SW Engineering CSC648/848 Fall 2022

MarketGator

Team 05

Patrick Celedio: pceledio@mail.sfsu.edu	Team Lead/Frontend Lead
Michael Feger: mfeger@mail.sfsu.edu	Backend Lead
Ahmad Adnan: aadnan@mail.sfsu.edu	Github Master
Rohit Devdhar: rdevdhar@mail.sfsu.edu	Frontend Team Member
Nyan Ye Lin: nlin2@mail.sfsu.edu	Frontend Team Member
Ethan Lunderville: elunderville@sfsu.edu	Backend Team Member

Milestone 4

December 9, 2022

History Table		
Milestone	Date Submitted	Date Revised
Milestone 01	October 08, 2022	October 12, 2022
Milestone 02	October 29, 2022	October 31, 2022
Milestone 03	November 16, 2022	November 21, 2022
Milestone 04	December 09, 2022	

Table of Contents

1. Product Summary

Name of product:

MarketGator

Description of Product:

The purpose of MarketGator is to connect the San Francisco State University community and create a platform where SFSU students and faculty can buy, sell, or share for free digital media. This platform shall facilitate the process of gathering items which shall range from obtaining textbooks, finding media resources that can be used for university projects, or to share created content among students and faculty. Our technology shall empower the SFSU students and faculty to pursue their academic and personal goals by providing an e-commerce platform that will undermine the old ways of purchasing works at high prices from online vendors who charge at a premium price.

Itemized list of all majors committed functions:

1. Registered users shall inherit all the access that unregistered users have.
2. Unregistered users shall be able to create a new account with valid SFSU email.
3. Unregistered users shall be able to view listings
4. Unregistered users shall be able to search listings
5. Registered users shall be able to post items for buying or selling.
6. Registered users shall be able to log in with valid SFSU emails.
7. Registered users shall be able to view their posts through the dashboard.
8. Registered users shall be able to view their messages through the dashboard.
9. Registered users shall be able to inquire about the items by sending a message to the sellers.
10. Registered users shall be able to receive responses from the sellers.
11. Administrators shall be able to log in with valid SFSU emails.
12. Administrators shall inherit all the access of registered users.

13. Administrators shall be able to review the listings from the registered users who are currently advertising a digital item for sale.
14. Administrators shall be able to deny listings that violate the rules and policies of the MarketGator before they are published.
15. Administrators shall be able to ban the registered users who violate the rules and policies of the website.
16. Unregistered users shall be able to view the details of the listings such as price, pictures of the digital media, description of the digital media, classes and professors related to the digital media, ratings, and the reviews of sellers.
17. Unregistered users shall be able to sort and filter the items and the listings based on categories and classes etc.

Unique feature of product:

A feature that makes MarketGator unique from competitors is its straightforward user interface.

URL to MarketGator:

Copy and paste in URL field of web browser:

http://54.218.4.54:3000/

2. Usability Test Plan

Test objectives:

For the usability test plan, we shall focus on item listing posting by the user. The goal of this usability test plan is to evaluate how feasible posting items for viewing on MarketGator is to use by the user through metrics such as effectiveness, efficiency, and satisfaction. Our target audience ranges from young adults to senior adults who work in academia. We want to ensure users of MarketGator that they can achieve the goals that they seek from the product in a manner that requires the least resources spent by them. And ultimately provide a comfortable user experience which maximizes the satisfaction of the user.

Test background and setup:

For the usability test plan, the user will enter the MarketGator site using the latest version of Google Chrome, which is 108.0.5359.98 as of December 8, 2022. This will ensure that MarketGator is being tested by the user on the latest and most popular cross-platform web browser and that the results coming from this usability test is the most accurate and most up-to-date.

Users will start already logged in to a registered MarketGator account. This eliminates any chances of the users running into any issues related to user registration and login, and saves time for the user testing the posting of items feature. However, each test user will have their own separate registered account in order to keep track of which user posted which listing.

The intended users of this usability test will be adults who study or work at San Francisco State University. The reason why these are our intended users is because MarketGator is specifically crafted for the SFSU community.

Usability task description:

Please create a listing indicating an item to advertise on MarketGator.

Evaluation of effectiveness:

To measure effectiveness, will make sure that the available functions allow the completion of creating a listing. Then we will measure what percentage of people completed the task in under 1 minute.

