GREG'S LIST

Final Technical Report

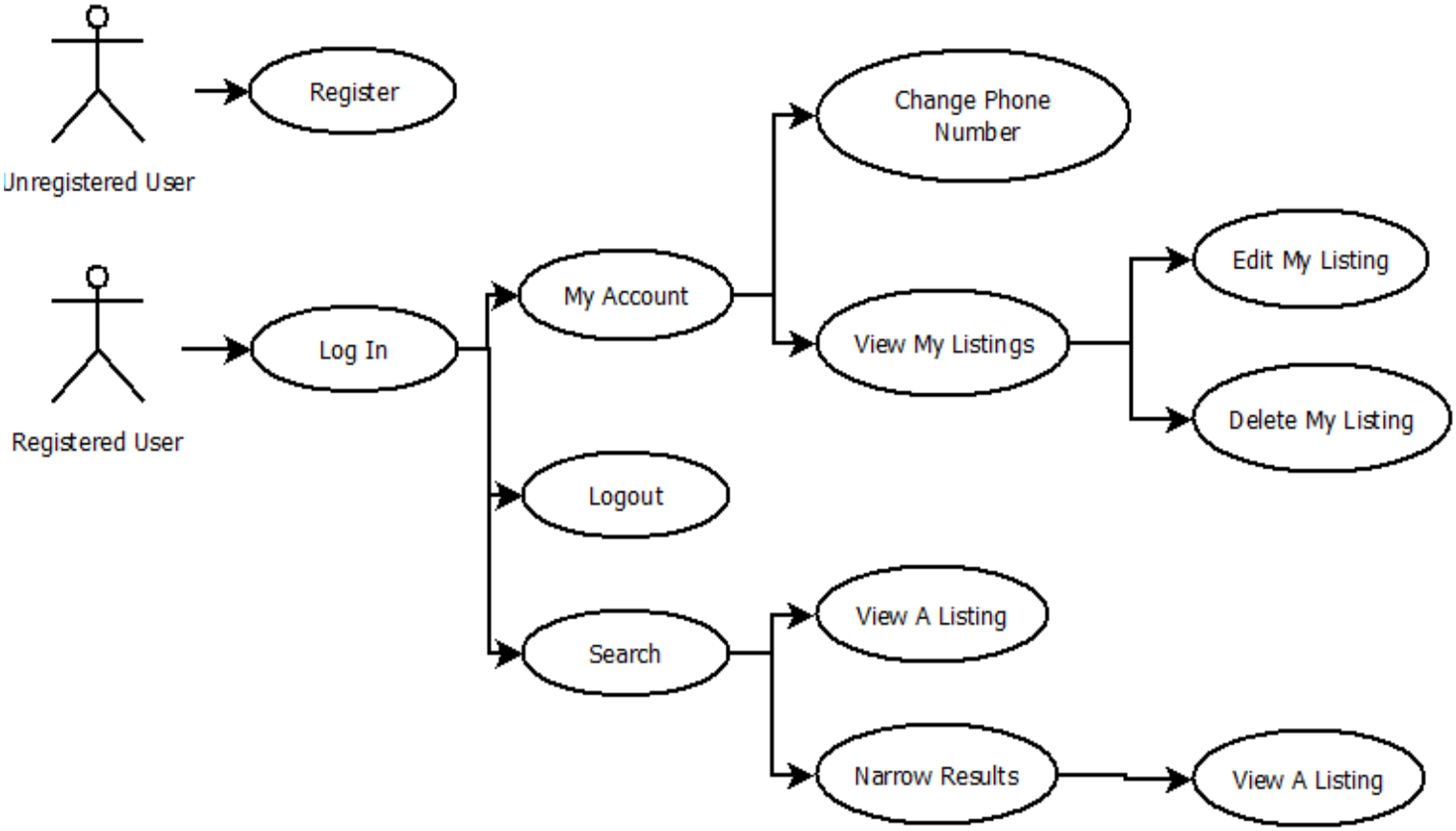
Table of Contents

Team Members

Amanda Doyle – Head Android Developer & Head of Design
Carter Dewey – Chief Website Developer
Bobby Santoski – Middleware Specialist
Greg Walters – Database Manager
Morgan Winslow – Database Specialist

