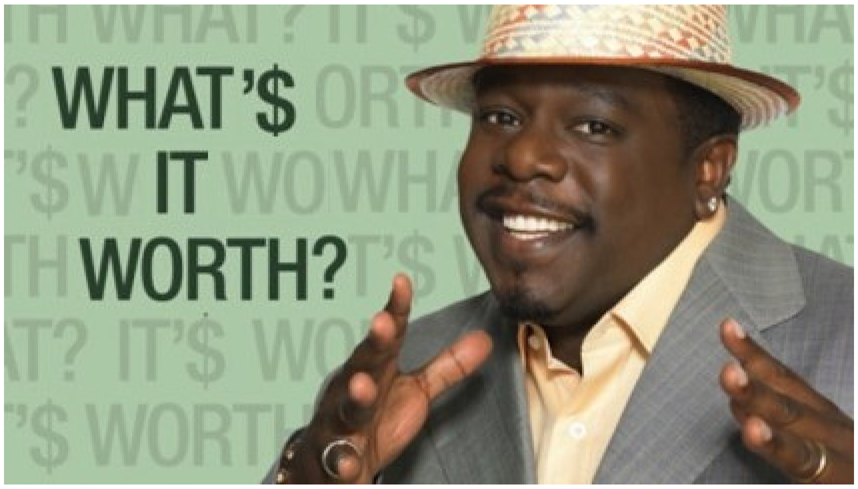


What’$ It Worth?

Theo Reinke, Nick Bachman, Kaileen Kraemer, Ben Pietrzak, Niraj Venkat, Sam Dickson

Purpose of the Application

Vendors of collectables do not have an easy way to update prices on the commodities that they are selling. For example, there are currently thirteen thousand *Magic: The Gathering* trading cards which all have potentially volatile values. It is currently impossible to keep up with the changing value of all these cards without some form of electronic assistance. While card buyers and sellers currently check todays market prices online by manually typing card names and sets into card pricing websites, this method of input is too time consuming for high frequency traders such as card shop owners or serious collectors. There should be an easier and quicker way to check the monetary values of collectables like trading cards.

Collectors of *Magic: The Gathering* trading cards, whether they be shopkeepers or an individual trader, commonly keep binders filled with pages of trading cards. When searching for the current price of a card, collectors have to keep in mind the full name of the card, the date it was printed, and other factors when determining its value. We want to create a way for collectors to simply open their binder, and use their mobile device's camera to detect a card in their binder and display its current monetary value.

Purpose of the Document

The purpose of this design document is to provide key details of the *What's It Worth?* application. It will summarize its functional requirements, general priorities, outline its design, and discuss design issues.

Summary of Functional Requirements:

- User will see a splash screen with the application's title upon opening the application.
- User will be able to access device's camera through the application.
- User will be able to capture photographs through the application by tapping any portion of the screen.
- User will be able to click a green "send" button that will submit the image.
- User will be able to see that their photo is being processed after submission.

- User will be able to see feedback in the form of an informational overlay on the screen.
- User will be able to see the item's current online price, if the item is recognized.
- User will be able to see the item's name, if the item in their sent image is recognized.
- User will be able to see an error message if their image was not recognized.
- User will be able to see an error message if the client is unable to connect to the server after multiple attempts.

- User will receive the suggestion to retake the photo of the item if their previous image was not recognized.
- User will be able to select a button on the main activity screen that will display a list of the names and prices of items they have submitted in the past.
- User will be able to clear their history of identified items.
- User will be able to remove single elements from their history of identified items.
- User will be able to observe a total value of items in history.
- User will be able to observe each item's value next to said item's name in history.
- User will be able to long press an item in the history to view detailed about that particular item such as:
    - Date and Time item was queried.
    - Detailed trading card game rules information pertaining to item including:
        - Format legality
        - Rule clarification
        - Set Number
        - Artist's Name
        - Expansion
        - Official Wizards of the Coast Community Rating
- User will be able to select a button on the main activity screen allowing them to search for specific items by the IME (Input Method Editor).

General Priorities

Usability

The user interface must be simplistic and intuitive. It must allow the user to have an adequate amount of space on the screen to take a photo. Once the capture of the photo is completed and submitted to the server, the user should see a simple but prominent display of the application's feedback on the current state of the image processing in the topmost section of the screen and still have adequate space to capture another photo. Upon completion of the image identification, the user will be able to easily navigate returned data.

Accessibility

The application will be developed with the intent of high ease of access. A splash screen with the application's title will appear upon opening the application and will then go straight to the main activity--main feed of the device's camera and minor menu items with verbose labels. Any user with basic knowledge of how a smartphone works will be able to find the application in the phone's "App Drawer" and open it. Once opened, the user simply points the camera at the card they wish to identify and taps the screen to capture an image.

Reliability

The system's reliability focuses on both front-end and back-end design. The front-end application should not crash under the condition that one photo is being captured at a time. On the back-end, the item's name and price information should be current and accurate.

Due to the field of image recognition being in its infancy, the application cannot guarantee the image captured by the device will be recognized. The system will be implemented with the optimal algorithms best suited for the aforementioned objects.

Performance

A client-server system is used to increase performance and streamline the image identification process. The front-end application will be connected to and transmit a captured image directly to the server back-end. The server back-end will perform all image processing and identification. The front-end device's hardware will not slow the application's performance significantly due to the bulk of the computational work being performed by the server.

Response Time

System response time should be no more than five seconds after the user's photo has been submitted to the server. Overall response time will depend on the strength of the user's data connection. The user should be able to see that their image is being processed through visual feedback from the application. If the image recognition is taking extraneous time, the front-end application will let the user know through display text. The image captured by the front-end of the system will be compressed to decrease network overhead.