Evaluation of efficiency:

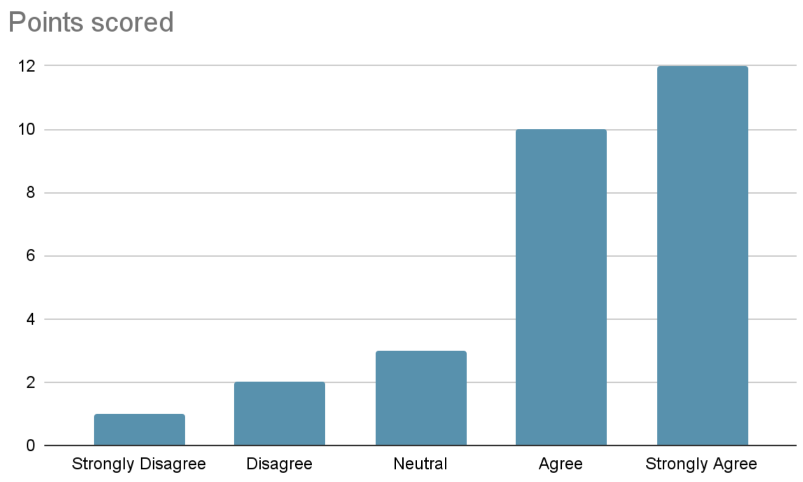
We will measure efficiency based on how many clicks it takes to reach "create listing" and the total time it takes from the dashboard to a successful listing post.

Evaluation of user satisfaction:

	Strongly disagree	Disagree	Neutral	Agree	Strongly Agree
The process of finding "Create Listing" is straightforward					

Your comments are appreciated to further improve our product
Enter Comment Here..

Percentage of users who agree or strongly agree	Average among all users	Standard deviation from average

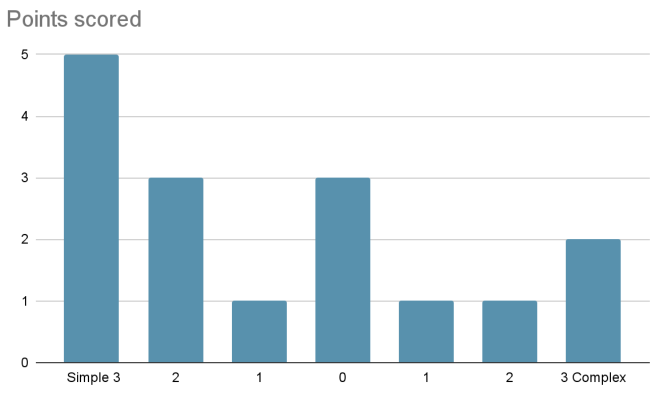


I found data entry into the input variables (title, description, etc) to be:

Simple	3	2	1	0	1	2	3	Confusing

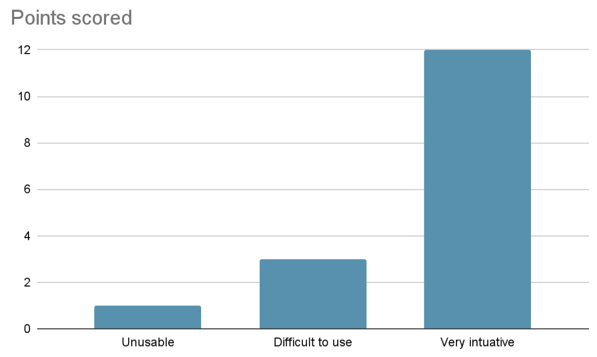
Your comments are appreciated to further improve our product
Enter Comment Here..

Percentage of users who selected towards simple	Average among all users	Standard deviation from average



	Unusable	Difficult to use	Very Intuative
Rate the look and feel of the listing creation page			

Percentage of users who selected Very Intuative	Average among all users	Standard deviation from average



3. QA Test Plan

Test objectives: To identify areas of improvement in the user experience of MarketGator
HW and SW setup:

Processor: Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz   2.59 GHz
Operating system: Windows 10 version 21H2
Browser: Google Chrome Version 108.0.5359.98 (Official Build) (64-bit)
Product URL: http://54.218.4.54:3000/create-listing

Feature to be tested: Listing post
QA Test plan:

Use case: User is logged in and creates a listing

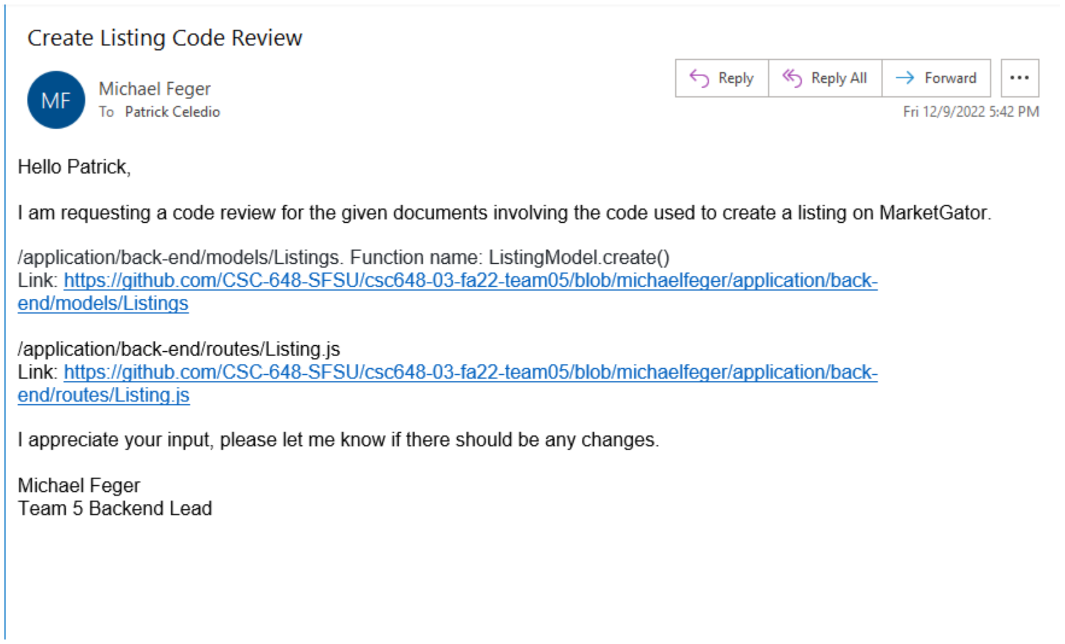
Number	Description	Test Input	Expected output	Pass/Fail
1	Create listing post	Title: test1 Category: Book Item Description: Test1 of listing post Price: 20 Upload photo: png/jpg file	User is redirected to their new listing page	Pass

Number	Description	Test Input	Expected output	Pass/Fail
2	Check database for posted listing that has just been posted	select * from listing WHERE listing.title="test1"	Listing appears in database with all fields correctly set from test case 1. A photopath is created to the /images/products destination and a thumbnail is created in /images/products/thumbnails	Pass

Number	Description	Test Input	Expected output	Pass/Fail
3	Check if item is NOT live	select listing.live from listing WHERE listing.title="test1"	Live: 0	Pass

Number	Description	Test Input	Expected output	Pass/Fail
4	Create listing post	Title: test2 Category: Book Item Description: Test1 of listing post Price: 20 Upload photo: Pdf file	Only images are allowed	Pass