As Head Android Developer, Amanda spent the early part of the project designing and Developing the Android application. As the project progressed, she designed the style aspects of the website and integrated those into the functionality of the web client. Carter Dewey developed the JavaScript components of the website and contributed to the back end functionality to develop the core of the web client. Bobby Santoski specialized in the middle-ware components of the web client functionality using mainly PHP and additional JavaScript. Morgan Winslow specialized in database management and middleware components of the web application. As Database Manager, Greg Walters came up with the design and implementation of the database and worked alongside Bobby and Morgan with the middleware components.

Use Case Diagram



Software Features

WEB CLIENT

- Requires an SMU email address to register
- Requires a registered SMU email address and password to log in
- Adds new users to the database via a signup page
- Redirects users to the home page after successful signup or login
- Allows users to access their account page to edit personal information and listings they created

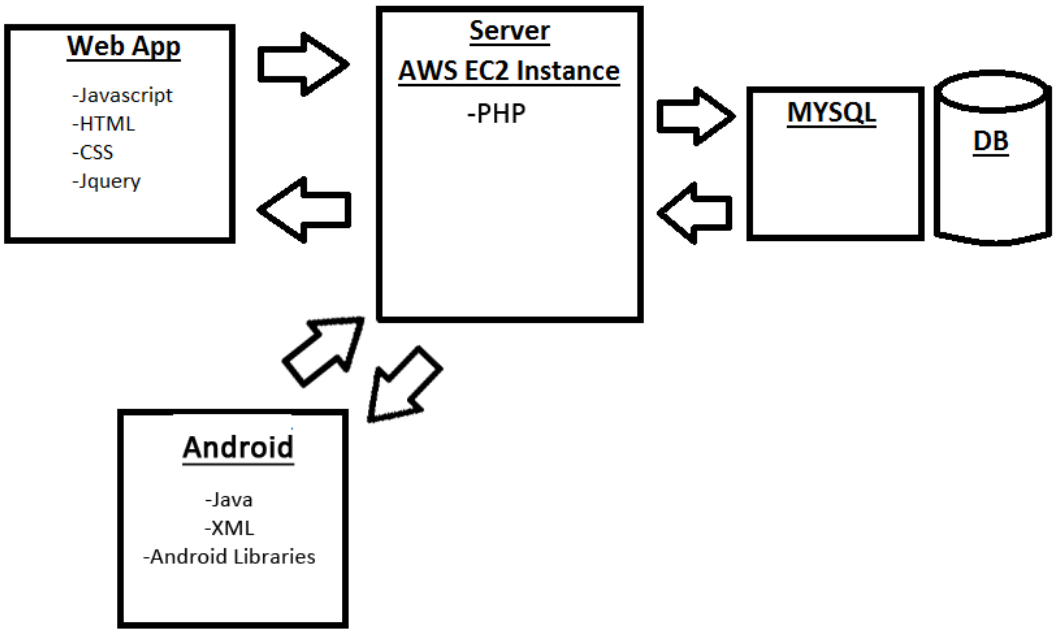
- The software populates creation fields when creating a listing based on category
- Allows users to upload photos associated with their listings
- Allows users to browse listings of a selected category
- Allows users to search for relevant listings – will return results where the search term is in the title or the description – can also search for a category and view all results from that category
- Can filter search results by category
- Displays the five most recent listings as a slide show on the home page
- Allows users to click on the title for a listing to view full listing details
- Lists the contact information of the user who posted the listing so that potential buyers can contact them
- Allows users to log out of the site

ANDROID APPLICATION – LIGHT VERSION

- Requires an SMU email address to register
- Requires a registered SMU email address and password to log in
- Adds new users to the database via a signup page
- Redirects users to the home page after successful signup or login
- Allows users to access their account page to view personal information and listings they created**
- Doesn't allow users to create new listings**
- Doesn't allow users to upload photos associated with their listings**
- Allows users to browse listings of a selected category from a search term**
- Allows users to search for relevant listings – will return results where the search term is in the title only**
- Can filter search results by category
- Displays the 30 most recent listings on the home page
- Allows users to click on the title for a listing to view full listing details
- Lists the contact information of the user who posted the listing so that potential buyers can contact them
- Allows users to log out of the application

