Introduction

A community of users is both a dynamic and an interesting concept to model within an application. However, communities often require a common goal or interest in order to sustain themselves. This application models how a community interacts both with and through a set of collectible trading cards. In particular, our project models a plausible set of American Football trading cards which display statistical information for given players. The structure of the National Football League (NFL) is also modeled in the background of the application so that relevant team, conference, and division information can be included where it is needed on the trading cards. The cards display both statistical information, and images of the players they are attempting to represent.

These cards are collectible in the sense that it is likely that not every user in the community will own a copy of each card in the set. Users are required to interact with each other in order to initiate trades and attain cards that they did not have previously. Thus, each user has a unique account (defined by a username and a password) which allows them to see the cards in their collection and take part in trades with other users. Users without an account cannot use the application, because a consistent online avatar is required to build a concrete community within the application.

In order to populate the pool of tradable cards within the community, players are able to purchase packs of cards within the application with points that they accumulate by logging into the application. The packs are initially randomized, but once a pack is available for purchase, the cards within it are static. In other words, if a user were to buy the same pack of cards twice, they would receive the same set of trading cards. However, users initially have no indication of what cards are in a given pack. The packs have variable costs, and users are unable to purchase packs if they do not have enough points.

Thus, this application acts as an interface for a trading card application and seeks to replicate the primary components of real-world trading card communities.

Requirements Document

User Stories

* As a user, I have a unique account that I can log into.
* As a user, I can have a unique collection of cards that are consistent across logins.
* As a user, I can see my unique collection of cards within the application and interact with my cards in order to view the information they present (flipping the cards).
* As a user, I can initiate trades with another user by providing the cards I am willing to trade along with the cards that I desire from that user.
* As a user, I can accept trades that have been initiated with me by providing the cards I am willing to trade with the initiating user.
* As a user, I can confirm trades that I have initiated and have subsequently been accepted.
* As a designer, I want the application to perform the trade only after a trade has been initiated, accepted, and confirmed, in that order. Error handling should be in place if a user attempts to perform a trade without those ordered steps occurring.
* As a user, I can accumulate points by logging into my account within the application.
* As a user, I can use my points through an in-application “store” in order to buy new packs of cards.
* As a user, I want options for packs that I can buy with variable pricing options.
* As a user, I want the experience of “opening” a new pack of cards (cards should not be viewable before a pack is purchased).
* As a designer, I do not want the application to allow the user to buy a pack of cards without having the required amount of points.
* As a designer, I want the application to deduct the pack value from a user’s accumulated point total on a purchase.
* As a user, I want the interface to be attractive and easy to use.

Security Measures

Logins are our primary security vulnerability. In order to create a secure login system, passwords in the database are hashed. Cookies (used to check login) are also encrypted. In order to protect against sql injection, input escaping and prepared statements have been implemented within the component of the application that handles the querying.

Special Functionality

Exporting Data

We are exporting data through the database, which simply performs an SQL dump.