4. Code Review



Create Listing Code Review

Michael Feger
To Patrick Celedio

Fri 12/9/2022 5:42 PM

Hello Patrick,

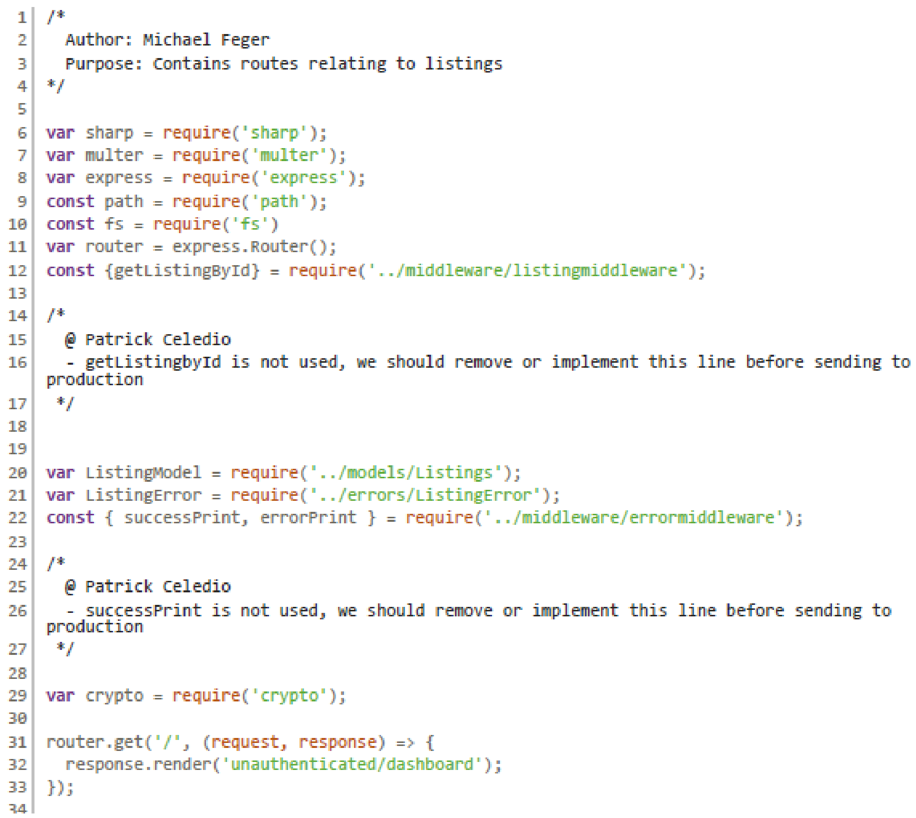
I am requesting a code review for the given documents involving the code used to create a listing on MarketGator.

/application/back-end/models/Listings. Function name: ListingModel.create()
Link: https://github.com/CSC-648-SFSU/csc648-03-fa22-team05/blob/michaelfeger/application/back-end/models/Listings

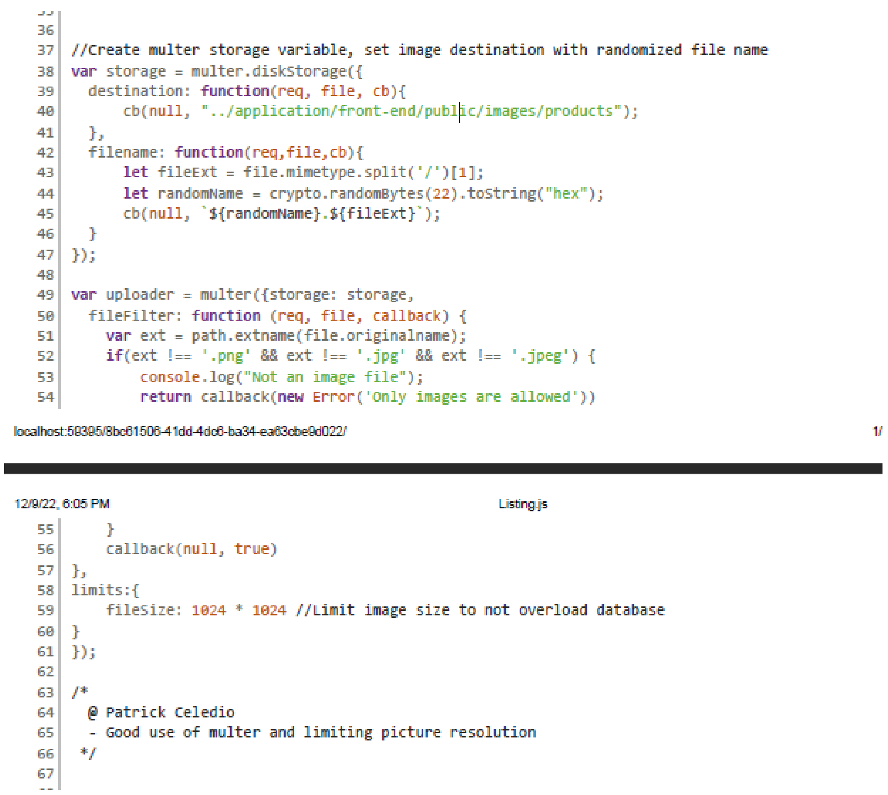
/application/back-end/routes/Listing.js
Link: https://github.com/CSC-648-SFSU/csc648-03-fa22-team05/blob/michaelfeger/application/back-end/routes/Listing.js

I appreciate your input, please let me know if there should be any changes.