**Light version differences from the web application

Software Architecture Diagram



Explanation

Client

WEB

-HTML- Used to format the web pages. Proper formatting allowed for easy styling and manipulation with CSS and JavaScript.
-JavaScript- Allowed for many user interactions such as the ability to create, edit, view, and delete a listing. Interpreted user actions and acted appropriately by changing CSS and calling new HTML.
-CSS- Used to style all of the webpages. Used ids and formatting found in the HTML.
-Jquery- Used in conjunction with JavaScript to streamline implementation

ANDROID

-Java- The workhorse of the app. Includes all methods and code for user functionality.

Includes calls to PHP to manipulate the database.

-XML- Used as the structure and formatter for the android application. Used to create layouts and declare UI elements.

-Android Libraries- Included within the Java in Android to create Android specific functionality

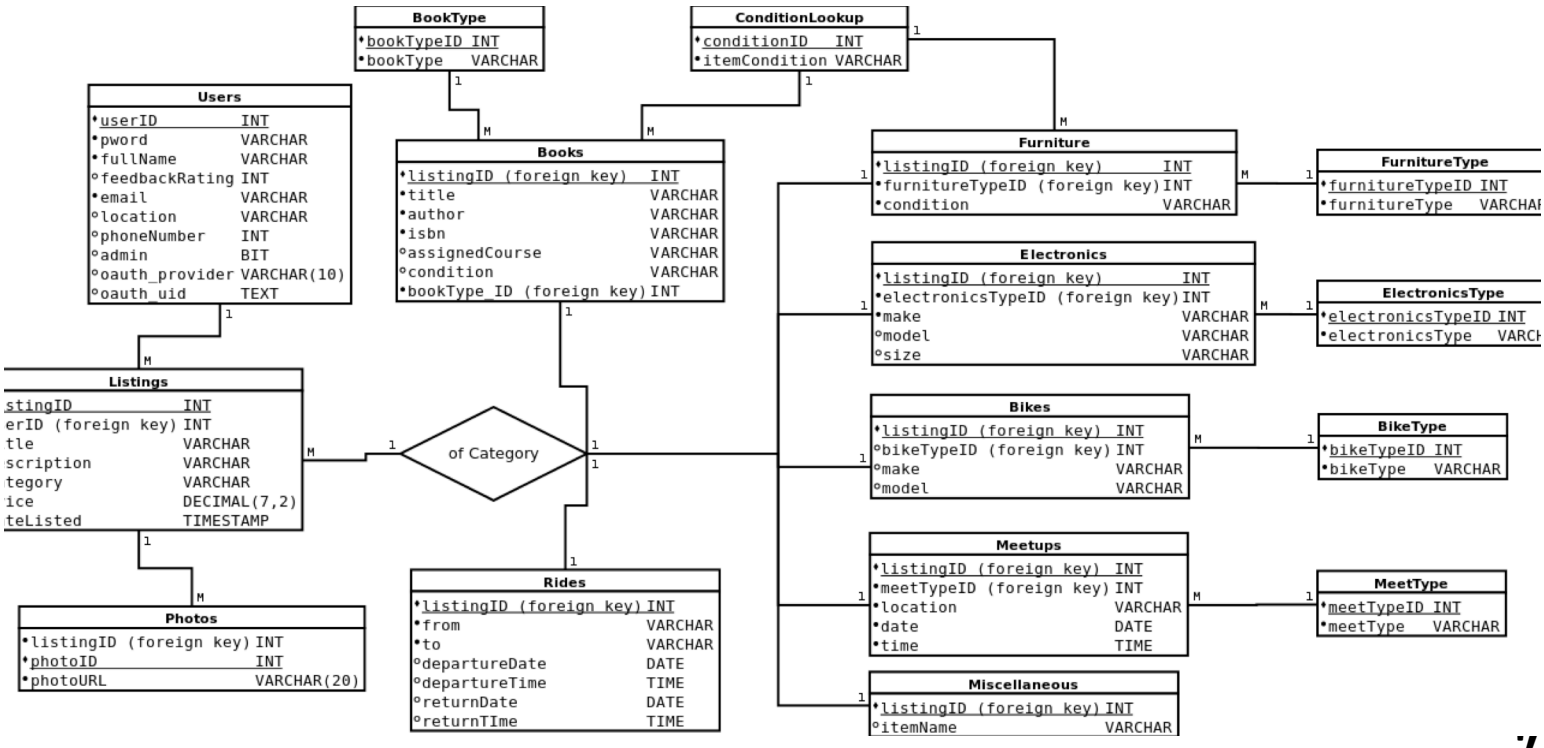
Server

PHP- Used on the server to manipulate information in the database. Called, and sent information, from JavaScript files. Sent MySQL queries to the MySQL database.

Database

MySQL- Database consisting of tables containing user and category information
The MySQL database provided the appropriate information when queried from the PHP scripts.

Physical Database Model



Explanation

The GregsList database model is about linking users with classified advertisements. The Users table contains user contact information, login information, and OAuth information for 3rd party authentication. Their unique user IDs are used to associate them with listings in the Listings table, which contains basic information about the items being sold common to nearly every listing. More detailed information specific to the kind of item being advertised is stored in separate tables for each category, with integer indexes to certain repeated values like type and condition. URLs to photos for listings are also kept in a separate table, theoretically allowing for a listing to have any number of photos. This model allows for varying levels of information to be retrieved as needed.

User Interface

Web Client

Figure 1: Landing Page

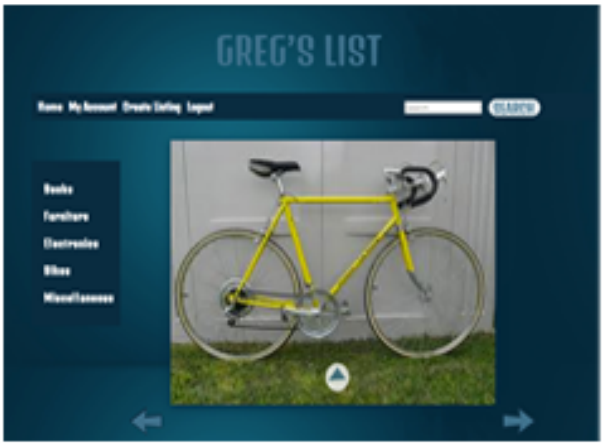


Figure 2: Home Page

Figure 3: Listing Slide Show View Information

