SW Engineering CSC648/848 Section 01 Fall 2016

Department of Computer Science

SAN FRANCISCO STATE UNIVERSITY

DAVIDSLIST

Milestone 4

Beta Launch

Team 1

Nora Schinkel, Team Lead | ncschinkel@gmail.com

David Chau, Technical Lead

Paul Derugin

Imran Irfan

Yansen Liu

Steven Nguyen

Submitted to—

Dr. Dragutin Petkovic

Department of Computer Science, San Francisco State University

Document Version: 1.1

Last Modified: 2016 December 11

Contents

1. Product Summary

In this report we present **DavidsList**, an apartment rental web service for students. With property taxes increasing every year, many Bay Area homeowners would like to rent out their homes and apartments (or simply a bedroom) to students. However, it is difficult for homeowners to make sure the students they meet on rental websites are actual students. This is where DavidsList comes in. We provide a service where students can be verified by their university email address, thus creating a safe and trustworthy marketplace for landlords and students.

DavidsList provides the following features:

- Guest users can browse the website, but cannot post listings or message landlords.
- Everybody can create an account on DavidsList and everybody can read the Terms and Conditions and the Privacy Policy.
- Everybody is able to use search for listings. Search results may be filtered based on number of bedrooms, price, and distance to campus.
- Registered users can log in with their account info. Once logged in, they can create, edit, and delete listings. In creating a listing at least one picture of the listing must be uploaded, accompanied by a title, price, and address. Landlords are given the option to show only an approximate address on the website; this means that only the ZIP code of the listing will be shown to other users.
- Registered users seeking an apartment can communicate with the landlord of a listing by clicking on the "Contact" button on the Listing Detail page of a listing. Registered users can view and reply to messages through the All Messages page under My Account.
- Administrators can remove users and listings via Workbench.

DavidsList is now live and can be found via the following URL:

http://sfsuswe.com/~f16g01/DavidsList/

2. Usability Test Plan

2.1 Test Objectives

The major function that we have selected to be tested for usability is the search function. We will mainly perform a “Validation” usability test to ensure that the product is usable. The test objectives are to thoroughly test the search function on DavidsList to get results for apartment listings and ensure that the search function is easy and intuitive to use without any further explanation. The placement of the search bar and input field placeholder text should be enough to guide the user through this process. During the usability test, we hope that the user is able to expose any usability flaws that impede the completion of the tested search functionality.

2.2 Test Plan

The system used for testing must be capable of running the Internet browsers Google Chrome versions 54.0.2840.99 m and newer, Mozilla Firefox versions 50.0.2 and newer, and Microsoft Internet Explorer versions 11.0.9600.18525 and newer. We cannot guarantee that DavidsList will function correctly on any other browsers as they have not been tested. The website can be accessed at the URL http://sfsuswe.com/~f16g01/DavidsList/. Upon reaching DavidsList, users will be greeted with our homepage where they are able to find the search function and featured listings. Valid input for the search query can include an address, city, or ZIP code. The task to be accomplished by the user performing the usability test is to find an apartment in San Francisco. For example, if the user searches for “San Francisco,” they should be presented with all listings that are located in San Francisco. Once the user reaches the search Results page that contains results matching the search criteria, we can conclude that the usability test has been completed successfully.

2.2.1 Task Description

Task	**Description**
Task: Find apartment in San Francisco	Find apartment in San Francisco.
Successful Completion Criteria	Search results obtained.
Benchmark	~10 sec. to complete.

2.3 Questionnaire

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
I was able to find the search function on DavidsList	○	○	○	○	○
I found results related to my search query	○	○	○	○	○
The search function behaved as I expected it to behave	○	○	○	○	○
Additional Comments					

3. QA Test Plan

3.1 Test Objectives

The objective of the QA test is to thoroughly test the search function of DavidsList in order to ensure that search results display apartment listings relevant to the search query. Searching by city, state, and ZIP code will be tested.

3.2 System Setup

Processor: Intel i3-4330 CPU @ 3.50 GHz

RAM: 16 GB

Operating System: Windows 10 Home 64-bit Desktop

Browser 1: Google Chrome Version 54.0.2840.99 m

Browser 2: Microsoft Internet Explorer Version 11.0.9600.18525

3.3 Feature to be Tested

This QA test will test the functionality of the search function from all of DavidsList's major pages: home page, search result page, rent out page, listing detail page and user account page, collectively hereinafter "Testing Pages." The test will test for the following Expected Functionality:

- The search result page shall show all listings that match the search query (the search query being either a city, state, or ZIP code), as well as the number of found listings.
- All results should be readable and openable.
- The search query shall not disappear from the search box after searching.
- All listings shall include a listing title, listing price, address, city, state, ZIP code, number of bedrooms, number of bathrooms, an indication of whether the apartment is furnished, and distance from the apartment to SFSU.
- All results shall be the same across both browsers.

3.4 Test Cases

3.4.1 Test 1 – Searching by City

Search for the city "San Francisco" from all of the Testing Pages. Validate that every Expected Functionality is met. If all of the Expected Functionality is met, the test result shall be PASS. Perform the search on both browsers.