Michael Feger
Team 5 Backend Lead



```
/*
  Author: Michael Feger
  Purpose: Contains routes relating to listings
*/

var sharp = require('sharp');
var multer = require('multer');
var express = require('express');
const path = require('path');
const fs = require('fs')
var router = express.Router();
const {getListingById} = require('../middleware/listingmiddleware');

/*
  @ Patrick Celedio
  - getListingbyId is not used, we should remove or implement this line before sending to production
 */


var ListingModel = require('../models/Listings');
var ListingError = require('../errors/ListingError');
const { successPrint, errorPrint } = require('../middleware/errormiddleware');

/*
  @ Patrick Celedio
  - successPrint is not used, we should remove or implement this line before sending to production
 */

var crypto = require('crypto');

router.get('/', (request, response) => {
  response.render('unauthenticated/dashboard');
});

```



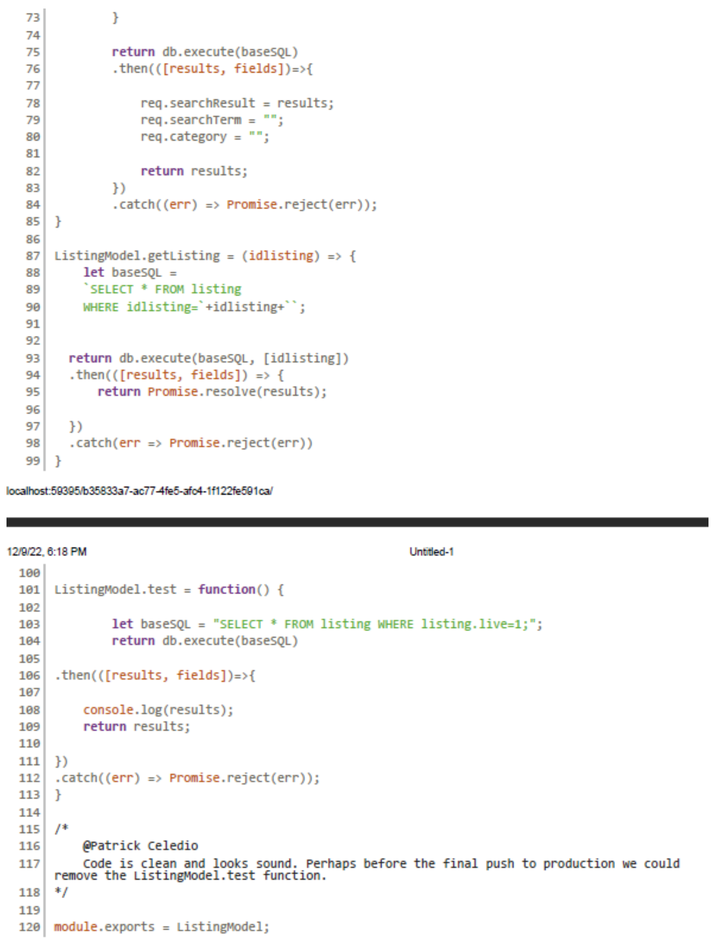
```
//Create multer storage variable, set image destination with randomized file name
var storage = multer.diskStorage({
  destination: function(req, file, cb){
      cb(null, "../application/front-end/public/images/products");
  },
  filename: function(req,file,cb){
      let fileExt = file.mimetype.split('/')[1];
      let randomName = crypto.randomBytes(22).toString("hex");
      cb(null, `${randomName}.${fileExt}`);
  }
});

var uploader = multer({storage: storage,
  fileFilter: function (req, file, callback) {
    var ext = path.extname(file.originalname);
    if(ext !== '.png' && ext !== '.jpg' && ext !== '.jpeg') {
        console.log("Not an image file");
        return callback(new Error('Only images are allowed'))
```

localhost:59395/8bc61506-41dd-4dc6-ba34-ea63cbe9d022/ 1/

12/9/22, 6:05 PM Listing.js

```
    }
    callback(null, true)
},
limits:{
    fileSize: 1024 * 1024 //Limit image size to not overload database
}
});

/*
  @ Patrick Celedio
  - Good use of multer and limiting picture resolution
 */
```