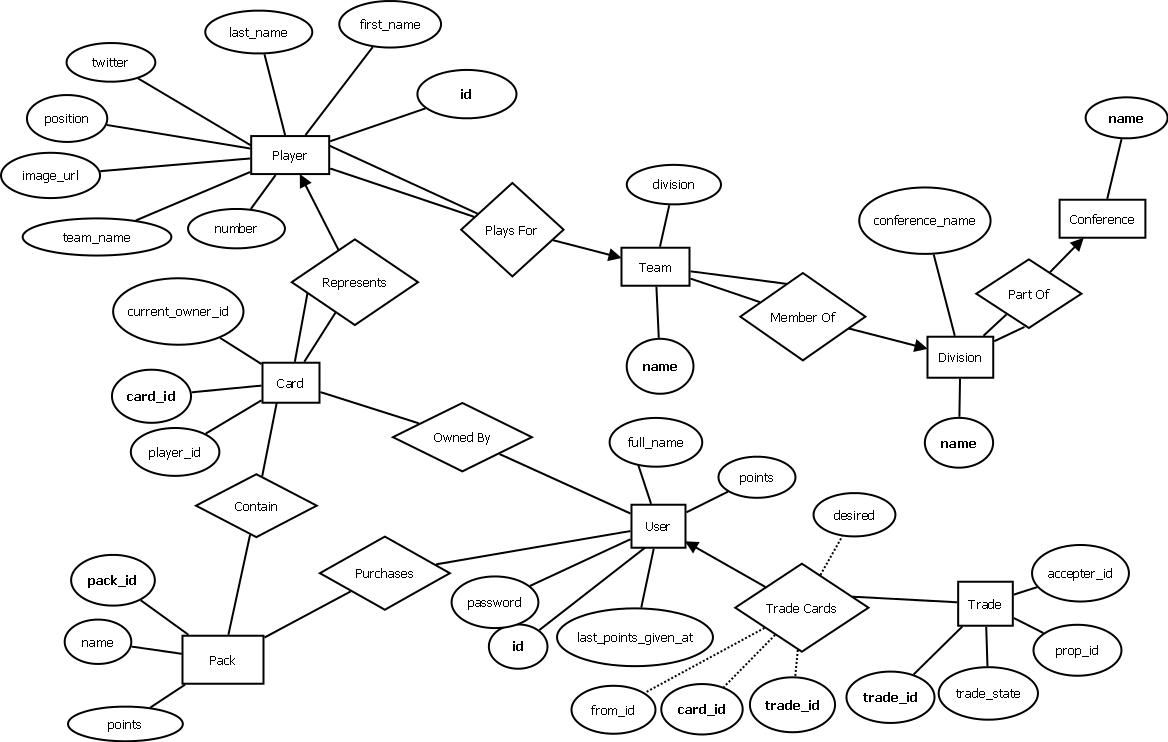
Design Process

There were three primary criteria that we were seeking to meet when we were deciding on our project topic. One, we wanted something that could be effectively modeled in the database. Two, we wanted to pick a topic that would have a great deal of associated data that we could pull from to populate our tables. Lastly, we wanted to pick something that would be interesting and fun to work on. As we were thinking, our initial thought was to do some sort of sports analysis application. Sports leagues lend themselves well to an E-R diagram because every component of the league, from overall structure down to individual players, exists in relationship with some other component of the league. American football was the logical choice because of its popularity and the subsequent statistical data that is available online. However, we decided to tweak our original project design a little bit as we got started because a simple sports analysis application was a bit boring and not very dynamic. We decided to take the statistics that we were going to originally just use for analysis and place them on trading cards. The underlying structure of our database did not change much (we only removed some of the more superfluous data tables and replaced them with tables to handle the structure required for the trading cards), but it was much more interesting to model.

We chose to code in Python primarily because it was the language that the whole group was comfortable with. There were also some very beneficial frameworks (Flask) that helped to handle more of the UI/routing side of the app. These allowed us to focus on the database components of the application rather than getting bogged down in designing the other building blocks we needed to get the app off the ground.

There were two primary concerns for app security that we faced as we were building the application. On one side, there are direct user input fields which exist within the app (specifically for login). These are an immediate red flag for SQL injection, so the queries are formatted to prevent malicious code from being injected into the queries. The other main problem we faced was having users access components of the app when they are not authorized to (i.e. accessing the interface without a login, confirming a trade without it having been initiated, etc.) These security concerns are handled programmatically within the code, whether it means querying the database to see where in the trade process a given trade is at before loading a page or checking a session variable to see if a user is logged in.

ER



Database Schemas

**card**

| **Column** | **Type** | **Null** | **Default** | **Comments** | **MIME** |
| --- | --- | --- | --- | --- | --- |
| card\_id | int(11) | No |  |  |  |
| player\_id | int(11) | No |  |  |  |
| current\_owner\_id | int(11) | No |  |  |  |

**Indexes**

| **Keyname** | **Type** | **Unique** | **Packed** | **Column** | **Cardinality** | **Collation** | **Null** | **Comment** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PRIMARY | BTREE | Yes | No | card\_id | 104 | A | No |  |

**conference**

| **Column** | **Type** | **Null** | **Default** | **Comments** | **MIME** |
| --- | --- | --- | --- | --- | --- |
| name | varchar(50) | No |  |  |  |
| year\_instated | int(4) | Yes | *NULL* |  |  |
| num\_divisions | int(11) | Yes | *NULL* |  |  |

**Indexes**

| **Keyname** | **Type** | **Unique** | **Packed** | **Column** | **Cardinality** | **Collation** | **Null** | **Comment** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PRIMARY | BTREE | Yes | No | name | 2 | A | No |  |

**division**

| **Column** | **Type** | **Null** | **Default** | **Comments** | **MIME** |
| --- | --- | --- | --- | --- | --- |
| name | varchar(50) | No |  |  |  |
| year\_instated | int(11) | Yes | *NULL* |  |  |
| num\_teams | int(11) | Yes | *NULL* |  |  |
| conference\_name | varchar(50) | Yes | *NULL* |  |  |