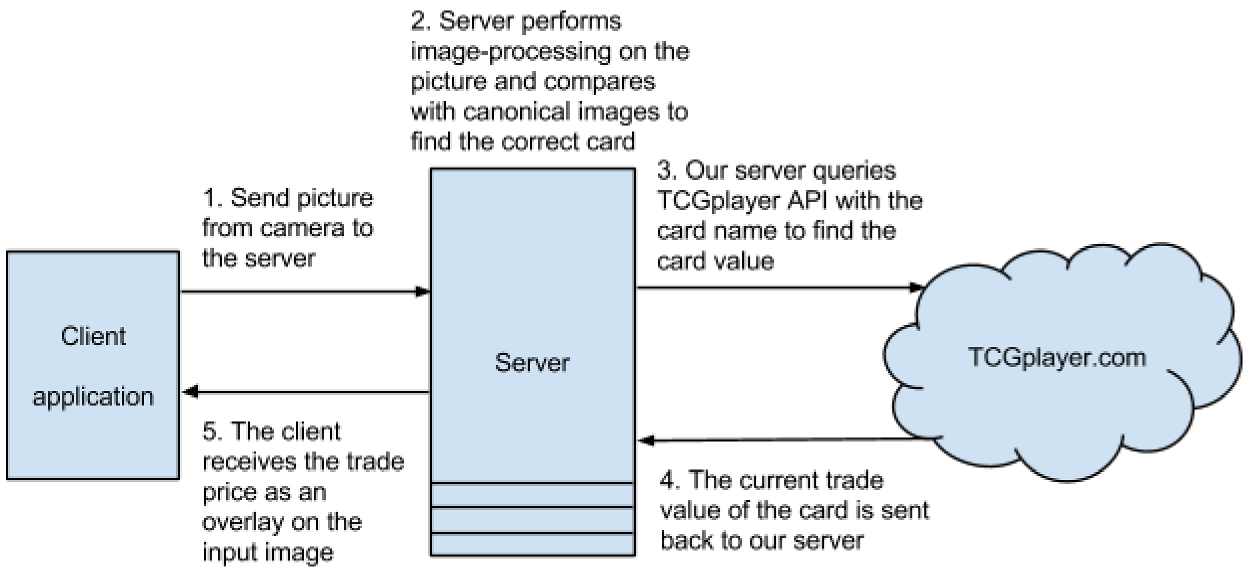
Battery Life

The battery life that the application consumes must be minimal. The Android device used to run the application is designed to be used to run many other applications and utilities, and therefore, excess battery life used by the system would harm the user’s experience. The application will have negligible overhead when not in focus.

Portability

The application will be developed on the Android platform, and will work on any phone running Android 2.3 through 4.3. The user will be able to open the application and take photos from anywhere, and will be able to send them anywhere with internet access. The application will be readily available on multiple app. marketplaces.

Outline of the design:


2. Server performs image-processing on the picture and compares with canonical images to find the correct card
1. Send picture from camera to the server
3. Our server queries TCGplayer API with the card name to find the card value
Client application
Server
TCGplayer.com
5. The client receives the trade price as an overlay on the input image
4. The current trade value of the card is sent back to our server

1. The client is responsible for taking the initial photo. The client will compress the photograph. The client will send the compressed source image to the server. In order to maintain an adequate response time from the server back to the client without compromising performance due to the client’s hardware, the client will not perform image preprocessing to reduce server computation.

2. On the server, edge detection algorithms will be used to determine what parts of the input image could be a card. Since each image will need to be compared against thirteen thousand cards, broad phase filtering will be performed by determining the dominant color of the cards border. If the card’s dominant color can be successfully identified, at least 84.57% of the database will be ruled out using this heuristic. If such color identification fails, comparing input image against the entire database will be resorted to. Object detection algorithms such as SURF will be used to detect areas of interest (hessian blobs) in the card region. Correlations between those areas of interest and our cached collection of blobs (generated by running blob generator on catalog

of stock card images) can be detected to determine the probability of a match. If a high-level of confidence in a match is reached, server will report back to the application the name of the card as well as other relevant information about the card. If the server cannot identify the name of the card with a high level of certainty, it will report the failure to the client, which will then prompt the user to enter the information manually.

3. From the server, a popular external price database, TCGPlayer, will be queried to make sure that the internal database prices of *Magic: The Gathering* cards are as accurate as possible. The most accurate way to determine the value of an object is to determine the lowest price that any retailer is willing to sell that object for. Since TCGPlayer is the largest aggregator of online collectible card stores, lowest price among all of its retailers can be easily determined. Most collectors do not know the value of their cards so they use sites like TCGPlayer to determine that value. Therefore, it would make very little sense to allow collectors to influence the pricing data that we retrieve for them. Therefore, TCGPlayer will be our master lookup table for prices.

4. TCGPlayer will respond to a query with high, median, low prices and the eight lowest priced retailers. A URL corresponding to a graph image of price data over time will also be returned. The server will perform basic market analysis on queried cards. The server will also monitor *shady* vendors whom mask shipping costs to achieve a price that appears more competitive. These vendors will be removed from pricing consideration.

5. The server sends the relevant information to the client. The card name determined by the server will be superimposed over the physical card’s title. The client then displays the high, median, and low prices to the user in a graphical overlay. Furthermore, if a price graph is determined, the graph will also be displayed.