Listing.js code review



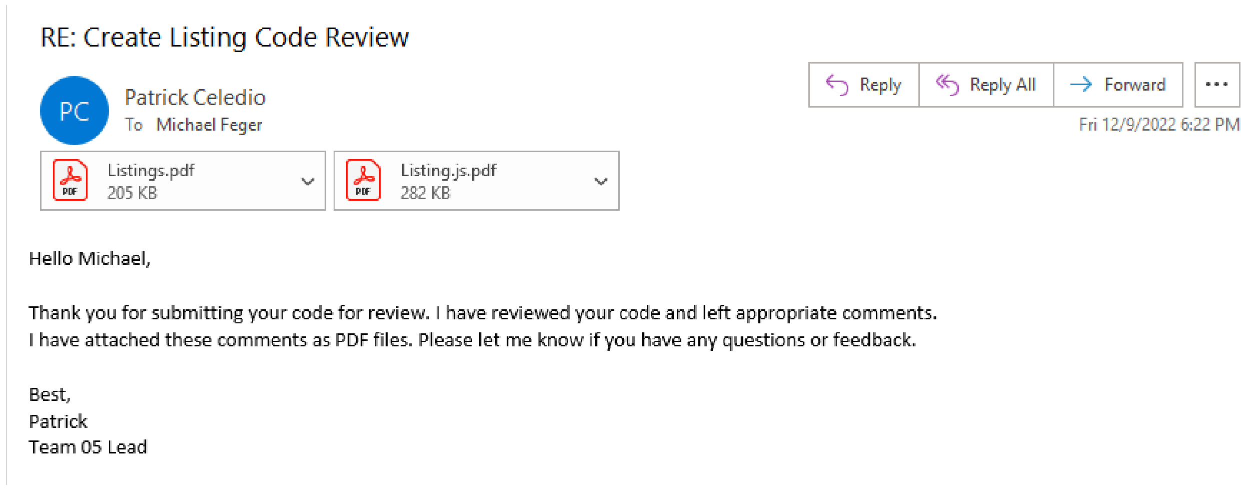
```
        }

        return db.execute(baseSQL)
        .then(([results, fields])=>{

            req.searchResult = results;
            req.searchTerm = "";
            req.category = "";

            return results;
        })
        .catch((err) => Promise.reject(err));
}

ListingModel.getListing = (idlisting) => {
    let baseSQL =
    `SELECT * FROM listing
    WHERE idlisting=`+idlisting+``;


  return db.execute(baseSQL, [idlisting])
  .then(([results, fields]) => {
      return Promise.resolve(results);

  })
  .catch(err => Promise.reject(err))
}
```

localhost:59395/b35833a7-ac77-4fe5-afo4-1f122fe591ca/

12/9/22, 6:18 PM Untitled-1

```

ListingModel.test = function() {

        let baseSQL = "SELECT * FROM listing WHERE listing.live=1;";
        return db.execute(baseSQL)

.then(([results, fields])=>{

    console.log(results);
    return results;

})
.catch((err) => Promise.reject(err));
}

/*
    @Patrick Celedio
    Code is clean and looks sound. Perhaps before the final push to production we could
remove the ListingModel.test function.
*/

module.exports = ListingModel;
```

Listings; SQL code review



RE: Create Listing Code Review

PC Patrick Celedio
To Michael Feger

Reply | Reply All | Forward | ...

Fri 12/9/2022 6:22 PM

Listings.pdf 205 KB
Listing.js.pdf 282 KB

Hello Michael,

Thank you for submitting your code for review. I have reviewed your code and left appropriate comments. I have attached these comments as PDF files. Please let me know if you have any questions or feedback.

Best,
Patrick
Team 05 Lead

5. Self-Check on Best Practices for Security

Asset to be protected	**Types of possible/expected attacks**	**Strategy to mitigate/protect the asset**
User images	Illegal content, nudity, or images irrelevant to the marketplace	Admins must approve listing posts using the listing.live variable. Listings will not be displayed to users until manually approved by admin.
User passwords	Database compromised by hacker	Passwords are encrypted inside the program and then inserted into database.
Database	SQL injection	Search function will not allow more than 40 alphanum chars. Search results and user login/registration are validated and not inserted into the database through template strings.

6. Self-check of the adherence to original Non-functional specs – performed by team leads

• MySQL setup and development knowledge	DONE
• Accommodating to team members have different schedules from each other	DONE
• Efficient storage of MarketGator listing images	DONE
• Efficient speed of retrieving images and search results	DONE
• Site and UI compatibility of MarketGator when the web browser is resized	DONE