Shop E-Z-Go Mobile

Project Delivery

December 10, 2012

Team^2

Table of Contents

Release Information

Current Version	1.0
Status	Working Correctly
Known Bugs	The text area on a button is not tapable

Installation Instructions

These instructions are to run our site as is. In order to run the site with the real database, some changes will need to be made as described in the Additional Information section.

.

1. Unzip ezgomobile.zip in the home directory for hosting the site. The html pages should all be in the main folder, and all the other pages will be in the unzipped subfolders (more information on these later).
2. Install PHP on the server if it does not already exist (for detailed instructions, see http://php.net/manual/en/install.php).
3. Install mySQL on the server if it does not already exist (for detailed instructions, see http://dev.mysql.com/doc/refman/5.5/en//installing.html).
    a. While not necessary, for easy access and use of the fake database, install PhpMyAdmin (for detailed instructions, see http://wiki.phpmyadmin.net/pma/Quick_Install)
4. Run the SQL code, found in the attached SQL file, using mySQL in order to create the fake database used for this website.

User Guide

This guide contains step by step instructions on how to use the mobile website. The features are listed in alphabetical order by feature name to make it easier to find a specific objective.

Add Part to Shopping Cart (if logged in)

1. Tap Parts in the header bar
2. Find a part by search or selection
3. Tap View (part name) button
4. Tap Add to Cart button

Checkout (if logged in)

1. Tap My Account button in the top left
2. Tap Shopping Cart
3. Tap Checkout

Create Account

1. Tap My Account button in the top left
    a. Tap Logout if currently Logged in
2. Tap Create New Account
3. Fill in all of the fields
4. Tap Submit

Find a Dealer

1. Tap Find a Dealer in the header bar
2. Tap to Share Location in the browser popup (if it appears)
3. Tap an icon on the map to view dealer information

Find a Dealer Near a Given Zip Code

1. Tap Find a Dealer in the header bar
2. Tap to Share Location in the browser popup (if it appears)
3. Type Zip Code into “Enter Zip Code” text box
4. Tap the Go button
5. Tap an icon on the map to view the appropriate dealer information

Find a Part by Search

1. Tap Parts in the header bar
2. Type in the search query into the “Enter Part ID or Keyword” text box
3. Tap Search
4. To view the next ten results, tap the Next 10 button at the bottom of the page
5. To view the previous ten results, tap the Previous 10 button at the bottom of the page

6. To view a specific part, tap the View (part name) button

Find a Part by Selection
1. Tap Parts in the header bar
2. Tap the part category you want
3. Tap the part subcategory you want
4. To view the next ten results, tap the Next 10 button at the bottom of the page
5. To view the previous ten results, tap the Previous 10 button at the bottom of the page
6. To view a specific part, tap the View (part name) button

Find Vehicle by Selection
1. Tap My Vehicle in the header bar
2. Tap Tap Here to select a Vehicle button
3. Tap the model you want
4. Tap the fuel type you want
5. Tap the submodel you want
6. Tap the year you want

Find Vehicle by Serial Number
1. Tap on My Vehicle in the header bar
2. Enter Serial Number into the text field labeled “Enter Serial Number”
3. Tap GO button

Login
1. Tap My Account button in the top left
    a. If currently logged in, tap Logout
2. Enter username in the Username box and password in the Password box
3. Tap Login

Logout (if currently logged in)
1. Tap My Account button in the top left
2. Tap Log Out

Remove Part from Shopping Cart (if currently logged in)
1. Tap My Account button in the top left
2. Tap Shopping Cart
3. Tap Remove Part button below item you wish to remove from the shopping cart.

Remove Vehicle from Account (if currently logged in)
1. Tap My Account button in the top left
2. Tap Account Vehicles
3. Tap the Remove Vehicle button below the vehicle you wish to remove from the account.

Save Vehicle to Account (if currently logged in)

1. Tap My Vehicle in the header bar
2. Find the appropriate vehicle using the serial number or by selection
3. Tap Save to Account

Select Vehicle from Account to be a part filter (if currently logged in)
1. Tap My Account button in the top left
2. Tap Account Vehicles
3. Tap Select Vehicle button below the vehicle you wish to select

Select Vehicle from My Vehicle page to be a part filter
1. Tap My Vehicle in the header bar
2. Find the appropriate vehicle using the serial number or by selection
3. Tap the Shop Now button

Update Account (if currently logged in)
1. Tap My Account button in the top left
2. Tap Update Account
3. Enter the fields that you wish to update
    a. If trying to change password you must enter your current password into the current password field as well.
4. Tap Update Account

View EZGO Contact Information
1. Tap Contact Us on the footer bar
2. Read list of contact information

