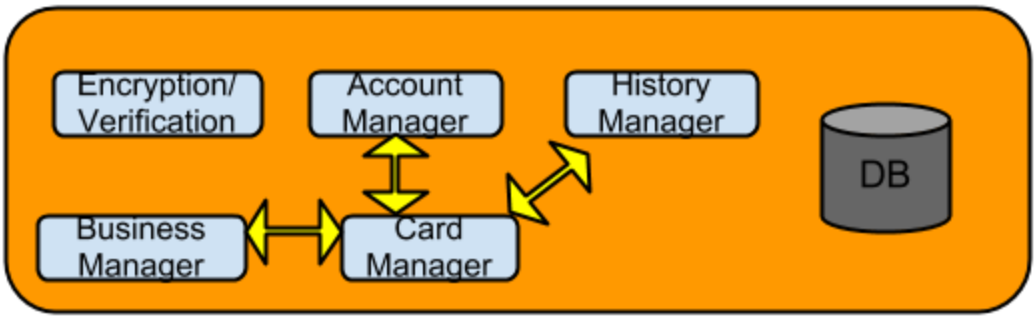


Figure 2 - Notice how every module within the same layer can interact with one another. Same goes for the client layer. For example if Business Manager needs to get the card that is owned by it, it can call on the card manager to get that card from the database.

The responsibilities of the server include storing all of the user and the businesses. The way each module is setup is based on their specific responsibilities. For example if a user wants to download a card, it will eventually have to go through the card manager to get that card. For more information on a module, please refer to the input/output specifications.

Interaction between client and server

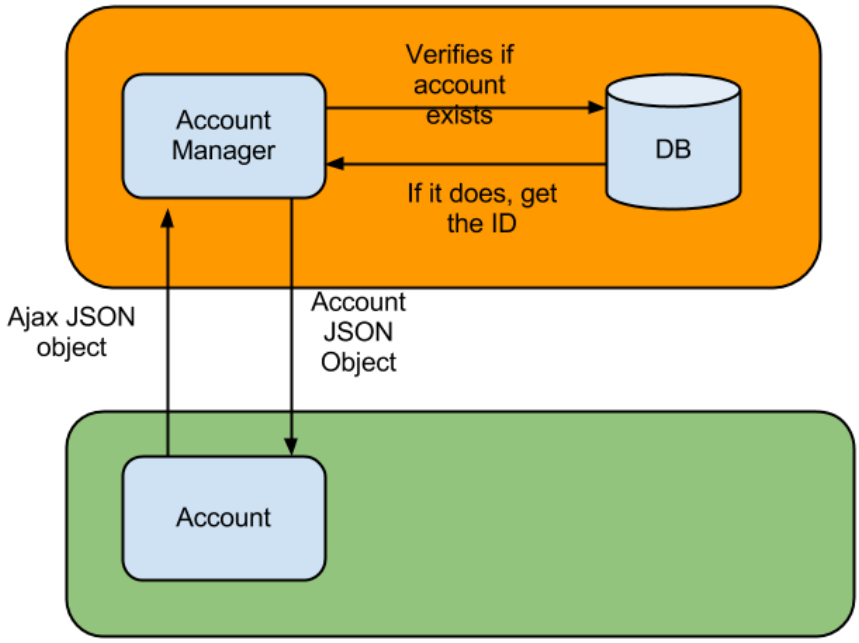


Figure 3 - Shows interaction between server and client

Interactions between client and server will done through JSON.In the example above, a user is logging in. The account module on the client side will ajax a GET request with an JSON param object { action : <method> , object : <data>} to the Account Manager on the server side. The action tells which method to call from the controller, and the data in this case is an Account Object with a user name and password. The Account Manager will parse out the Account object and verify if it exists in the database. If it does, the Account Manager will return an Account Object with a id > 1 and -1 otherwise. All communication with the server will be done in a similar fashion.

Software Design Patterns

1. Model–view–presenter

We are using the model-view-presenter design pattern in order to separate logic from the view from the model. A picture below describes the mvp structure.

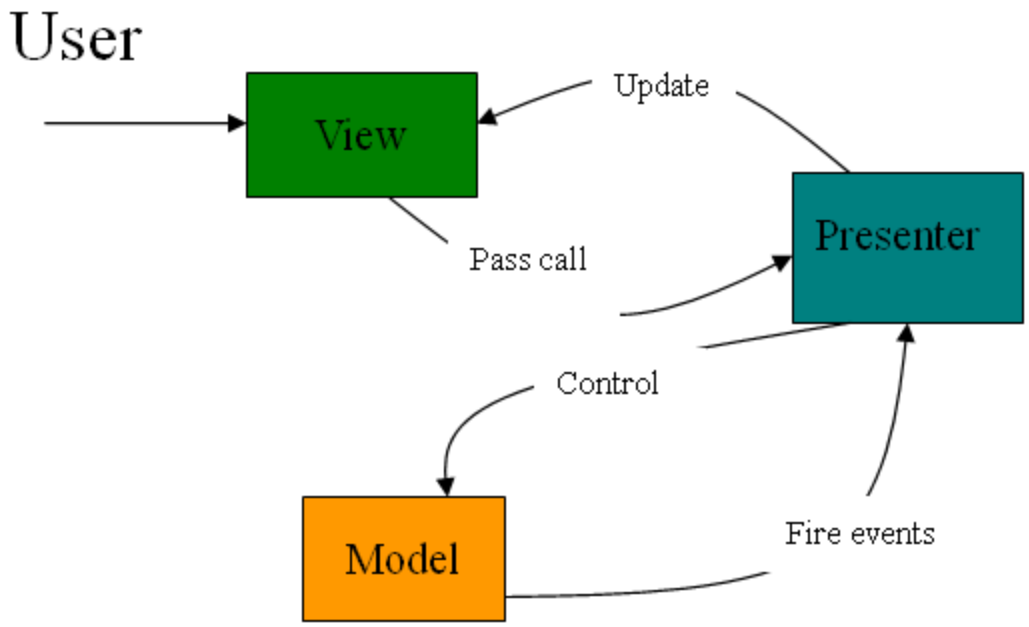


Figure 4 - Shows the MVP pattern (Source : http://dotnetslackers.com/articles/silverlight/A-WaspKiller-Game-with-Silverlight-3-NET-RIA-Services-MVP-and-MVVM-Patterns-Part-1.aspx)

In this case, the view is all of our UI. Whenever their UI needs to update/request information, they will need to call a specific controller(presenter) assigned for handling certain jobs. That controller will then update/retrieve the data and do any logic/format changes to the data before it is passed back to the View. The Model ‘s only job is to update/retrieve data. The Views job is only to display data. The controller will handle all the logic. Once again because of the client-server architecture along with the offline capability, we almost have two-part controllers in this case. Think about this, when the user attempts to create an account, the GUI will need to interact with the account controller on the client side. The account controller on the client side will then quickly do input checking ...etc , before sending the data to the Account manager controller on the server side to store and verify the account has been created.

Input/Output Specifications

Server Side

Each input for the modules shall be wrapped around a JSON Object and converted to a string before it is sent over to a server in this format.

JSON Object - Which contains the data .
For example account object would have
object = { username : <value> , password <value> }

Which is then wrapped around another object to be sent over with another action attribute.
The action key specifies which function to call within the controller.
param = { action : <value> , object <data object> }

and finally this is passed to a ajax call by { param : Stringify(param) }

Example : A call to the Account Manager Controller for a login

```
var account = { username : "hello" , password : "goodbye"};
        var param = { action : 'Login', object : account };
        $.ajax({
                type: 'GET',
                url : 'http://localhost:81/loyaltyprogram/AccountManager.php',
                data : { param  : JSON.stringify(param)},
                success : function ( data){
                        alert(data);
                }
        });
```

Card Manager

The card manager module will handle creation of cards and also retrieving cards.

Module Function	Input	Output
Store created loyalty card onto the server	Title/Description/Number of punches/Level/Expiration Date/Place ID, download by date	Verification that loyalty card is created and added it to the server

Retrieving loyalty cards from server	Place ID	List of loyalty cards information that match with the Place ID
Update card template	Place ID, card metadata, and download by date	Verification that loyalty card is updated correctly

Test Case	**Success**	**Fail**
User(merchant) Create loyalty cards for their business	Loyalty card does not already exist for specific business in the database.	Loyalty card already existed in the database
User(Customer) Selected loyalty card from specific business	Loyalty card is not expired and it placeID match with specific	Loyalty card exists or placeID does not match with specific business ID.