**Indexes**

| **Keyname** | **Type** | **Unique** | **Packed** | **Column** | **Cardinality** | **Collation** | **Null** | **Comment** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PRIMARY | BTREE | Yes | No | name | 8 | A | No |  |

**packs**

| **Column** | **Type** | **Null** | **Default** | **Comments** | **MIME** |
| --- | --- | --- | --- | --- | --- |
| pack\_id | int(11) | No |  |  |  |
| name | varchar(20) | No |  |  |  |
| points | int(11) | No | 1 |  |  |

**Indexes**

| **Keyname** | **Type** | **Unique** | **Packed** | **Column** | **Cardinality** | **Collation** | **Null** | **Comment** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PRIMARY | BTREE | Yes | No | pack\_id | 5 | A | No |  |

**packs\_players**

| **Column** | **Type** | **Null** | **Default** | **Comments** | **MIME** |
| --- | --- | --- | --- | --- | --- |
| pack\_id | int(11) | No |  |  |  |
| player\_id | int(11) | No |  |  |  |

**Indexes**

No index defined – both player\_id and pack\_id are foreign keys

**player**

| **Column** | **Type** | **Null** | **Default** | **Comments** | **MIME** |
| --- | --- | --- | --- | --- | --- |
| id | int(11) | No |  |  |  |
| first\_name | varchar(50) | No |  |  |  |
| last\_name | varchar(50) | No |  |  |  |
| number | int(11) | No |  |  |  |
| team\_name | varchar(50) | No |  |  |  |
| position | varchar(10) | No |  |  |  |
| image\_url | text | Yes | *NULL* |  |  |
| twitter | text | Yes | *NULL* |  |  |

**Indexes**

| **Keyname** | **Type** | **Unique** | **Packed** | **Column** | **Cardinality** | **Collation** | **Null** | **Comment** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PRIMARY | BTREE | Yes | No | id | 1975 | A | No |  |

**team**

| **Column** | **Type** | **Null** | **Default** | **Comments** | **MIME** |
| --- | --- | --- | --- | --- | --- |
| name | varchar(50) | No |  |  |  |
| wins | int(11) | Yes | *NULL* |  |  |
| losses | int(11) | Yes | *NULL* |  |  |
| division | varchar(50) | No |  |  |  |

**Indexes**

| **Keyname** | **Type** | **Unique** | **Packed** | **Column** | **Cardinality** | **Collation** | **Null** | **Comment** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PRIMARY | BTREE | Yes | No | name | 32 | A | No |  |

**trade**

| **Column** | **Type** | **Null** | **Default** | **Comments** | **MIME** |
| --- | --- | --- | --- | --- | --- |
| trade\_id | int(11) | No |  |  |  |
| prop\_id | int(11) | No |  |  |  |
| accepter\_id | int(11) | No |  |  |  |
| proposed\_at | timestamp | Yes | CURRENT\_TIMESTAMP |  |  |
| accepted\_at | timestamp | Yes | *NULL* |  |  |
| confirmed\_at | timestamp | Yes | *NULL* |  |  |

**Indexes**

| **Keyname** | **Type** | **Unique** | **Packed** | **Column** | **Cardinality** | **Collation** | **Null** | **Comment** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PRIMARY | BTREE | Yes | No | trade\_id | 1 | A | No |  |

**trade cards**

| **Column** | **Type** | **Null** | **Default** | **Comments** | **MIME** |
| --- | --- | --- | --- | --- | --- |
| id | int(11) | No |  |  |  |
| trade\_id | int(11) | No |  |  |  |
| card\_id | int(11) | No |  |  |  |
| from\_id | int(11) | No |  |  |  |
| desired | tinyint(1) | No |  |  |  |

**Indexes**

| **Keyname** | **Type** | **Unique** | **Packed** | **Column** | **Cardinality** | **Collation** | **Null** | **Comment** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PRIMARY | BTREE | Yes | No | id | 16 | A | No |  |

**users**

| **Column** | **Type** | **Null** | **Default** | **Comments** | **MIME** |
| --- | --- | --- | --- | --- | --- |
| id | int(11) | No |  |  |  |
| email | varchar(30) | No |  |  |  |
| full\_name | varchar(100) | No |  |  |  |
| password | varchar(40) | No |  |  |  |
| last\_points\_given\_at | timestamp | Yes | *NULL* |  |  |
| points | int(11) | No | 0 |  |  |