Figure 4: Account Page

Figure 5: Create Listing Page

Figure 6: Edit Listing Page

Figure 7: Search Results Page

Figure 8: View Listing Page

Figure 1: The Landing page is the first page the user will see when navigating to the website. From this page the user can either choose to log in if they already have an account set up or to register a new account

Figure 2: The Home page is the first page accessed after successfully logging in. It displays a slide show of the most recent listings and provides links to all of the main pages along with a search bar.

Figure 3: The Listing Slide Show View Information is accessed after clicking on the arrow on the slide show to show more information about that specific listing. It gives an overview of the listing including name, price, and description.

Figure 4: The Account Page, accessible from anywhere once logged in by way of the navigation bar, shows information pertaining to the user's account including email, phone number, and the user's listings. The user can choose to either edit or delete a listing from this page.

Figure 5: The Create Listing Page, accessible from anywhere once logged in by way of the navigation bar, initially displays a name, price, category, and image field. Once a category is chosen fields pertaining to that category appear. The user can fill out these fields and optionally add an image. Once the create listing button is clicked the listing will be posted for other users to view.

Figure 6: The Edit Listing Page, accessible from the account page after clicking on the edit button next to a listing, allows a user to view and resubmit a listing if they want to make changes to it.

Figure 7: The Search Results Page, accessible from either the search bar in the navigation bar or the category bar on the left side of the screen, displays listings based on certain criteria.

The user can choose to enter a specific search in the search bar and then narrow down the Results by category or just choose a category and find the latest listings from that category.

Figure 8: The View Listing Page, accessible from either the search results page or the account page, displays the full details for a listing along with the image associated with that listing. It also displays the contact information of the user who created that listing.