Account Manager

The account manager will contain code to create, lookup accounts, & reset passwords. When creating an account, users will input an email and password combination. If a user already has an account, they can login by submitting their email and password combination.

Module Function	**Input**	**Output**
User login	Email/Password	Verification that a user is logged in (*Session?)*
User creates account	Email/Password	Stores account in database and notifies user.
User resets password	Reset password	Password is reset and sent to user's username

Test Case	**Success**	**Fail**
User creates account.	email does not already exist in the database.	email already exists in the database
User login	email and password combination exists in the user database	email and password combination does not exist in the user

User resets password	user password resets	

User Card Manager

The User Card Manager will be responsible for maintaining a instances all cards that have been downloaded to a user's phone. When the card is redeemed, the database entry corresponding to that card will be marked as completed. It will also be able provide meta data with a user's phone to keep track of partially complete cards so a user can download their cards in case they lose access to their phone (e.g. phone breaks and they get a replacement phone or they upgrade phones).

Module Function	**Input**	**Output**
Download Card	Card Object	Card Object with QR Code
punchCard	Card Object	Card Object with updated punches
Retrieve Card	*Card Object*	Card Info
Redeem reward	Card Object	Card Object with updated state
Get Card For User	Card Object with user_card_id	Card Object matching user_card_id
Delete Card	Card Object	Deleted Card

Test Case	**Success**	**Fail**
Add Card	Card is added to Database	Card is not added to Database
Update Card	Card is updated in Database	Card is not updated in Database
Retrieve Card	Uncompleted cards are downloaded to a user's phone	Uncompleted cards are not downloaded to a user's phone
View History	User receives a list of cards matching search criteria	User does not receive a list of cards matching search criteria

Test Case	Sucess	Fail
Download Card	Card Object with QrCode	Card Object with QR Code
punchCard	Card Object with updated punches and Card.id > 0	Card not updated with punch
Retrieve Card	*Card Object*	decrypted QR code does not match
Redeem reward	Card Object with state = 2 and and Card.id > 0	Card Object with updated state
Get Card For User	Card Object matching user_card_id	no Card matching user_card_id
Delete Card	Card Object deleted	card not deleted

Business Manager

The business manager module will handle creating and linking up business to users. It will also handle request for anything information involving businesses.

This module must :

1. Create a business and link up to user account
2. Link up other accounts to business
3. Retrieve businesses of a user
4. Search request for business based on name
5. Search for businesses based on location - not implemented yet
6. Retrieve top 5 most downloaded businesses
7. Retrieve recommended by on category
8. Return categories available for creation restaurants.

Module Function	Input	Output
Create a business and link up to user account, and generate qrcode and hash to business and email qr code to user.	UserID/Place Name/Location	Verification that business is created and linked up to an account with qrcode created. (Business JSON with id) -1 -business already exists
Save an edited business	Business info	Verification that business information has changed

Link up other accounts to business	UserIDs/Place/BusinessID	Verification that the business is linked up to those accounts
Retrieve businesses of a user	UserID	List of business information linked up to the UserID
Search request for business based on name	Place name	List of businesses matching with place name with locations.
Search request for business based on location	GPS location	List of businesses near GPS location.
Get business with top 5 most cards downloaded.	nothing	List of op 5 business with most cards downloaded
Get 5 random recommended businesses	category	List of 5 random business with the same category
Get category for creation of restaurants	nothing	list of categories.

Test Case	**Success**	**Fail**
User(Merchant) Creates a business	Business must be validated before it is created. User will be linked up to business Qrcode is generated and emailed to user.	Business creation request is declined.
Merchant edits and saves a business	Business information must changed	Business creation request is declined.
User(Merchant) Links up other users to help manage business.	User has the role(linked to business already) to manage the business.	User does not have the role(linked to business already) to manage the business.
User(Merchant) retrieves his/her businesses to manage.	User is linked to any businesses.	User is not linked to any businesses.
User Search request for business based on name	Business matches that name with in the database.	Business does not match the name with in the database.
User searches for business based on a his/her location.	There are businesses within a range of that location	There does not exists not any businesses with a range of that location.
User is on home page	Returns list top five business	

	with most downloaded cards	
User is on home page	Returns 5 random business with same category	
User creates a business	Returns a list of categories for business creation.	

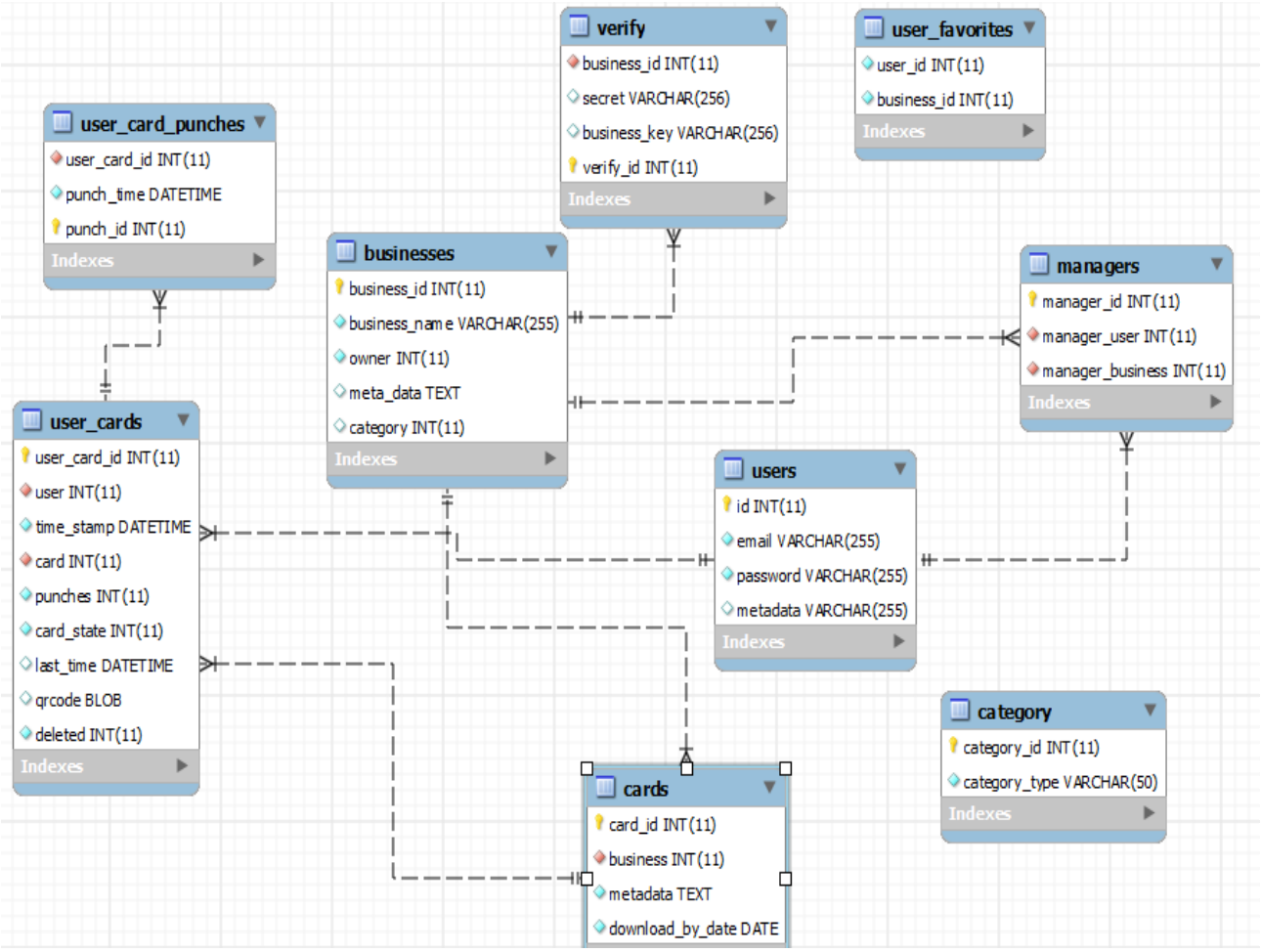
Encryption/Verification

This module is responsible for hashing (sha-256) of input from the user such as passwords. It will also have the responsibility of generating a QR code (with http://phpqrcode.sourceforge.net/) for a specific card for a merchant to do punches.