View EZGO TV
1. Tap EZGO TV in the header bar
2. Tap the button for the category of video you wish to view
3. Tap the Youtube link to view a specific video

View the Full Website (shopezgo.com)
1. Tap Full Site in the footer bar

Requirements and Design

See included files in the documentation folder. Specifically:
- Architecture Design Rationale
- Design Validation
- Detailed Design - Static and Dynamic
- Dynamic Architecture
- Er Diagram

        - Static Architecture Design
        - Project Plan

Database Connection

Since we did not have access to the shopezgo databases when we started the project, we had to make our own temporary database. We knew the site would eventually have to be connected to the main database so we tried to make this as easy as possible. Right now our code is broken into 5 layers as shown in the Static Architecture document. The top two layers (Presentation and Interaction Logic) should require no change at all. The Business logic layer will require minor change. The Business logic layer was created to easily allow the data layer to change with minimal effort. This layer makes an http request to the Data Access Layer which makes the calls to the database. Currently it points to the temporary Data Access code. To change this:

- In the file /server/init.ini change the url to the url of the new Data Access web service.

The last layer (Data) would need to be completely replaced with the existing database, so that should also take no effort. The only layer that will need to be actively rewritten is the Data Access layer. This layer makes SQL calls and writes out the results as a json that can be read by the Business Logic layer and will need to be changed in one of two ways.

- If using PHP to access the database is an option, only two changes are needed. First, change the file /db/init.ini to have the login information of the correct database. Second, all the SQL calls in /db/sql.ini will have to be rewritten as appropriate. In order to make sure the new calls work with the existing code correctly, make sure that the JSON object return has the correct format (for the specific format of a particular JSON on a particular page, see the List of JSON Objects section in this document)
- If PHP cannot be used to access the database, simply do not use any of the files in the /db folder and instead write your own Data Access Layer. There are a few caveats to this. First, the database calls need to return the columns names as mentioned above (look at the JSON section on page 9 for details). Second, you must have a response ready for each possible function in /db/index.php (see that file for more details). Finally, you must output on the web page a json object in the same format as the jsonResponse function in index.php (which will be decoded by functions.php)

List of JSON Objects

List of JSON objects used by each JS function (in alphabetical order by page name).

- accountVehicles.js
    - Default AJAX call-
      {
        success- true if the PHP call returned successfully, false otherwise.
        data- Array set only if success was true. Each entry has the following format

        {
            image- A string URL of the vehicle image.
            model- A string of the model name.
            fuel- A string of the fuel name.
            submodel- A string of the submodel name.
            year- A string of the year name
            serialNumber- A string of the vehicle serial number.
        }
        errors- Array set only if success was false. It has the following format:
        {
            reason- A string with a description of what went wrong.
        }
    }

    - selectVehicle-

      {

        success- true if the PHP call returned successfully, false otherwise.

        data- Array set only if success was true. Each entry has the following format

        {

            found- A boolean indicating if the vehicle exists or not.

            image- A string URL of the vehicle image.

            model- A string of the model name.

            fuel- A string of the fuel name.

            submodel- A string of the submodel name.

            year- A string of the year name

            serialNumber- A string of the vehicle serial number.

        }

        errors- Array set only if success was false. It has the following format:

        {

            reason- A string with a description of what went wrong.

        }

      }

    - removeVehicle- No JSON response.

- ezgotv.js
    - retrieveVideoList-

      {

        success- true if the PHP call returned successfully, false otherwise.

        data- Array set only if success was true. Each entry has the following format:

        {

            URL- A string of the video URL.

            Image- A string of the video thumbnail image.

            Title- A string of the video title.

}
errors- Array set only if success was false. It has the following format:
{
reason- A string with a description of what went wrong.
}
}

- parseList- No AJAX call
- showInstallation- No AJAX call
- showFeatured- No AJAX call
- showPerformance- No AJAX call
- showMaintenence- No AJAX call

- findADealer.js
    - initialize- No AJAX call
    - getUserLocation- No AJAX call
    - retrieveDealers-
      {
      success- true if the PHP call returned successfully, false otherwise.
      data- Array set only if success was true. Each entry has the following format:
      {
      address- A string of the dealer address
      city- A string of the dealer city
      state- A string of the dealer state.
      zip- A string of the dealer zip code.
      name- A string of the dealer name.
      phone- A string of the dealer phone number.
      url- A string of the dealer URL.
      }
      errors- Array set only if success was false. It has the following format:
      {
      reason- A string with a description of what went wrong.
      }
      }
    - panToZip- No AJAX call
    - displayError- No AJAX call

- login.js
    - login-
      {
      success- true if the PHP call returned successfully, false otherwise.
      data- Array set only if success was true. It has the following format:
      {
      auth- A boolean indicating if the login was successful.