Major design issues:

1) Programming Languages:

Option 1: Java
Option 2: C++
Option 3: Python
Option 4: Objective-C

Option used: Java
Reason:
We decided to use Java for our front-end application because we decided to go with the Android platform. We are not familiar with iOS development or Objective-C syntax. Java is the language we are collectively most familiar with, has good network protocols, and supports multiple platforms. The back-end server will also be written in Java because future iterations may move all server functionality to the user’s personal device. The augmented reality feature requires that the identification latency is minimized.

2) Mobile Platform Used

Option 1: iOS
Option 2: Android
Option 3: Windows Phone
Option 4: Firefox OS
Option 5: Blackberry
Option 6: Palm OS
Option 7: Symbian
Option 8: WINDOWS CE 6.0

Option used: Android

Reason:
Android is a popular platform and supports developers. Android is less restrictive than iOS. The group is more familiar with the Android environment as opposed to the others listed above. Furthermore, Android is projected to hold 80% of worldwide smartphone sales in the third quarter of 2013, giving it a large advantage in terms of users over iOS and other mobile operating systems. Due to the majority of our service being performed on a server, the client application will be lightweight and should be easy to port to other systems in future iterations.

3) Architecture Used

Option 1: Client-Server
Option 2: Pipe-and-Filter

Option used: Client-Server
Reason:
The Client-Server architecture works best for our system. The client, which is the Android application, will contact the server. Image recognition requires heavy duty processing which the server better suited to perform. This will be a light client, heavy server architecture. In future iterations, the image identification process may be streamlined to the point of being performed entirely on the client negating the need for a server.

4) User Login Creation

Option 1: Users create a username and password in order to access the functionality of the application
Option 2: User has ability to create a username and use it to access the application
Option 3: User has ability to create a password and use it to access the application
Option 4: Users do not have the option to create user login information

Option Used: Users do not have the option to create user login information
Reason:
Users will be able to access the application quickly and easily, and won't have to spend time entering login information each time they use the app. Users will not be prompted to enter personal information, making user login creation unnecessary. Users' history of previous image identification will be cached locally on the device further negating the need for having a username/password security structure. Also, the information stored locally is believed to not be sensitive to warrant such security measures.

5) Front-End Application:

Option 1: Desktop application that accesses webcam

Option 2: Mobile device that accesses built-in camera
Option 3: A web front-end where users can upload images

Option Used: Mobile device that accesses built-in camera
Reason:
The mobile device is portable, convenient for the user, has an existing architecture, and reaches a large base of potential users. Many times the application will need to be accessed in locations where the user does not have access to a personal computer or it is not practical to do so. Other options are considered for future iterations to expand our user-base.

6) Database Management System:

Option 1: MySQL
Option 2: SQL Server
Option 3: Oracle
Option 4: Berkeley DB
Option 5: Microsoft SQL

Option Used: MySQL
Reason:
MySQL is easy to install and integrates well with Java. It is also free to use. MySQL has readily accessible documentation on the internet free of cost.

7) Displaying Card Price:

Option 1: Displaying the lowest, medium, and highest current selling price of the card in a grid above the card's image
Option 2: Displaying the card's price information over the card's image
Option 3: Displaying the card's price information below the card's image
Option 4: Displaying the card's price information in an entirely different spot on the screen, and outside of a color-coded grid

Option used: Displaying the lowest, medium, and highest current selling price of the card in a grid above the card's image
Reason:
The price information is best displayed above the recognized card's image, a color-coded grid. The background colors of each section of the grid are best as follows: red for the lowest price, blue for medium price, green for highest price. Preceding each price in US dollars are the initials "L, M, H," to further clarify the listed monetary values for the user. Displaying three different prices gives the user a

better insight into what the card is really worth. If a graph of the cards price history is available, it will be displayed to help the user better predict future price trends.

8) Displaying Card Price Graph:

Option 1: Detailed price information is displayed in a price vs. time graph
Option 2: Price vs. time information is not displayed to the user at all

Option Used: Detailed price information is displayed in a price vs. time graph
Reason:
We decided to provide additional details of the card to the user to improve user experience. This graph will be displayed clearly and prominently under the card image. A price vs. time graph allows the user to get better insight into future price trends of the card.Option 2 may also be explored if we feel that the user interface has become cluttered and overwhelming. Also, this information could be summarized in future iterations as a simple *up* or *down* arrow indicating a general trend of the card’s price from the near past to the present.

9) Displaying Card Name:

Option 1: Displaying the card’s name over the card’s image within a black box
Option 2: Displaying the card’s name elsewhere within the activity screen

Option Used: Displaying the card’s name over the card’s image within a black box
Reason:
This method of displaying the card’s name is simple and clear to the user. In future iterations, the application may identify multiple cards from a single image captured by the device. It is important to correlate information such as the card’s name to the correct card in the image.

10) Displaying History

Option 1: User history comes up in standard list form through a swipe-down gesture
Option 2: User history comes up from a separate menu
Option 3: User history is not able to be displayed at all
Option 4: User history can be accessed through a button on application main-screen

Option Used: User history can be accessed through a button on application main-screen
Reason:
The display of user history will be button-based, and the user will easily be able to view the last cards they scanned in a list. This functionality does the user a great service, as they may not want to re-scan a

card. This will save the user time and free up server resources for identification of other cards.