Module Function	**Input**	**Output**
Hash a user password	User password	Hashed (password)
Generate a QR code	Merchant Name + Merchant ID	QR code of hashed Merchant Name + Merchant ID (salt) = merchantId + name + id

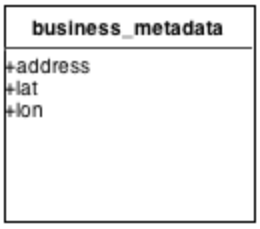
Test Case	**Success**	**Fail**
User password input	Password is hashed	none
Generate QR code	Qrcode (User_card_id) is generated	none

Database



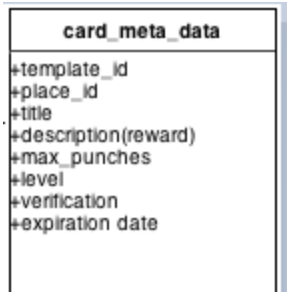
**Each Database will have a corresponding object relating to it.*

Business_meta_data JSON



address- address of the business
lat - latitude location of the business
lon - longitude location of the business

Card_meta_data JSON



- template_id - the id of the downloaded card
- place_id - the id of the card's business
- title - title of the card
- description - description of the reward
- max_punches - max number of punches needed to redeem a reward
- level - level of the card
- verification - the verification code for this card
- expiration Date - the expiration date of this card

Client Side

Account

The account module will be responsible for the login/logout process of the user. It will also handle the creation of accounts.

1. Login
2. Logout
3. Create Accounts

Module Function	Input	Output
User login	username and password	Verification if user is successful or not, and open a session to the user
User logout	user clicks logout	logs user out by closing a session to the user
Create accounts	email, password, and confirm password	send information to server side to create an account

Test Case	Success	Fail
User log into the system	username and password match with username and	mismatching username and password

	password on the server	
User logout of the system	the system close user session	user session still run
Create new account	email and password doesn't exist on the server	email and password exist on the server

Loyalty Card

This module will be use when the customer downloads a selected loyalty card. Once the download card is complete, it will be stored onto the phone database. The module will then be required to handle punches of the a card. It will also mark the card is completed when punches have been filled.

Cards have have only have one of four states

1. Not completed - 0
2. Completed - 1
3. Redeemed -2
4. Expired - 3

Module Function	**Input**	**Output**
Download Loyalty Card	Card Template ID	Download the new loyalty card and store it the phone
Punches card	QR code and timestamp of last punch.	Loyalty point to the card and update the phone database if QR code matches.
Last punch of card	QR code,	Update history module and and mark card as completed in database.
Get current loyalty cards	Selected loyalty card	Gets a list of current loyalty cards.
Get completed loyalty cards	Selected completed loyalty card	Gets a completed list of loyalty cards.

Test Case	**Success**	**Fail**
QR code verification	hashed value of QR code matches the one in the database with the specified card id.	hashed value of QR code doesn't match the one in the database.

Reward Verification

This module is responsible for the verification of a punch. When the user scans a QR code, this module check the QR code's value against the one in the database.

Module Function	Input	Output
Scan and Verify QR code	Card id, hashed value from QR code	Success or fail

Test Case	Success	Fail
Scan QR code verification	hashed value of QR code matches the one in the database with the specified card id.	hashed value of QR code doesn't match the one in the database.

Business

The business module will mainly interact with the business manager module on the server side to get requested information based on a search criteria.

Module Function	Input	Output
Create a business	UserID, Place name, and location	Verification that business is created and linked up to an account
Retrieve businesses of a user	UserID	List of business information linked up to the UserID
Search request for business	Place name	List of businesses matching with place name with locations

Test Case	Success	Fail
User(merchant) creates a business	Business must be validated before it is created. User will be linked up to business	Business creation request is declined

User(merchant) links up other users to help manage businesses	User has the role to manage the business	User does not have the rule to manage the business
User(merchant) retrieves his/her businesses to manage	User is linked to any businesses	User is not linked to any businesses
User search request for business based on name	Business matched that name within the database	Business does not match the name within the database

User Card Manager

The user card manager module will interact with the user card manager on the server to add and update cards stored in the server's database, download non-completed cards, and view a user's history.

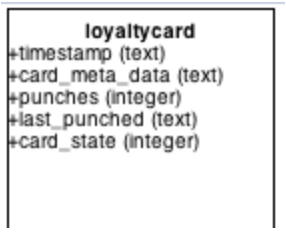
Module Function	**Input**	**Output**
Download Card	Card Object	Card Object with QR Code
punchCard	Card Object	Card Object with updated punches
Retrieve Card	*Card Object*	Card Info
Redeem reward	Card Object	Card Object with updated state
Get Card For User	Card Object with user_card_id	Card Object matching user_card_id
Delete Card	Card Object	Deleted Card

Test Case	**Sucess**	**Fail**
Download Card	Card Object with QrCode	Card Object with QR Code
punchCard	Card Object with updated punches and Card.id > 0	Card not updated with punch
Retrieve Card	*Card Object*	decrypted QR code does not match
Redeem reward	Card Object with state = 2 and and Card.id > 0	Card Object with updated state

Get Card For User	Card Object matching user_card_id	no Card matching user_card_id
Delete Card	Card Object deleted	card not deleted

Client Database

The client database will consist of :



loyalty card table

- time_stamp - When the card was downloaded
- card_meta_data - refer below
- punches - Number of punches a card (Default of 0)
- last_punched - timestamp of last punch
- card_state
  0 - not complete
  1 - completed
  2 - reward redeemed
  3 - expired

card_meta_data will be a string version of this JSON object - explained in server database part

Card Sync

This module shall be responsible for keeping the data on the phone and the server up to date. That way, if the user were to switch to a new phone, the application should automatically download the information they had when using the old phone to their new phone. To accomplish this the module must:

1. Keep server up to date with local information.
2. It must also handle updating even if there is no connection to the internet.

3. Check at login to see if local information matches with server. (i.e. check number of current cards and number of completed cards match). If it doesn't match, it must download information from the server.

4. Must update when:
    When they complete a card.
    When they punch a card.
    When they redeem a reward.
    When they login.
    When they log out.
        -If not internet request to them to save info
    When they close the program.
    When they download a card.

Module Function	Input	Output
Update server with local information	User ID, # of punches, Timestamp of Last Punch(YYYY:MM:DD hh:mm:ss) and, reward claimed (True/False)	Send request to history manager with punch card transaction information
Update even without internet connection	Current and previous punch card transactions.	Send request to history manager with punch card transaction information
Check for matching information with server at login	Current cards	Send over local information such as number of current cards and number of completed cards

Test Case	Success	Fail
Update server with local information	Punch card transaction is logged on to server	Punch card transaction is not logged onto server
Update event without internet connection	card transactions saved locally when there is no internet, all local saved transactions will be log onto server database when there is a internet connection	card transactions not saved locally without internet, or all local saved transactions is not logged onto server database with internet connection

Check for matching information with server at login	Local information from phone and server matches	mismatching information

Software Specification

Technologies

Languages: PHP, JavaScript, MySQL, and sqlite.
Libraries: php qr code, phpMailer,JSON
SDK: Appcelerator Titanium