Mobile Application

Figure 1: Landing Page

Figure 2: Login Page

Figure 3: Signup Page

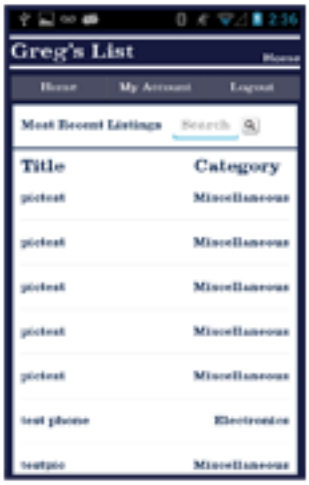


Figure 4: Home Page

Figure 5: Account Page

Figure 6: Search

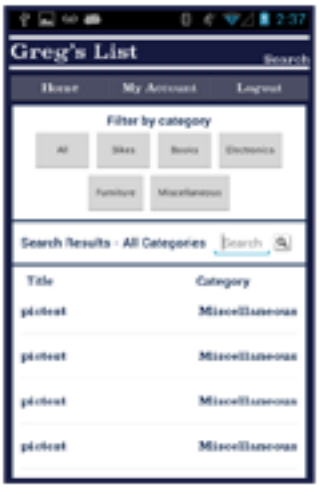


Figure 7: Search Results

Figure 8: View Listing

Figure 1: The landing page is the first page a user will see when they click on the Greg's List icon. The user has a choice to register for Greg's List or login – they make their choice by clicking the respective buttons.

Figure 2: Once the user clicks on the login button they will be taken to the login page. Both fields are required and the user will receive an error message telling them which field they did not fill in. After clicking the login button if the user information is correct they will be taken to their home page. If their information is incorrect, they will receive an error prompt and will be given the chance to try again.

Figure 3: If the user clicks the Sign Up button they will be taken to this sign up screen. The user must enter information into all fields. The application will check to see if the email address is already in the system, and if that is the case, the user will be prompted to try and sign in. The application also makes sure the password entered is between 8 and 12 characters and that the two passwords match.

Figure 4: The home page is where the user is taken after a successful login. The home page shows the 30 newest listings. Each listing is clickable and will take the user to the view listing screen. The user can also search all listings from this page by entering a keyword into the search box.

Figure 5: The account page displays the user's contact information as well as all of their listings. Each listing is clickable and will take the user to the view listing screen.

Figure 6: A user can search all listings from the home page or from the search results page. All the user has to do is type in a search term and click either the search button on the keyboard or the search icon on the screen.

Figure 7: The search page returns all of the listings whose title contain the search term. Each listing is clickable and will take the user to the view listing screen. Search results can be filtered by category by clicking on any of the category buttons.

Figure 8: The view listing page shows all of the details of the listing that was clicked as well as the contact information for the user who posted the listing.

Testing

We tested out product in two different ways. Firstly we performed usability tests with both professor Raley and multiple people not in the classes. Secondly we had our website tested by another team doing the same assignment. For the usability tests the main feedback we received was suggestions to improve navigation of the website in a logical manner. The team that tested our product was helpful in finding bugs and giving feedback on the website. We looked into and attempted to solve every bug posted and considered all of the feedback given for the final product. As a test team I believe our team was quite helpful in finding bugs and giving feedback. We were able to find bugs that the other team had missed and they were able to use this information to improve their product. All of our bug reports and feedback was at least considered and responded to and much of it was implemented.

Team Reflection

By far a large amount of the technical problems came with PHP. Some of the most challenging parts were returning JSON from a PHP file to a JavaScript file, managing sessions properly, uploading photos and creating multiple ways to search and filter results from the database. We solved these problems with a fair amount of hacking and lots of Google searching.

To return JSON from a PHP file to JavaScript we were using an XMLHTTPRequest from the JavaScript to call the PHP file. For a very long time we were able to confirm that the PHP file was generating valid JSON but were not sure why it was not being returned to the JavaScript in 'responseText.' The issue was that we were using 'return $json' in our PHP files while the correct syntax was 'echo $json.' Such a simple mistake but one no one knew to use 'return.'