userName- A string of the model name.
accountId- A string of the accountId
}
errors- Array set only if success was false. It has the following format:
{
reason- A string with a description of what went wrong.
}
}

- loginHeader.js
    - myAccount-
      {
      userName- A string of the current user's userName or undefined if the user is not currently logged in.
      }

- myAccount.js
    - updateAccount- No AJAX call
    - accountVehicles- No AJAX call
    - shoppingCart- No AJAX call
    - logout- No JSON response

- myVehicle.js
    - serialNumberSearch-
      {
      success- true if the PHP call returned successfully, false otherwise.
      data- Array set only if success was true. It has the following format:
      {
      found- A boolean indicating if the vehicle exists or not.
      }
      errors- Array set only if success was false. It has the following format:
      {
      reason- A string with a description of what went wrong.
      }
      }
    - selectVehicle- No AJAX call

- partInfo.js
    - Default AJAX call-
      {
      success- true if the PHP call returned successfully, false otherwise.
      data- Array set only if success was true. It has the following format:
      {
      name- A string of the part name.

description- A string of the part description.
price- A string of the part price.
categoryName- A string of the part category name.
subcategoryName- A string of the part subcategory name.
availability- A string of the part availability.
image- A string URL of the part image.
}
errors- Array set only if success was false. It has the following format:
{
reason- A string with a description of what went wrong.
}
}

- addToCart-
  {
  success- true if the PHP call returned successfully, false otherwise.
  errors- Array set only if success was false. It has the following format:
  {
  reason- A string with a description of what went wrong.
  }
  }

- parts.js
  - First default AJAX call
    {
    success- true if the PHP call returned successfully, false otherwise.
    data- Array set only if success was true. Each entry has the following format
    {
    found- A boolean indicating if the vehicle exists or not.
    image- A string URL of the vehicle image.
    model- A string of the model name.
    fuel- A string of the fuel name.
    submodel- A string of the submodel name.
    year- A string of the year name
    }
    errors- Array set only if success was false. It has the following format:
    {
    reason- A string with a description of what went wrong.
    }
    }
  - Second default AJAX call
    {
    success- true if the PHP call returned successfully, false otherwise.

                data- Array set only if success was true. Each entry has the following
                format
                {
                        categoryId- A string of the part category id.
                        name- A string of the part category name.
                }
                errors- Array set only if success was false. It has the following format:
                {
                        reason- A string with a description of what went wrong.
                }
        }
    - cancel- No JSON response
    - partsSearch-

        {
                success- true if the PHP call returned successfully, false otherwise.
                errors- Array set only if success was false. It has the following format:
                {
                        reason- A string with a description of what went wrong.
                }
        }
    - selectCategory- No JSON response

- partsSearchResults.js
    - getResults-

        {
                success- true if the PHP call returned successfully, false otherwise.
                data- Array set only if success was true. Each entry has the following
                format
                {
                        image- A string URL of the part image.
                        name- A string of the part name.
                        partNumber- A string of the part number.
                        price- A string of the part price.
                }
                errors- Array set only if success was false. It has the following format:
                {
                        reason- A string with a description of what went wrong.
                }
        }
    - selectPart- No JSON response

- register.js
    - register-

        {

success- true if the PHP call returned successfully, false otherwise.
data- Array set only if success was true. It has the following format:
{
userName- A string of the model name.
accountId- A string of the accountId
}
errors- Array set only if success was false. It has the following format:
{
reason- A string with a description of what went wrong.
}
}

- selectFuel.js
    - Default AJAX call-
      {
      success- true if the PHP call returned successfully, false otherwise.
      data- Array set only if success was true. Each entry has the following format
      {
      fuelId- A string of the vehicle fuel id.
      name- A string of the vehicle fuel name.
      }
      errors- Array set only if success was false. It has the following format:
      {
      reason- A string with a description of what went wrong.
      }
      }
    - selectFuel- No JSON response
- selectModel.js
    - Default AJAX call-
      {
      success- true if the PHP call returned successfully, false otherwise.
      data- Array set only if success was true. Each entry has the following format
      {
      modelId- A string of the vehicle model id.
      name- A string of the vehicle model name.
      }
      errors- Array set only if success was false. It has the following format:
      {
      reason- A string with a description of what went wrong.
      }
      }
    - selectModel- No JSON response

- selectSubcategory.js
    - Default AJAX call-
      {
        <u>success</u>- true if the PHP call returned successfully, false otherwise.
        <u>data</u>- Array set only if success was true. Each entry has the following format
        {
          <u>subcategoryId</u>- A string of the part subcategory id.
          <u>name</u>- A string of the part subcategoryname.
        }
        <u>errors</u>- Array set only if success was false. It has the following format:
        {
          <u>reason</u>- A string with a description of what went wrong.
        }
      }
    - selectSubcategory- No JSON response