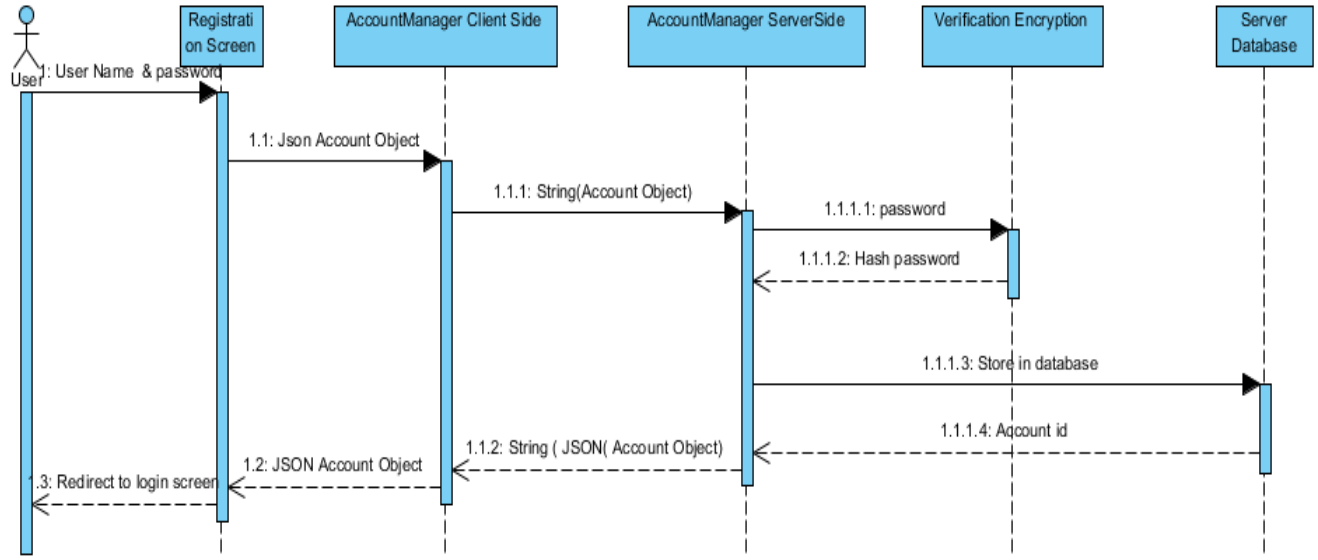
Server : Apache, MySQL, Amazon EC2

Appcelerator Titanium SDK
Using appcelerator, we are able to code once yet have it available for multiple platforms such as android and iOS. We write our client side code in JavaScript and it is compiled to native android and iOS code.

System Test Plan

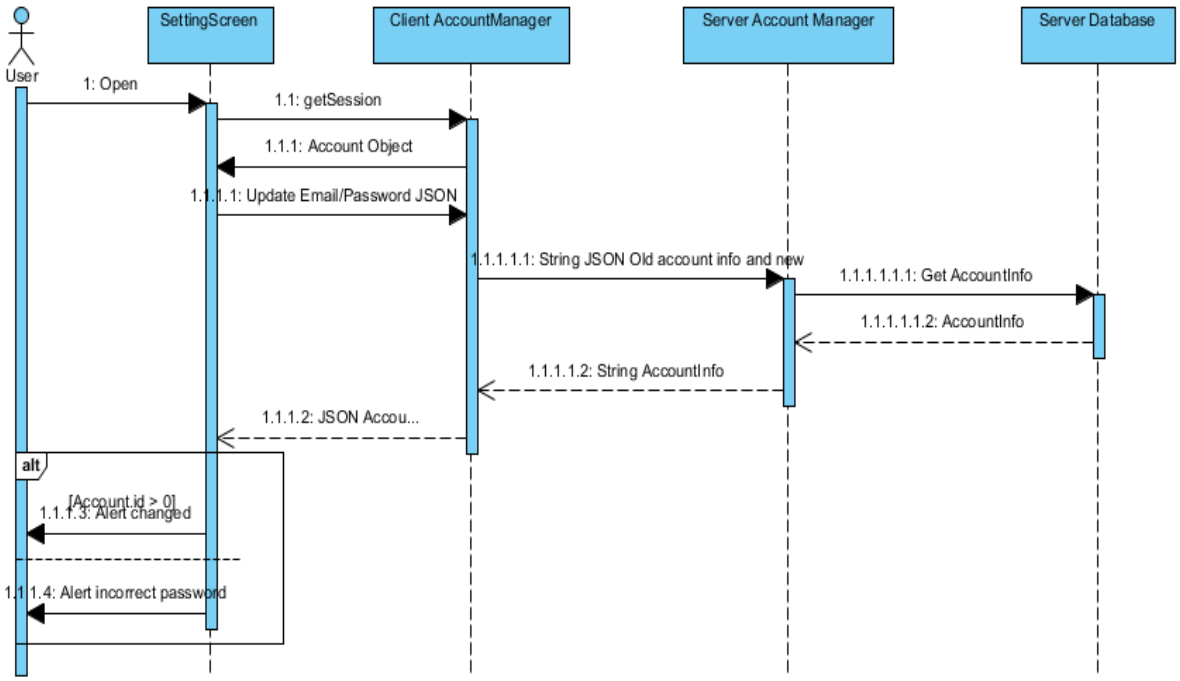
Mainly done through use case scenarios
Use Case: UC1*-Create Account*
Trigger: *User requests to create an account*
Precondition: *None*
Interested Stakeholder: *Customers and merchants*
Actor: *Customers and merchants*
1. *The user will have to input their email, password, and confirmation password.*
2. *When they hit sign up, their name and password will be logged into the database.*

E1.1 The account is already existed in the database
E1.1 The error message will be displayed to the user that the account is already existed.
Output: *New account is created and it's going to be added to the system.*



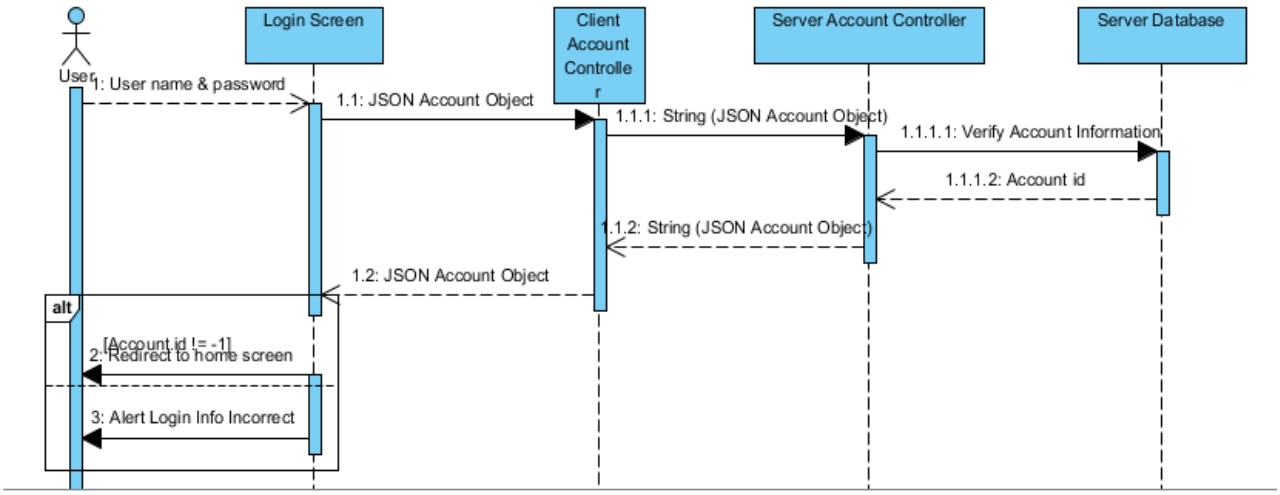
Use Case: UC *-Update Account Info*
Trigger: *Hits update account info in settings screen*
Precondition: *User must have signed up for an account.*
Interested Stakeholder: *User*
Actor: *Database,Client Account Manager , Server Account Manager, Database*
1. *The setting screen will allow the user to enter in username, password, and new password*
2. *This information is wrapped into a JSON object and sent to Client Account Manager*
3. *Client Account Manager will convert JSON into string and AJAX to Server Account Manager.*
4. *Server Account Manager will parse and use information to query database.*
5. *Database will get any account with inputted password hashed and try to match one in the*

database along with user name.
A5 Match means password will be change to new one
6. Account Manager will wrap this info into a JSON and stringify and send back to Client Account Manager.
A6.1 If Information does not exist, Server Account Manager will return Account.id = -1.
7. Client Account Manager will return info to Setting Screen.
8. Alert user password has been changed
E8.1 if Account.id = -1 then alert old password incorrect