Managing sessions didn't have any tricky syntax we simply used the 'session_start();' command and to access data from the session we could use the '$_SESSION[ ]' array. The tricky part came through maintaining sessions within pages and ending sessions on logout. For example when we loaded our 'myAccount.php' file there and several calls to other PHP files which each had a start session command. It turns out that we only need to call the start session command once per page load so we were getting annoying output telling us our sessions were being restarted. When logging out we always called the 'session_destroy();' command. However it was possible on logout for users to hit the back button and be redirected back into the website without logging in. We fixed that by doing a check at the top of each page to see is the session userID was set to null and if so the user is redirected back to the login page.

Photo upload was probably the single hardest thing to accomplish. The photo upload was difficult because it depended on so many different things working. First, the program has to check the type of the image and move it to a directory on the server. For this to work, permissions on the /tmp and User_Photos directories have to be such that the program could create files there. It also needs to be saved with a unique file name that we know, so we implemented an auto-increment scheme outside of MySQL. Then, it's new file name should be stored in the database with the listing's listingID, but only if the file is successfully uploaded and the rest of the listing is also. Anything that caused the row in Listings not to be inserted would also prevent the photo information from being inserted into the database because of the foreign key constraint. All of these requirements made this segment of code one of the most persnickety, and all of these things had to be checked on the local build environments and the server any time the uploader stopped working.

The search function was complex because we offered several ways in which the user could search/filter results. The actual searching and filtering functionality was handled by MySQL. The struggle was sending the proper data to those functions. On every page but the login page there is a category bar where users can select one of 5 categories and they are taken to a results page with all listings from that category. On every page but the login page there is also a search bar that allows users to enter custom queries and all results matching those queries are displayed with the option to filter by category. The problem was that I had to send the data of which category was clicked or what search term was entered to the search results page for further processing. I managed to do this with the 'window.location' method by manually setting the url to 'searchResults.html?text' or 'searchResults.html?category' and then extracting that extra string in search results.

If we were to do this project again we would have a lot more communication between the PHP development and the front end development. We would often try calling PHP files that were expecting totally different input arguments than what was being supplied. Along with this we would recommend following group structure: 2 MySQL/PHP devs, 2 HTML/JavaScript/PHP devs and 1 Android dev. This allows for both the front and back end structures to be created then had more than ample man power to work on the communication between the two ends. We would have also designed a site with just the absolute core features added. We had to cut a few features from our original design because they would have simply took to long to implement. We would also recommend trying to complete one core feature per week to evenly space out the work.

2.0 Features

For future implementations of our product we would like to implement the following features:
-Additional Categories such as Rides and Meet ups
-“Watched listings” feature that would let users bookmark a listing and return to it later
-Multiple pictures attached to one listing
-Captions for images
-Password reset feature
-Search by other criteria such as ISBN or Model
-Email verification for account creation
-Add alternate/additional emails to a user account

Appendix A: Data Dictionary

Users

The Users table contains user login credentials, contact information, an ID for linking to listings, and OAuth information.

- userID is an auto-incremented integer that serves as the primary key.
- pword is the user's password
- fullName is a single text value for the name the user wishes to display
- feedbackRating is an integer meant to hold the user's feedback rating, but was not used.
- email is a text value to hold the user's SMU e-mail address
- location is a text field meant to describe which dorm or apartment complex the user lives in
- phoneNumber is an integer for storing the phone number (if any) the user wishes to be contacted at in response to listings
- admin is a single bit used to indicate if the user has administrator privileges or not. 1 indicate that the user has administrator privileges, 0 indicates a regular user and is the default value.
- oauth_provider is a text value used to indicate which 3rd party authentication the oauth_uid belongs to.
- oauth_uid is a text value used to store a 3rd party authentication ID.

userID is the primary key. Users has no foreign key constraints, and 1 user may have 0 to many Listings.

Listings

The Listings table contains descriptive information common to all listing types, a unique ID, and the ID of the user that created it.