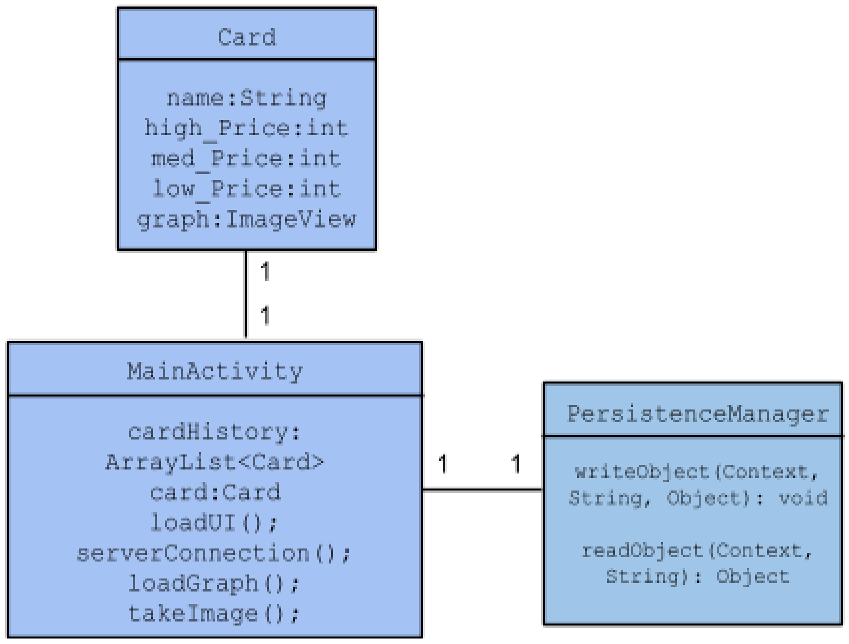
11) Augmented Reality Component

Option 1: Card information is displayed in a separate window
Option 2: Card information is displayed as an overlay on top of the camera-captured image
Option 3: Card information is displayed as an overlay on top of live camera feed.

Option Used: Card information is displayed as an overlay on top of the camera-captured image
Reason: An overlay on top of the camera captured image is more aesthetically appealing to the user. When multiple cards are being detected, an overlay will help show the difference between cards. In future iterations when image processing moved entirely to the client device, a real time overlay of the card information may be possible.

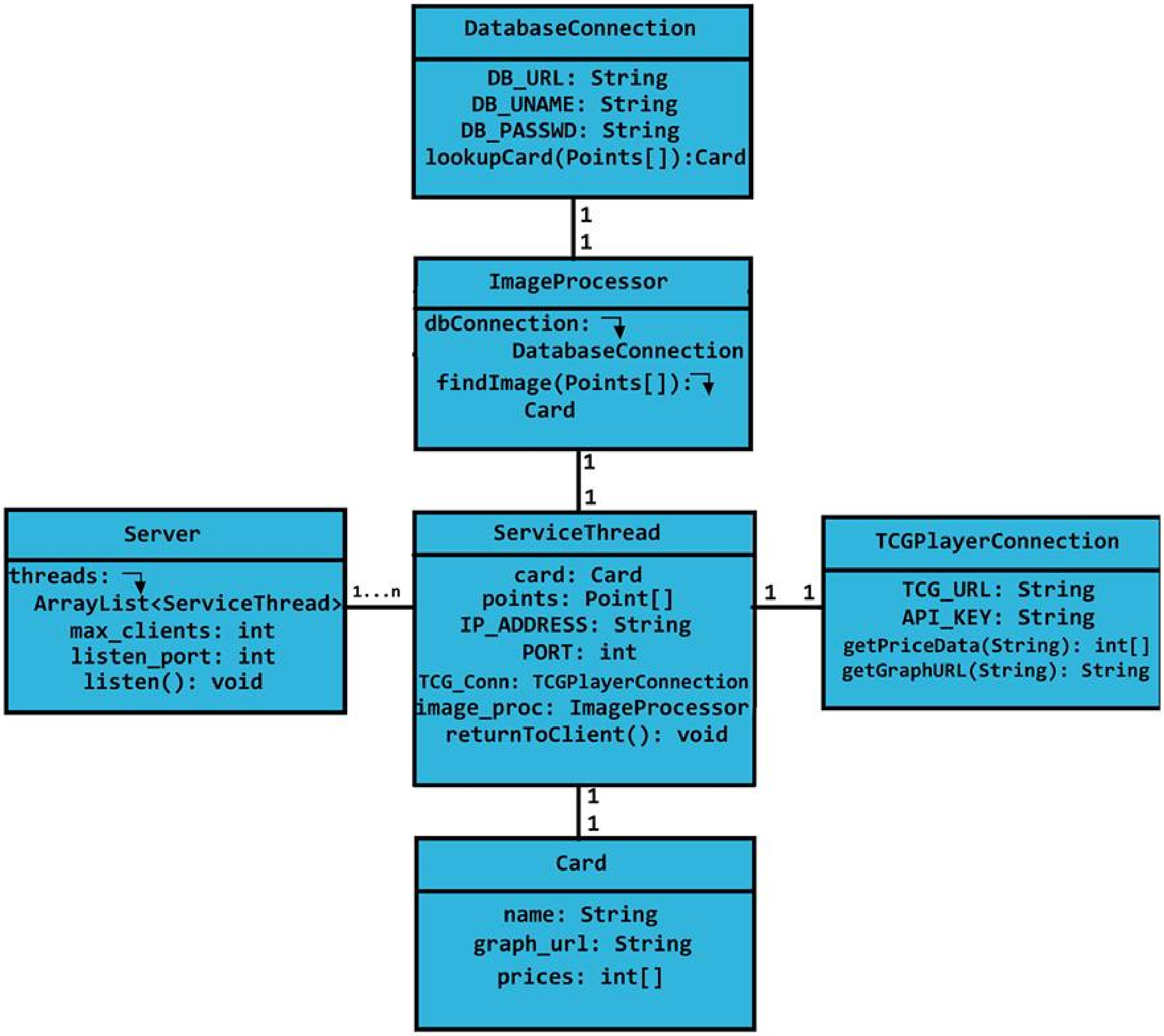
Outline of the Design:

Class Diagram (Client):



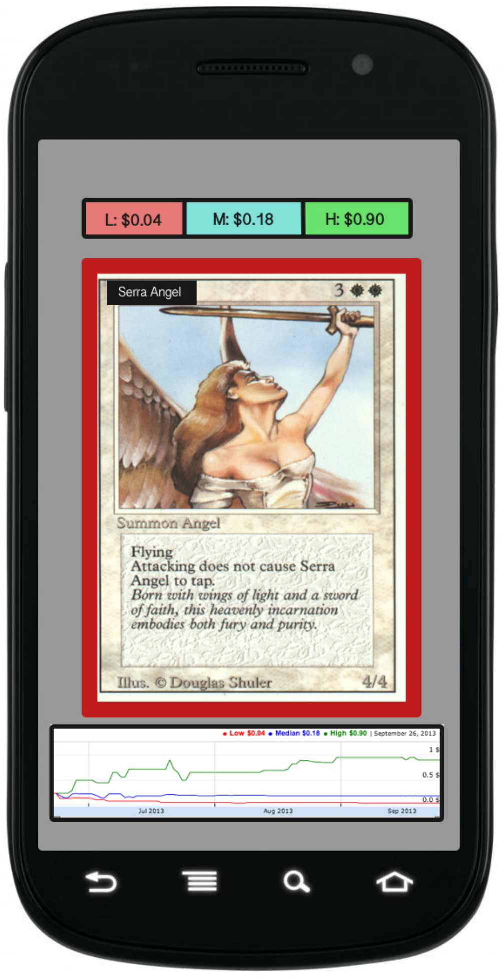
The MainActivity class handles the main UI of the app, specifically the camera view and the card information view. MainActivity handles the capture from the camera and then sends it to the server. The Card class stores all the card information that is given back from the server, namely the high and low prices, the price graph and the card name. MainActivity stores the current Card and an ArrayList of Cards that the user has scanned in the current session and caches them using PersistenceManager.

Class Diagram (Server):



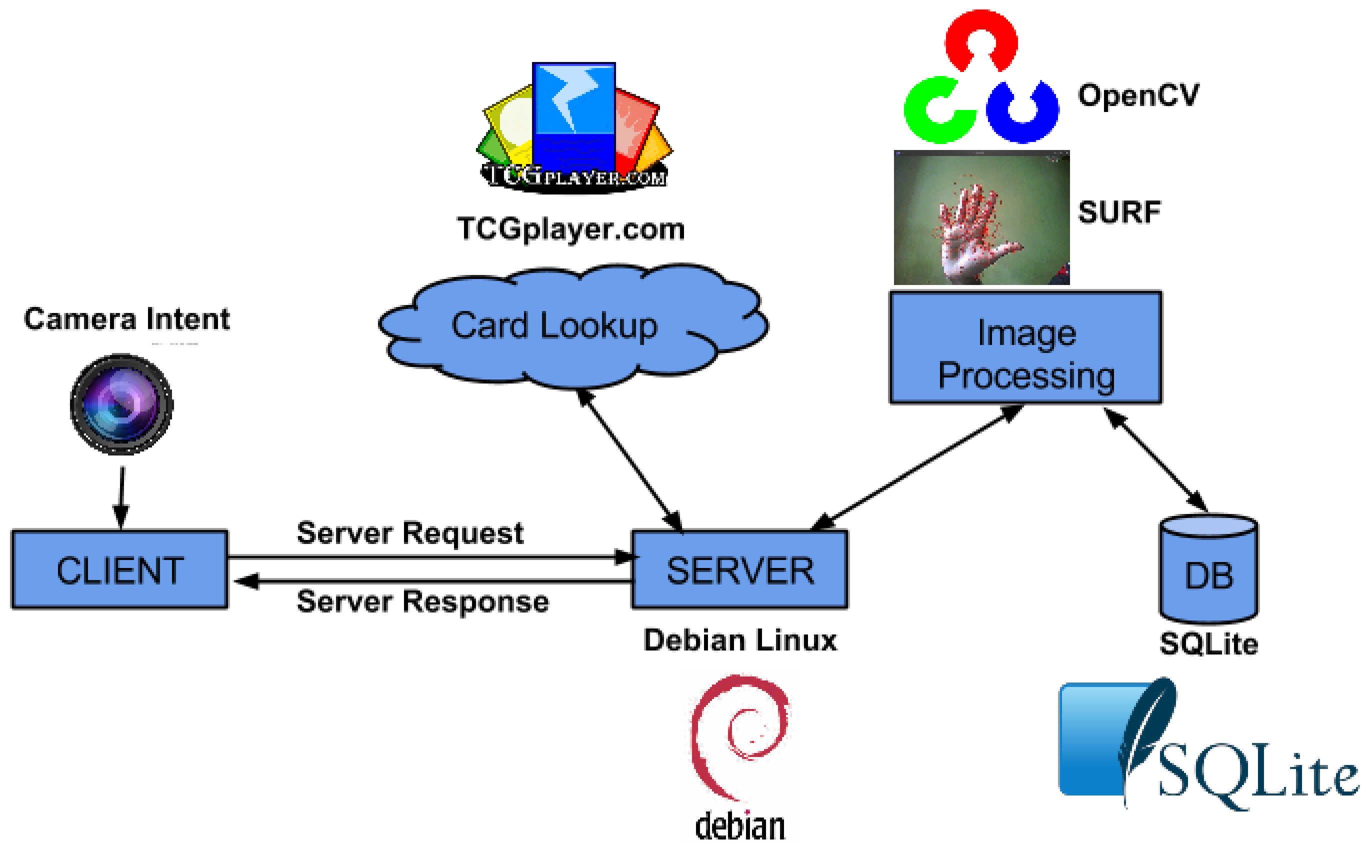
The server receives image data from the client and creates a new thread to handle the identification process. The card information is first sent to the ImageProcessor class and the SURF algorithm is applied to output unique identifying data. This unique data is used to locate the identified card in the MySQL Database, which returns the name of the card in a new Card object. Price and other identifying information is then retrieved from TCGPlayer via TCGPlayerConnection. The final card object is send back to the client via an ObjectOutputStream.