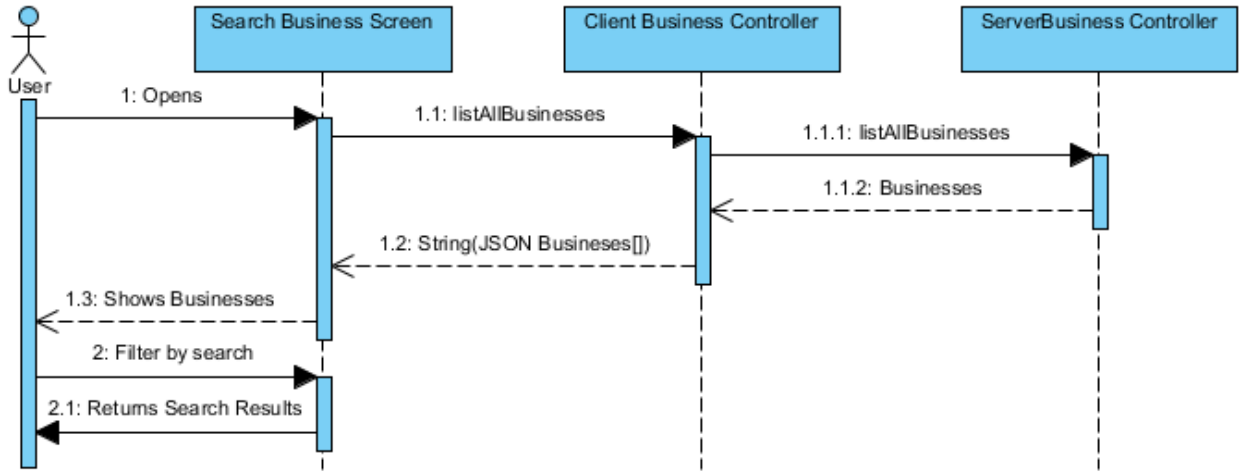
Output: *Login information is verified and the application is displayed to the user.*

Use Case: UC 2*-Login/Logout*
Trigger: *User start the application*
Precondition: *User must have signed up for an account.*
Interested Stakeholder: *homeowner and merchant*
Actor: *homeowner and merchant*
1. The application will prompt the user to a login screen to enter their email and password.
2. The user will then hit login, which will then verify if their email and password exists in the database.
E2.1 Name and password combination do not exist
E2.1 The error message will be displayed to the user that login information is incorrect.
Output: *Login information is verified and the application is displayed to the user.*



Use case: UC 3*-Create loyalty card*
Trigger: *Merchant hit the create card button*
Precondition: *Must be login*
Interested Stakeholder: *Merchant*
Actor: *Merchant*
1. The application will prompt the merchant to a create card screen where they have to enter their new card information.
2. The merchant will then hit create, the system will then verify if their new card information exists in the database.
E3.1 Loyalty card already existed in the database.
E3.1 The error message will be displayed to the merchant that this card is already existed.
Output: *New loyalty card is created/edited into system.*

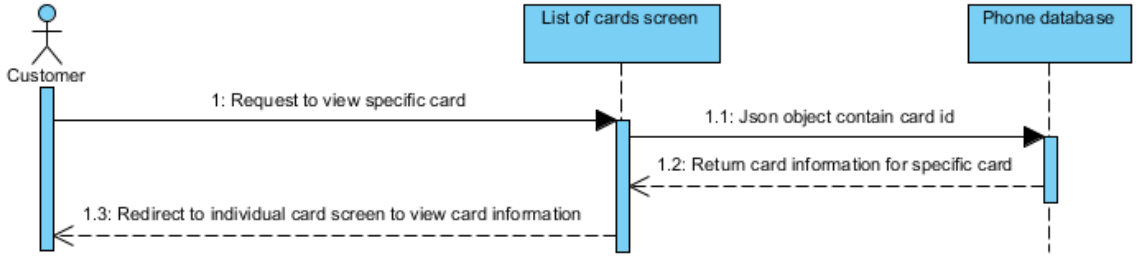
Use Case: UC 7-*Search Businesses*
Trigger: *Customer enters search criteria*
Precondition: *None*
Interested Stakeholder: *Customer*
Actor: *Customer*
1. The customer specifies search criteria
2. Businesses matching search criteria are displayed
Output: *List of businesses*



Use Case: UC 8-*view individual card*
Trigger: *Customer select specific card from a list of cards*
Precondition: *Must be logged in.*
Interested Stakeholder: *Customer*
Actor: *Customer*

1. *The customer selects a card from a list of cards (it can be a completed or current card)*
2. *Information about the card is displayed*

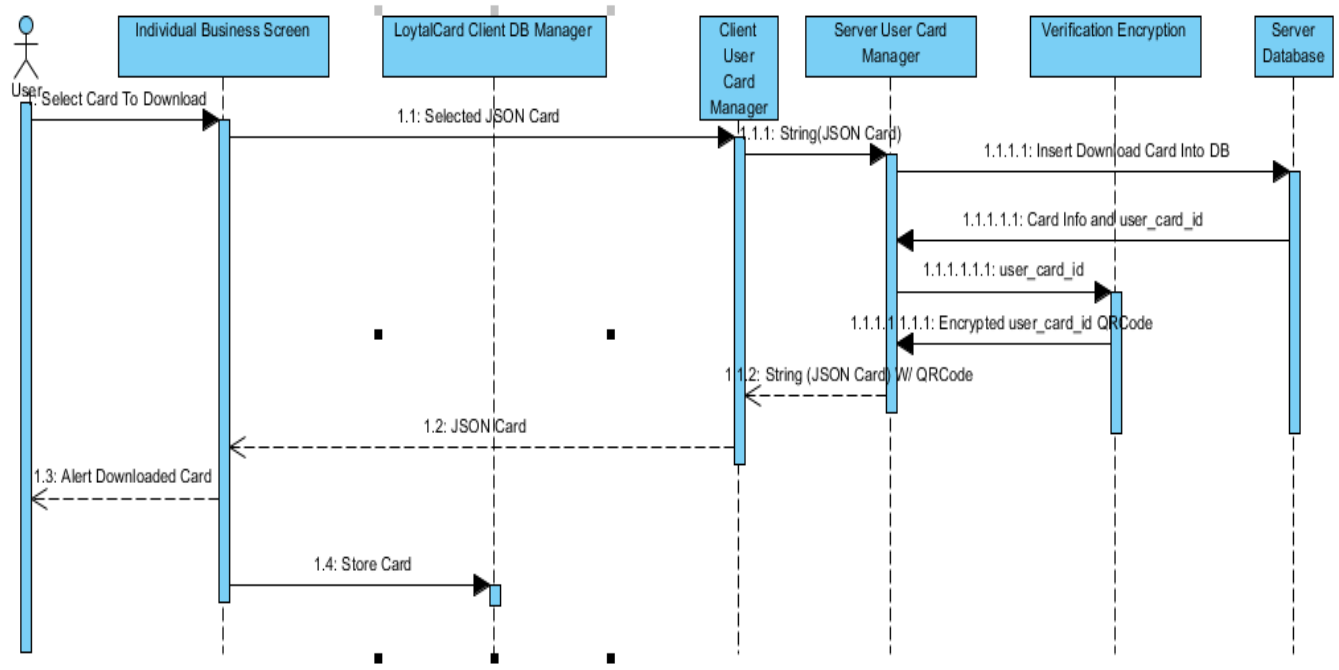
Output: *Card information*



Use Case: UC 9-*Download Card*
Trigger: *Customer presses a button to download card*
Precondition: *Must be logged in, have a card selected*
Interested Stakeholder: *Card*
Actor: *Card*

1. *The customer presses a 'download card' button*
2. *Card information is downloaded from server and saved on the phone*

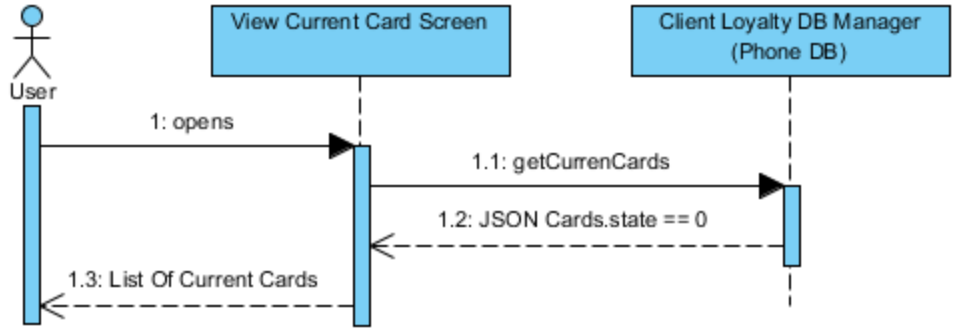
Output: *Card information*



Use Case: UC -*List Current Cards*
Trigger: *Customer views currently downloaded cards*
Precondition: *Must be logged in*
Interested Stakeholder: *Customer*
Actor: *Client Database*