**Indexes**

| **Keyname** | **Type** | **Unique** | **Packed** | **Column** | **Cardinality** | **Collation** | **Null** | **Comment** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PRIMARY | BTREE | Yes | No | id | 3 | A | No |  |
| email | BTREE | Yes | No | email | 3 | A | No |  |

**Evaluation of Product**

**Testing Procedures**

**Because the queries that we are running on the database are foundational for getting the application to work like we want, the majority of our testing was done on our queries to make sure they were running correctly. For the most part, this involved inputting test data for the queries and then either observing what they return or how they updated the database tables. Unique sets of data were passed into the queries to see how the database would react, and “boundary” inputs were checked (if an input that is a set only has one element, etc.)**

**Outside of the queries, the application was generally tested by trying to break it. For example, we worked through all possible combinations of trying to enter the trade logic through an illegitimate route, and we checked to see how the application would react to users trying to operate it without being logged in. When an unauthorized use of the application was found to be a possibility, it was resolved and then re-tested.**

**Sample Data and Sample Queries**

**Note: In the application, these queries are protected against sql injection, but for readability they are included in their simplified form here, without any sort of preparation.**

Login(email, password)

SELECT \* FROM users WHERE email=email AND password=password;

Trade\_card(card\_id, new\_owner\_id)

UPDATE card SET current\_owner\_id=new\_owner\_id WHERE card\_id=card\_id;

Insert\_trade\_cards(trade\_id, card\_id, from\_id, desired)

INSERT INTO `trade cards` (`trade\_id`, `card\_id`, `from\_id`, `desired`) VALUES (trade\_id, card\_id, from\_id, desired);

Cancel\_trade(user\_id\_1, user\_id\_2)

DELETE FROM trade WHERE prop\_id=user\_id\_1 AND accepter\_id=user\_id\_2;

Get\_pack(pack\_id)

SELECT \* FROM packs INNER JOIN packs\_players ON packs.pack\_id = packs\_players.pack\_id

INNER JOIN player ON player.id=packs\_players.player\_id

WHERE packs.pack\_id=pack\_id;

Deduct\_points(user\_id, pack\_id)

UPDATE users SET users.points = users.points – (SELECT packs.points FROM packs WHERE packs.pack\_id = pack\_id) WHERE users.id = user\_id;

Sample Data

Users

Id email full\_name password last\_points\_given\_at points

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | aml7hp@virginia.edu | Andy Locascio | ec186cb5f875d6bcc5a584543adc531c | NULL | 0 |
| 4 | zdv8rb@virginia.edu | Zack Verham | 14aa4b5c422edb0c3874b2737b6e9335 | NULL | 8 |
| 5 | ase3b@virginia.edu | Sam Elliott | 5f4dcc3b5aa765d61d8327deb882cf99 | 2013-11-25 20:33:56 | 6 |

Players

Id name number team pos. img\_url

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | A.J. | Bouye | 34 | Houston Texans | CB | NULL |
| 2 | A.J. | Davis | 20 | New Orleans Saints | CB | http%3A//cdn.fansided.com/wp-content/blogs.dir/15/files/2012/12/6410222.jpg |
| 3 | A.J. | Green | 18 | Cincinnati Bengals | WR | NULL |
| 4 | A.J. | Hawk | 50 | Green Bay Packers | ILB | NULL |
| 5 | A.J. | Jefferson | 24 | Minnesota Vikings | CB | http%3A//extras.mnginteractive.com/live/media/site569/2013/0412/20130412\_\_1-aj%2520jefferson\_400.jpg |

Trade

Id prop\_id accepter\_id proposed\_at accepted\_at confirmed\_at

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 | 4 | 1 | 2013-11-25 21:33:12 | 2013-11-25 21:33:53 | 2013-11-25 21:34:45 |
|  |  |  |  |  |  |

Packs

Pack\_id name points

|  |  |  |
| --- | --- | --- |
| 1 | Pack 1 | 2 |
| 2 | Pack 2 | 4 |
| 3 | Pack 3 | 6 |
| 4 | Pack 4 | 8 |
| 5 | Pack 5 | 10 |

Card

Card\_id player\_id current\_owner\_id

|  |  |  |
| --- | --- | --- |
| 2 | 9 | 1 |
| 3 | 5 | 4 |
| 4 | 6 | 1 |
| 5 | 7 | 4 |
| 6 | 5 | 2 |