- selectSubmodel.js
    - Default AJAX call-
      {
        <u>success</u>- true if the PHP call returned successfully, false otherwise.
        <u>data</u>- Array set only if success was true. Each entry has the following format
        {
          <u>subcategoryId</u>- A string of the vehicle submodel id.
          <u>name</u>- A string of the vehicle submodel name.
        }
        <u>errors</u>- Array set only if success was false. It has the following format:
        {
          <u>reason</u>- A string with a description of what went wrong.
        }
      }
    - selectSubmodel- No JSON response

- selectYear.js
    - Default AJAX call-
      {
        <u>success</u>- true if the PHP call returned successfully, false otherwise.
        <u>data</u>- Array set only if success was true. Each entry has the following format
        {
          <u>yearId</u>- A string of the vehicle year id.
          <u>name</u>- A string of the vehicle yearname.

}
errors- Array set only if success was false. It has the following format:
{
reason- A string with a description of what went wrong.
}
}

- selectYear- No JSON response

- shoppingCart.js
  - Default AJAX call-

{
success- true if the PHP call returned successfully, false otherwise.
data- Array set only if success was true. Each entry has the following format
{
image- The string URL of the part.
name- The string of the name of the part.
price- The string of the price of the part.
partNumber- The string of the part number.

}
errors- Array set only if success was false. It has the following format:
{
reason- A string with a description of what went wrong.
}
}

  - checkout- No JSON response
  - removePart- No JSON response

- updateAccount.js
  - Default AJAX call-

{
success- true if the PHP call returned successfully, false otherwise.
data- Array set only if success was true. Each entry has the following format
{
email- A string of the account email.
address- A string of the account address.
city- A string of the account city.
state- A string of the account state.
zip- A string of the account zip code.
offers- A string for if the account is getting offers or not (‘Y’ means true, anything else means false.)
}
errors- Array set only if success was false. It has the following format:

{
reason- A string with a description of what went wrong.
}
}

  - updateAccount-
    {
    success- true if the PHP call returned successfully, false otherwise.
    data- Array set only if success was true. It has the following format:
    {
    updated- A boolean that is true if the update was successful and false otherwise.
    }
    errors- Array set only if success was false. It has the following format:
    {
    reason- A string with a description of what went wrong.
    }
    }

- vehicleResults.js
  - Default AJAX call-
    {
    success- true if the PHP call returned successfully, false otherwise.
    data- Array set only if success was true. It has the following format:
    {
    model- A string of the vehicle model.
    fuel- A string of the vehicle fuel.
    submodel- A string of the vehicle submodel.
    year- A string of the vehicle year.
    image- A string URL of the vehicle image.
    }
    errors- Array set only if success was false. It has the following format:
    {
    reason- A string with a description of what went wrong.
    }
    }
  - cancel- No JSON response
  - saveToAccount- No JSON response
  - shopNow- No JSON response

Shopping Cart and Checkout

Right now the checkout button does not do anything other than clear out the user's shopping cart. In order to connect the shopping cart to the checkout functionality of shopezgo.com code would have to be added to the file javascript/shoppingCart.js. In this file there is a checkout function that currently calls checkout in server/functions.php. While this still would need to be called to clear out the shopping cart, an additional call would need to be made to whatever file/ system takes care of receiving payment and setting up the order. This would likely either be a simple call to a different javascript file/html page, or a new function in server/functions.php which would in turn call the actual checkout functionality.

Future Work

Serial number and barcode scanner:

While adding this to the design is simple, actually coding it in practice is very difficult. Most likely a separate program will need to be created on the server to do image processing. There are several open source libraries that can facilitate this depending on the type of programs allowed to run on the server. If the server can run c++ code, one of the better libraries to use would be OpenCV (opencv.org) because of the vast amount of documentation and integration with different platforms. This program may also make a good native application on Android or iOS. Our recommendation is to contact Professor Waters with this project for another semester Sr. Design group.

Once this software has been created integration to the website should be minimal. A file uploader will need to be added to the myVehicle page. Functionality will likely need to be placed in the myVehicle.js file to upload the file to the server using an Ajax call. Once the image has been processed on the server a JSON object with the serial number can be used to retrieve the vehicle information using the "Find Vehicle by Serial" function we have created. If a native application is created there will need to be slight modifications to the page in order to retrieve the vehicle information using the serial number.

SQL

- Please see the attached sql document EZGoMobile.sql located in the db folder of the zipped file.
- This SQL will generate our temporary database used to create the prototype.
- Simply run this in some SQL software and it will create the temporary database.