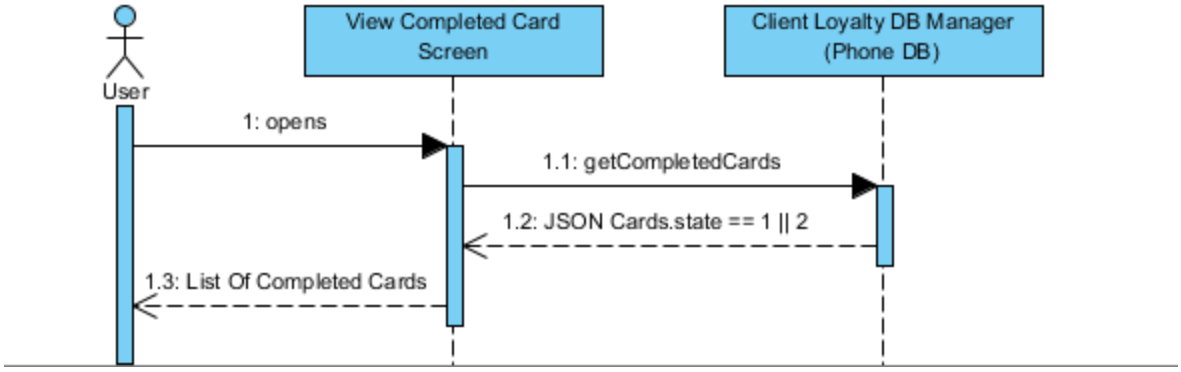
1. *The customer visits view current card screen*
2. *Database finds cards with state = 0*

Output: *List of current Cards*

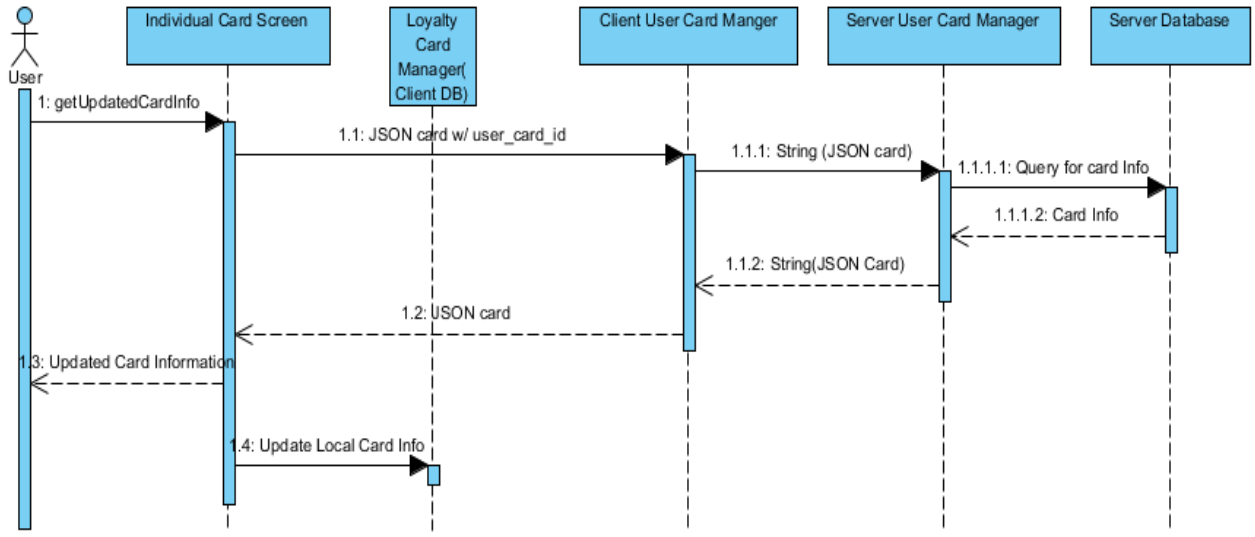


Use Case: UC -*List Completed Cards*
Trigger: *Customer views completed downloaded cards*
Precondition: *Must be logged in*
Interested Stakeholder: *Customer*
Actor: *Client Database*
1. *The customer visits view completed card screen*
2. *Database finds cards with state = 1 || 2*

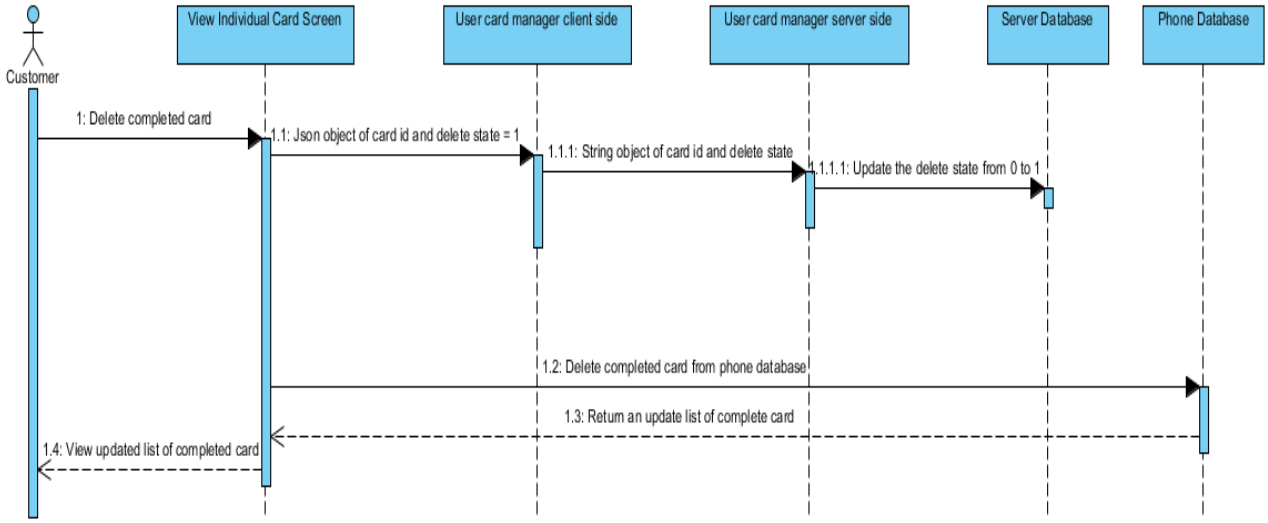
Output: *List of completedCards*



Use case : UC - *Get Updated Card Info*
Trigger : *User presses get updated card info*
Precondition : *Must have downloaded card*
Actors : *User, local database, server database, client user card manager, server user card manager, Individual Card Screen*
1. Individual Card Screen will wrap up current selected card in JSON.
2. Client User Card Manager will convert JSON to string and AJAX to Server User Card manager
3. Server User Card Manager will query server data base for any user cards matching user card id.
4.Database will return card with matches to Server User Card Manager
5. User Card Manager will Stringify JSON card back to client side.
6.Client User Card Manager will convert back to JSON and pass to Individual card screen.
7. Card screen will update info with new card info and then pass info to Loyalty Card Manager (client database).
8. Loyaltly Card Manager (client database) will update selected card with new information.
Outcome : *User's punch has been verified,card is updated with a new punch, and card is updated in server side database if there is internet.*