- listingID is an auto-incremented integer that serves as the primary key.
- userID is a foreign key used to indicate the user that created it.
- title is a text value meant to serve as a very short description of the item being listed
- dateListed is a timestamp used to indicate the time created and sort returned listings.
- category is a text value used to indicate what kind of listing it is for sorting purposes and for looking up the detailed information
- price is a 7-place decimal to the hundreths place for storing the asking price for the item being offered
- description is a large text value for describing the item being offered however the user chooses

Listings has primary key listingID and foreign key userID. A user may have many Listings, and if a user is deleted, the child Listings are deleted as well. A listing may have 0 to many Photos, and must have one category (Bikes, Books, Furniture, Electronics, Meetups, or Miscellaneous).

Bikes

The Bikes table contains detailed listing information specific to bicycles.

- listingID is an integer foreign key used to indicate which Listing the information belongs to. It also serves as the primary key.
- bikeTypeID is an integer foreign key used to represent the type of bicycle and is used to look up a text value in BikeType. This forces the BikeType to be of a type specified by GregsList.
- make is a text value to specify the brand of the bicycle.
- model is a text value to specify the model of the bicyle.

Bikes has primary key listingID and two foreign keys. A Bikes row belongs to a Listing and is bound by foreign key listingID. If the parent Listing is deleted, the Bikes row is also deleted. BikeType rows cannot be deleted unless they have no children in Bikes. A Bikes row belongs to one listing and has one BikeType.

BikeType

BikeType is a lookup table for getting text values for the different types of bicycles in GregsList.

- bikeTypeID is the primary key, currently a value between 0 and 7.
- bikeType is the text representation of a type of bike.

bikeTypeID	bikeType
0	other
1	road
2	city
3	mountain
4	cruiser
5	hybrid
6	unicycle
7	electric

BikeType has primary key bikeTypeID and no foriegn keys. One BikeType may belong to many Bikes. A BikeType row cannot be deleted while a child Bike row exists.

Books

The Books table contains detailed listing information specific to Books.

- listingID is an integer foreign key used to indicate which Listing the information belongs to. It also serves as the primary key.
- bookTypeID is an integer foreign key used to represent the type of book and is used to look up a text value in BookType. This forces the BookType to be of a type specified by GregsList.
- title is a text value for holding the title of the book being sold.
- author is a text value for holding the author of the book being sold.
- isbn is a text value for holding the isbn of the book being sold.
- assignedCourse is a text value for holding the name of any SMU course that may be using the textbook
- conditionID is an integer foreign key used to look up the condition of the book being sold.

Books has primary key listingID and three foreign keys. One Books row belongs to one Listings row and has one bookType and one condition. If its parent Listing is deleted, it is also deleted.

BookType

BookType is a lookup table for the types of books available on GregsList.

- bookTypeID is an integer primary key, currently a value between 0 and 6.

- bookType is a text value describing the kind of book being sold.

BookType has primary key bookTypeID and no foriegn keys. One BookType may belong to many Books. A BookType row cannot be deleted while a child Book row exists.

bookTypeID	bookType
0	other
1	textbook
2	literature
3	poetry
4	reference
5	nonfiction
6	graphic novel

ConditionLookup

ConditionLookup is a lookup table for the condition of used books and furniture.

- conditionID is an integer primary key, currently a value between 0 and 5.
- itemCondition is a text value describing the condition of the item being sold.

conditionID	itemCondition
0	unspecified
1	new
2	excellent
3	light wear
4	normal wear
5	damaged

ConditionLookup has primary key conditionID and no foreign keys. One conditionID may belong to many Books or Furniture rows. A ConditionLookup row cannot be deleted while a child Book or Furniture row exists.

Electronics

The Electronics table holds detailed listing information specific to electronics.

- listingID is an integer foreign key used to indicate which Listing the information belongs to. It also serves as the primary key.
- electronicsTypeID is an integer foreign key used to represent the type of device and is used to look up a text value in electronicsType. This forces the electronicsType to be of a type specified by GregsList.
- make is a text value to specify the brand of the device.
- model is a text value to specify the model of the device.
- size is a text value to specify either physical dimensions, as might be the case with a television, or storage capacity, as might be the case with an mp3 player or hard drive.

Electronics has primary key listingID and two foreign keys. One Electronics row belongs to one Listings row and has one electronicsType. If its parent Listing is deleted, it is also deleted.