Main User Interface

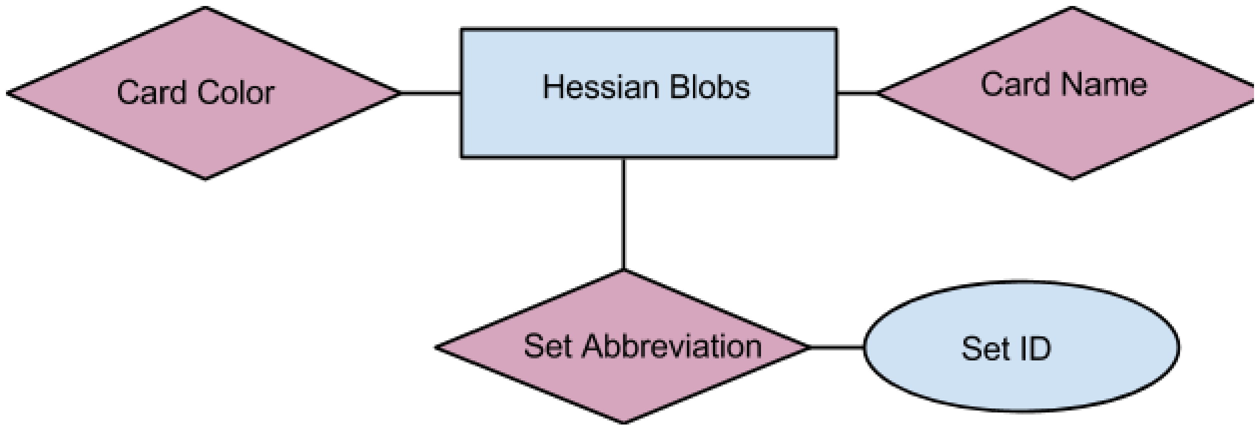

L: $0.04
M: $0.18
H: $0.90
Serra Angel
3
Summon Angel
Flying
Attacking does not cause Serra Angel to tap.
Born with wings of light and a sword of faith, this heavenly incarnation embodies both fury and purity.
Illus. © Douglas Shuler
4/4
• Low $0.04 • Median $0.18 • High $0.90 | September 26, 2013
1 $
0.5 $
0.0 $
Jul 2013
Aug 2013
Sep 2013

Prices are placed at the top, the card image is in the middle, and the graph of price/time is at the bottom. Red represents the low price, blue the median, and green the high price.

Component Diagram:



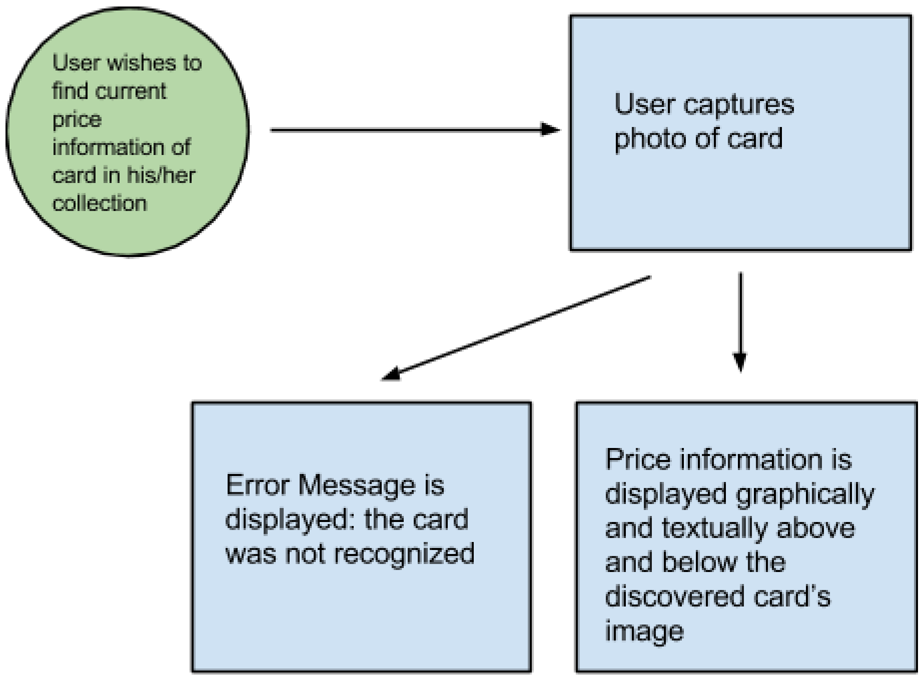
E-R Diagram of Card Lookup Database:



The main entity of the relationship is the Hessian Blobs, which has a card name, card color, and a set abbreviation linked to it. Attributed to the set abbreviation is the Set ID.