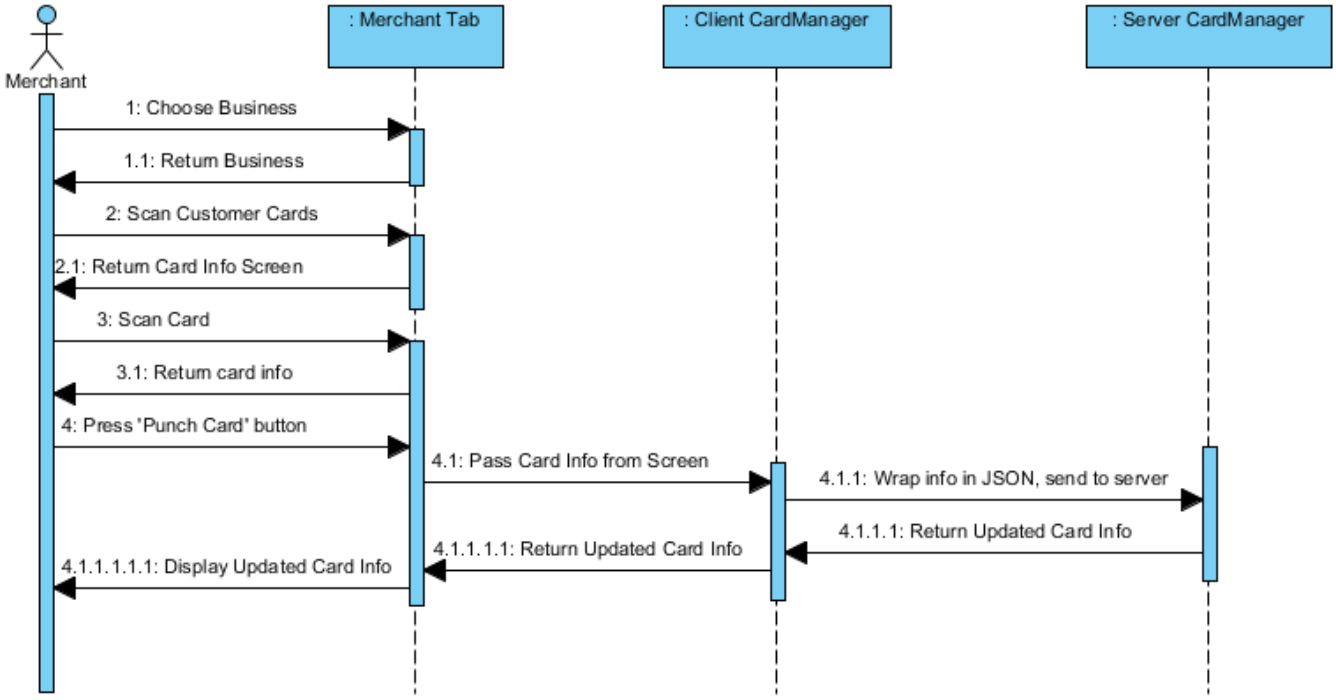


Use case : UC Delete completed card
Trigger : customer clicked on delete button
Precondition: Card must be a redeemed or expired cards
Actor: Customer
1. *Specific card will be deleted from the phone database*
2. Specific card will get updated in the server database to notify that the card is deleted
Outcome: Phone and server database is updated and redirect user back to the list of cards screens.

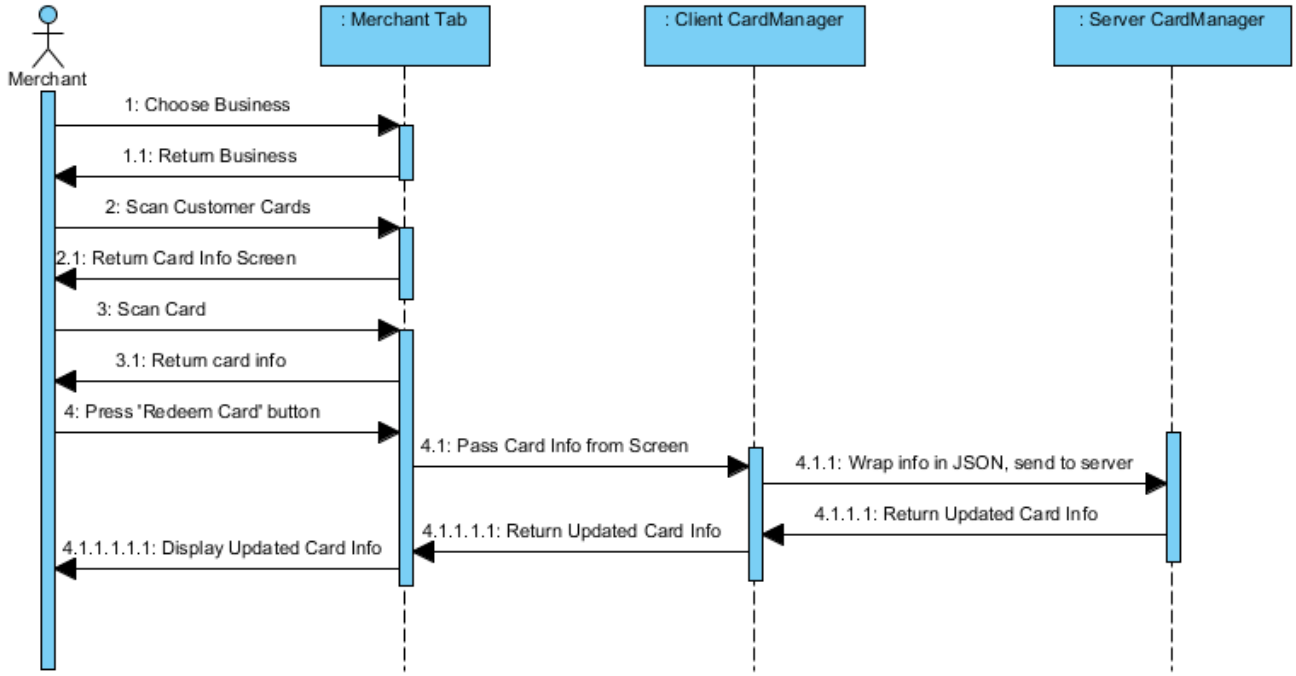


Use case : UC 10-*Redeem punch*
Trigger : *User punches card*
Precondition : *Card must not be completed or expired.*
Actors : *User, local database, server database.*
1. *The hash value of the qr code is compared with the card's local hash value.*

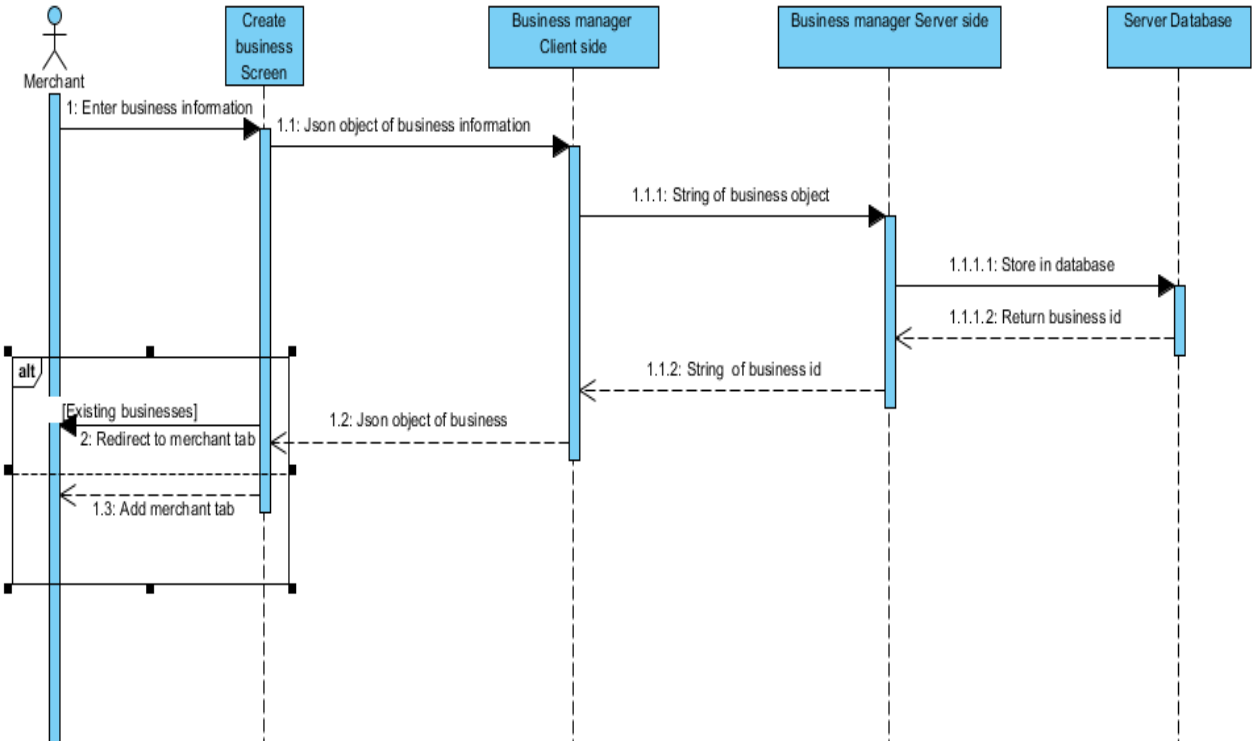
E1 *Hash value does not match, card is not updated with punch.*
E1 *User is notified qr code is not valid.*
3. *Card's number of punches is incremented .*
A1 *If card's number of punches matches the max punches assigned to card, card is changed to a completed state.*
4. *Update local database.*
5. *Update server data base if internet is available.*
Outcome : *User's punch has been verified,card is updated with a new punch, and card is updated in server side database if there is internet.*