ElectronicsType

ElectronicsType is a lookup table for the types of devices available on GregsList.

- electronicsTypeID is an integer primary key, currently a value between 0 and 5.
- electronicsType is a text value describing the kind of device being sold.

electronicsTypeID	electronicsType
0	other
1	TV
2	computer
3	phone
4	speakers
5	iPod

ElectronicsType has primary key electronicsTypeID and no foriegn keys. One ElectronicsType may belong to many Electronics. An ElectronicsType row cannot be deleted while a child Electronics row exists.

Furniture

The Furniture table holds detailed listing information specific to furniture.

- listingID is an integer foreign key used to indicate which Listing the information belongs to. It also serves as the primary key.
- furnitureTypeID is an integer foreign key used to represent the type of device and is used to look up a text value in FurnitureType. This forces the furnitureType to be of a type specified by GregsList.
- conditionID is an integer foreign key used to look up the condition of the furniture being sold.

Furniture has primary key listingID and three foreign keys. One Furniture row belongs to one Listings row and has one furnitureType. If its parent Listing is deleted, it is also deleted.

FurnitureType

FurnitureType is a lookup table for the types of furniture available on GregsList.

- furnitureTypeID is an integer primary key, currently a value between 0 and 6.
- furnitureType is a text value describing the kind of furniture being sold.

furnitureTypeID	furnitureType
0	other
1	couch
2	bed
3	chair
4	table
5	futon
6	shelves

FurnitureType has primary key furnitureTypeID and no foriegn keys. One FurnitureType may belong to many Furniture rows. A FurnitureType row cannot be deleted while a child Furniture row exists.

Meetups

The Meetups table contains detailed listing information specific to a group meeting rather than the sale of an item.

- listingID is an integer foreign key used to indicate which Listing the information belongs to. It also serves as the primary key.
- meetupTypeID is an integer foreign key used to represent the type of gathering and is used to look up a text value in MeetupType. This forces the MeetupType to be of a type specified by

GregsList.
- location is a text value describing where the gathering will take place
- date is a date value describing the day on which the gathering will take place
- time is a time value describing the time at which the gathering will start

Meetups has primary key listingID and foreign keys listingID and meetupTypeID. A Meetups row belongs to one Listings row and is deleted if the parent listing is deleted. A Meetups row has one MeetupType.

MeetupType

MeetupType is a lookup table linking a meetupTypeID to a text description of the kind of gathering.

- meetupTypeID is an integer primary key, currently a value between 0 and 3.
- meetupType is a text value describing the type of gathering

MeetupType has primary key and foreign key meetupTypeID. One MeetupType belongs to one Meetups row. A MeetupType row cannot be deleted while a child Meetups row exists.

Miscellaneous

Miscellaneous is a table to serve as a catch-all for whatever listings do not fit another GregsList category.

- listingID is an integer foreign key used to indicate which Listing the information belongs to. It also serves as the primary key.
- itemName is a text field that can store a short descriptor for an item, but is not used in the final design.

Miscellaneous has primary and foreign key listing ID. Miscellaneous rows each belong to one Listings row. If the parent Listings row is deleted, the child row is also deleted.

Rides

Rides contains detailed listing information for carpooling rather than the sale of an item.
- listingID is an integer foreign key used to indicate which Listing the information belongs to. It also serves as the primary key.
- leavingFrom is a text value describing the location from which the driver will leave
- goingTo is a text value describing the destination to which the driver is going

- departureDate is a date value describing the day on which the driver will leave
- departureTime is a time value describing the time at which the driver will leave
- returnDate is a date value describing the day on which the driver will return
- returnTime is a time value describing the time at which the driver intends to return

Rides has primary key and foreign key listingID. A Rides row belongs to one Listings row and is deleted if the parent listing is deleted.

Photos

Photos links listings to photos stored in the server.

- listingID is an integer foreign key used to indicate which Listing the information belongs to.
- photoID is an integer primary key used to order and keep track of the photos.
- photoURL is a text value that holds the file name of the photo

Photos has foreign key listingID and each Photos row belongs to one Listings row. A Listing may have more than one photo. If a parent Listing is deleted, all child Photos are deleted.