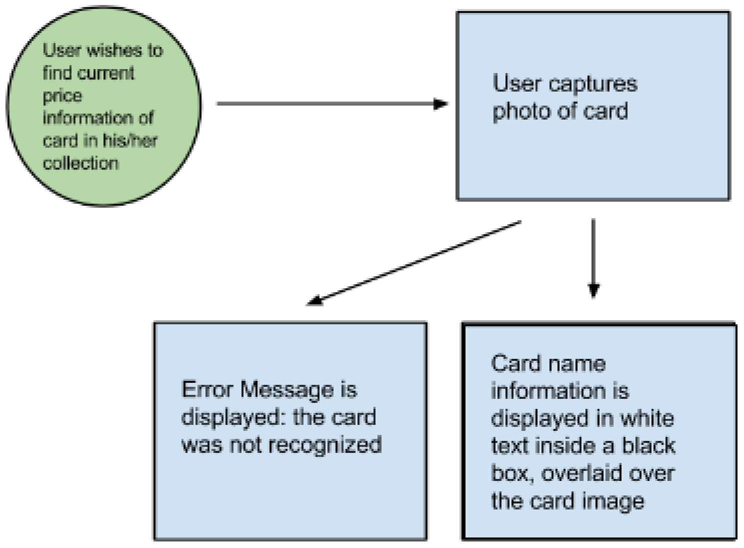
Activity Diagrams:

User wishes to discover a card's price


User wishes to find current price information of card in his/her collection
User captures photo of card
Error Message is displayed: the card was not recognized
Price information is displayed graphically and textually above and below the discovered card's image

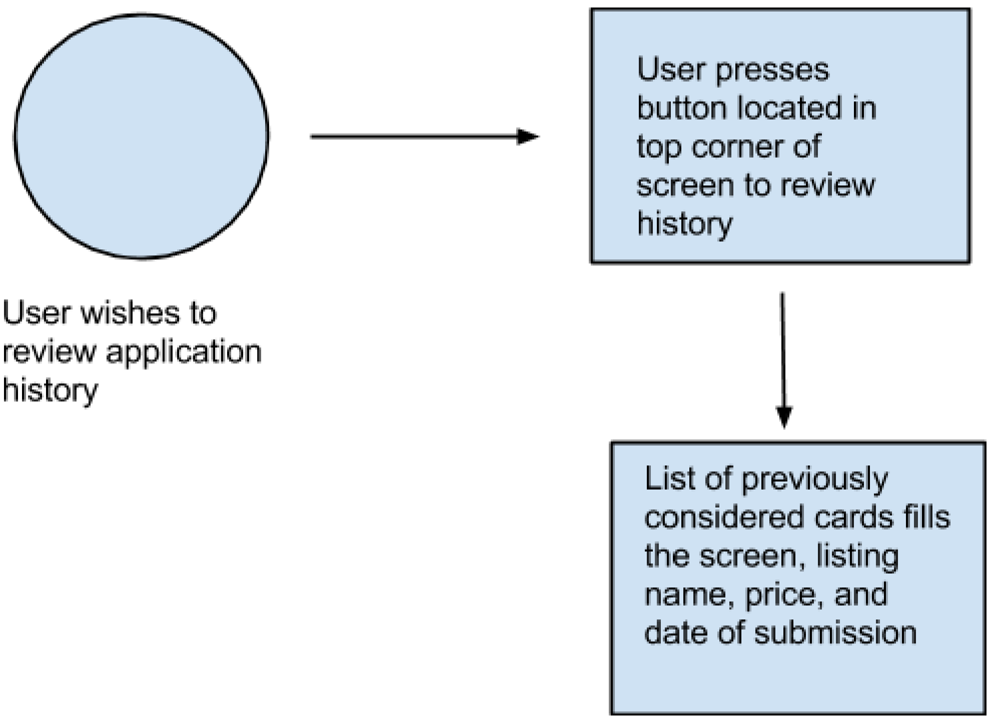
When the user wishes to know the price information of the card, they open the application and capture a photo of the card. If the card is successfully recognized, the price information will be displayed clearly to the user on the activity screen. If the card is not successfully recognized, the user will know after an error message appears on the activity screen.

User wishes to discover the card's name


User wishes to find current price information of card in his/her collection
User captures photo of card
Error Message is displayed: the card was not recognized
Card name information is displayed in white text inside a black box, overlaid over the card image

When the user wishes to know the name of the card, they open the application and capture a photo of the card. If the card is successfully recognized, the name will be displayed clearly in white text inside of a black box, hovering over the card. If the card is not successfully recognized, the user will know after an error message appears on the activity screen.