Use case : *UC-11 Redeem reward*
Trigger : *User collects reward*
Precondition : *Card must be in a completed state and not expired*
Actors : *User, local database, server database.*
1. *The hash value of the qr code is compared with the card's local hash value.*
E1 *Hash value does not match.*
E1 *User is notified qr code is not valid and no reward is given.*
2. *Qr Code is verified and card is transitioned to a completed state.*
3. *Update local database.*
4. *Update server data base if internet is available.*
Outcome : *User has redeemed reward, information is updated locally and on the server.*



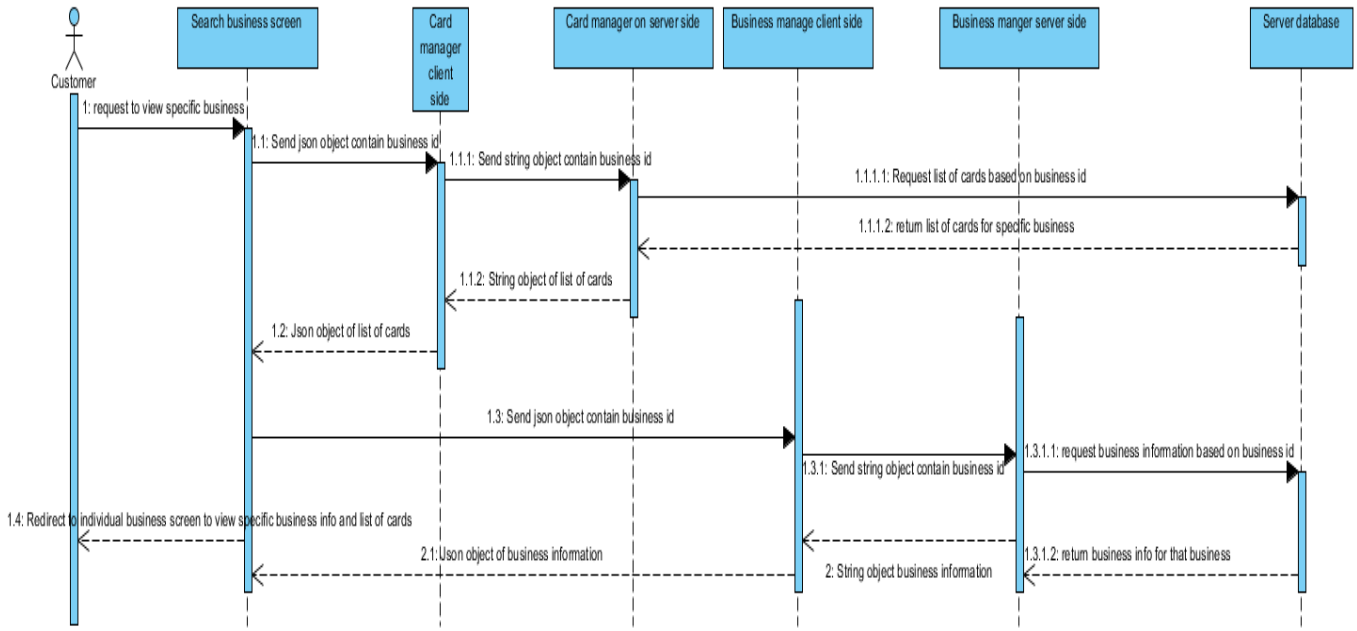
Use case: UC- Create business
Trigger: User click on create business in setting page
Precondition: login
Actor: Merchant
1. Merchant enter their business information
2. Store business information in the database
3. Check to see if merchant already have existing businesses or not
Alt 1. Merchant already have existing business in the database then redirect the to the merchant tab.
Alt 2. This is the merchant first business then create a merchant tab for the merchant.
Outcome: User can manage their business using merchant tab



Use case : UC View individual business
Trigger: User select a specific business from the search businesses screen
precondition:
Actors: Customer
1. Request for specific business information and list of cards using business id from the server database
2. Server database return business information and list of cards if business id is valid
ER: if business id is not valid

1. Send back an error message

Outcome: Customer can view specific business information and list of cards for that business on individual business screen



Use case : *UC-12 Store card*
Trigger : *Card information has been updated*
Precondition :
Actors : *Server database.*
1. *Take new card information and updated it in the server database.*
Outcome : *Card information is updated in the server side database, allowing for merchants to see user history.*

Conclusion & Future Work

Our team has developed a functional loyalty card application that has enough features to capture the attention of any small business. Security and performance were our two main concerns when designing and implementing this system. For the security aspect, we wanted to address verifying a user has earned an authentic punch. We had considered using GPS, handing the phone to a cashier to scan a code, and generating codes the user would enter to redeem a punch. Ultimately, putting a QR code on the customer's phone for the merchant to scan turned out to be the best solution for several reasons. This is because with this solution, the phone never has to leave the customer's hand, we rely on a merchant's WiFi connection instead of a 3G network, and the customer cannot forge punches to get a free reward.

This application will eventually become part of a larger system, so future work includes integrating this system with the application being developed by iapps24. As the system grows, being able to search for businesses by location will be desirable. The logic of which businesses to display on the homepage may also need some revision for advertising and monetization purposes. Upgrading to a business account will need some sort of verification that fits the business model for iapps24.

Self Assessment

Specifications	Implementation	Rating
The system shall allow the user to login and logout.	Our application has a login system to verify the user by using a email and a password.	100% complete
The system shall allow the user to create an account	If the user doesn't have an account, they can use the login screen to register for an account by using an email and a password.	100% complete
The system shall allow merchants to create loyalty cards	The application will let the merchant create loyalty card by entering the card information such the card title, reward description, expiration data and etc.... After that the card template will be store in the	100% complete

	database.	
The system shall allow merchants to update loyalty cards	The application will let the merchant update all of their existing loyalty cards that got store in the database.	100% complete
The system shall allow merchants give punches or redeem reward.	The application will let the merchant give customer a punch or redeem reward by scanning the QR code on the customer card.	100% complete
The system shall allow customers to search for local businesses	Users can find merchants participating in our loyalty program by searching for them by name using our search page.	100% complete
The system shall allow customers to select loyalty cards	Once a user has found a card, they can download that card and it will sit in their phone. They can use it by selecting that card in our application	100% complete
The system shall allow customers to download loyalty cards	The application will let customer download card template. Once the customer clicked on the download button then it will store that card template into the customer phone database and the user cards table in the server database.	100% complete
The system shall allow customers to redeem punches or rewards	The application will allow the customer redeem the reward by letting the merchant scan their card QR code to redeem and once the card is redeemed then the card state will be update to complete in the phone and server database.	100% complete
The system shall allow the database to store a history of completed punch cards	The application will store the user cards as completed cards once the customer redeemed their reward.	100% complete
The system shall allow the customer to view personal card history for completed rewards	The application will let the customer view all of their completed card which provided by the information that stored in the phone database.	100% complete
The system shall be developed using the	All of the user interface and functionalities will be program using	100% complete

Appcelerator SDK	Appcelerator SDK for Android and iOS.	
The system shall use an e SQLite database for local storage to prevent other applications from interacting with a user's card data	By default android does not allow any application to access another's. We the phones sqlite database to store meta data about the card. Such as the qrcode representing the card. This allows for offline capability. The user can still get their card punched, even though they do not have internet.	100% completewd
The system shall utilize encryption/decryption for punch a reward verification	Each user has a unique card, once they download it. When a user downloads a card, that card contains a qrcode along with it. This qrcode contains the encrypted unique string that represents that card. Only the correct merchant should be able to decrypt and interact with this card.	100 % complete