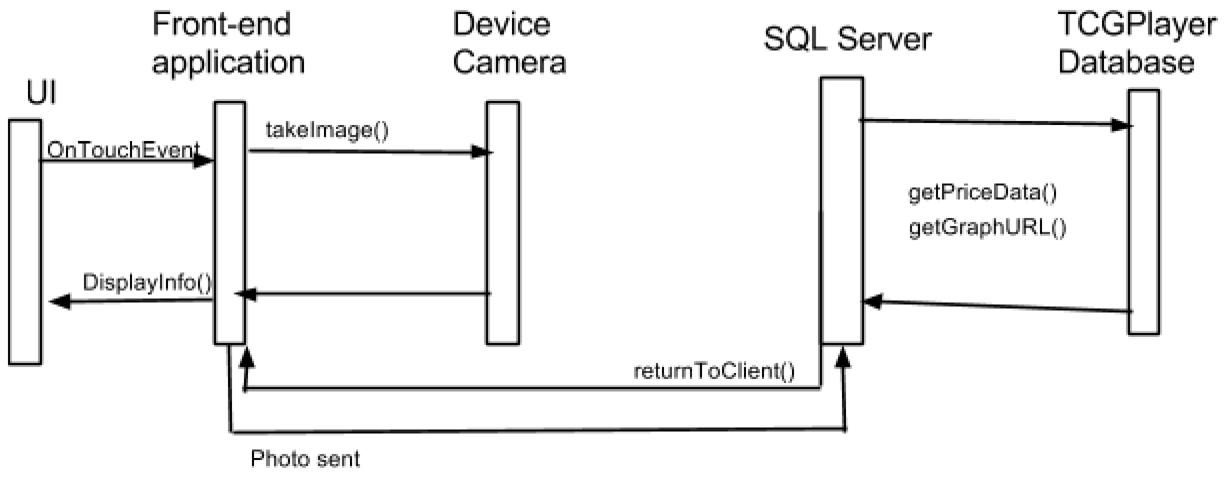
User wishes to view history of submitted cards


User presses button located in top corner of screen to review history
User wishes to review application history
List of previously considered cards fills the screen, listing name, price, and date of submission

When the user wishes to view the list of their submission history, they open the application and can press the history button located at the top right of the screen. The list will appear, and this functionality does not depend on on the amount of cards the user has submitted in the past: a list will appear in every case. This list will display the card name, price, and date of the query (submission).

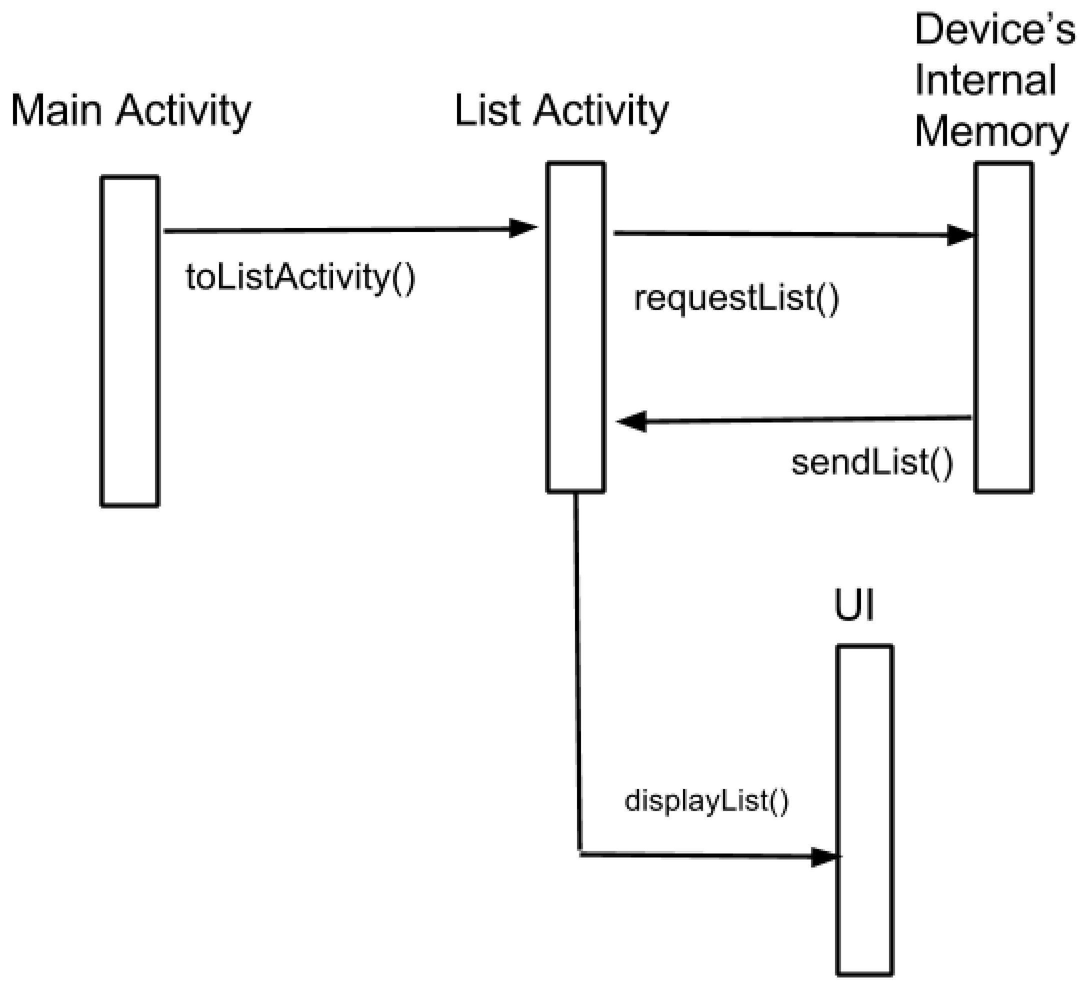
Sequence Diagrams

Displaying the Card's Information



The user interface allows for the front-end of the application to capture an image with a single tap on the screen. The device's camera captures the image then the application sends the image to the server. On the server, the image is processed and if a match is found, the database returns a card name. The card name is used to lookup pricing information from the external database which is then sent back to the server. The server forwards that information to the application which is then displayed by the user interface.

Retrieving the History



The main activity contains a button that will launch the list activity screen. Upon launch, the list activity will request the list of user history stored on the device, which contains the user’s submissions in reverse-chronological order (most recent at the top), from the server. The list will be sent back to the list activity and displayed in Android’s standard list format through the UI.