3.4.2 Test 2 – Searching by State

Search for the state "CA" from all of the Testing Pages. Validate that every Expected Functionality is met. If all of the Expected Functionality is met, the test result shall be PASS. Perform the search on both browsers.

3.4.3 Test 3 – Searching by ZIP Code

Search for the ZIP code “94122” from all of the Testing Pages. Validate that every Expected Functionality is met. If all of the Expected Functionality is met, the test result shall be PASS. Perform the search on both browsers.

3.5 Test Results

					Test Results	
Test No.	**Test Title**	**Test Description**	**Test Input**	**Expected Output**	**Browser 1**	**Browser 2**
1	Search by City	Search for “San Francisco” from all Testing Pages	“San Francisco”	All listings in the city of San Francisco (27 total)	PASS	PASS
2	Search by State	Search for “CA” from all Testing Pages	“CA”	All listings in the state of California (36 total)	PASS	PASS
3	Search by ZIP Code	Search for “94122” from all Testing Pages	“94122”	All listings with ZIP code “94122” (6 total)	PASS	PASS

4. Code Review

4.1 Coding Style

We opted to use several standards while developing this application. For our naming conventions, we decided to use meaningful names for our identifiers, and classes. We also decided to name our identifiers using camel case. Lastly we used braces on the same line of use.

4.2 Review

This section contains a sample of our internal code review process. The code reviewed was the back-end of the search function. Below are the email exchange and code snippet relevant to the review.

4.2.1 Email Exchange – Email 1

From: Nora Schinkel
Sent: Monday, December 12, 2016 4:48 PM
To: Paul Derugin
Subject: CSC648 - Code Review - Search Function

Hi Paul,

Attached for your review is my code for the back-end of the search function.

-Nora

1 Attachment

<search.php>

4.2.2 Email Exchange – Email 2

From: Paul Derugin
Sent: Tuesday, December 12, 2016 2:27 PM
To: Nora Schinkel
Subject: Re: CSC648 - Code Review - Search Function

Hi Nora,

Your code looks good overall. It follows our chosen coding style and is mostly pretty easy to follow. I've attached a copy of your code with my comment blocks, which all begin with "`PEER REVIEW COMMENT (PD)`"

-Paul

1 Attachment

<search_code-review-PD.php>

4.2.3 Reviewed Code Snippet

First 117 lines of search_code-review-PD.php

```
<?php

/******************************************************************************
* PEER REVIEW COMMENT (PD)
*   Expand the header to include more attributes such as filename, author, date
*   last modified, and a basic copyright.
******************************************************************************/

/*
 * Controller specifically for searching by address, possibly either returning
 * HTML for the page to render the search or returning JSON for the front-end to
 * do it manually
 */

/******************************************************************************
* PEER REVIEW COMMENT (PD)
*   Your header mentions that this class "either return[s] HTML ... or
*   return[s] JSON." To me this comment leaves too much ambiguity--which does
*   it ultimately return?
******************************************************************************/
class Search extends Controller {

/******************************************************************************
* PEER REVIEW COMMENT (PD)
*   I would add to the below header that this function, in addition to
*   performing a search based on city, also performs search if provided a state
*   or ZIP code.
******************************************************************************/
    /*
     * Creates repositories for address and images
     * Does a database call for cities
     * Loops through the results and finds the first image thumbnails of the cities
     * returns as a json encoded array
     */
    public function searchApartments() {
/******************************************************************************
* PEER REVIEW COMMENT (PD)
*   I see that a comment for this variable, $thresh, is included the next time
*   it is defined (inside the first else statement); howver, I would have found
*   it helpful if the comment was here.
******************************************************************************/
        $thresh = 70;

/******************************************************************************
* PEER REVIEW COMMENT (PD)
*   You have a comment on the next line of code indicating to "change this."
*   Change what to what?
******************************************************************************/
        $searchInput = strip_tags(strtolower($_POST["search"])); // change this
        $addressRepo = RepositoryFactory::createRepository("address");
        $listingImageRepo = RepositoryFactory::createRepository("listing_image");

        $listingRepo = RepositoryFactory::createRepository("listing");
        $listingDetailRepo = RepositoryFactory::createRepository("listing_detail");


        // $addresses = $addressRepo->find($city, "city"); //this is the search line
        if(is_numeric($searchInput)) $addresses = $addressRepo->find($searchInput, "zipcode");

/*******************************************************************************
* PEER REVIEW COMMENT (PD)
*   Good use of comments inside this else statment.
*******************************************************************************/
        else{
            $thresh = 80; // threshold for percentage of similar_text
            $addresses = array(); //prepare array

            // If searchInput ends in street or avenue, reduce
            if(strcmp(substr($searchInput, -6),"street")==0
                  || strcmp(substr($searchInput, -6),"st")==0
                  || strcmp(substr($searchInput, -6),"avenue")==0
                  || strcmp(substr($searchInput, -6),"ave")==0){
                $searchInput = substr($searchInput, 0, -6);
            } elseif (strcmp(substr($searchInput, -2),"st")==0
                  || strcmp(substr($searchInput, -3),"ave")==0) {
                $searchInput = substr($searchInput, 0, -3);
            }


            $addressArray = $addressRepo->fetch();// find other way to fetch all

            foreach($addressArray as $address){

                // Compare search query to city
                $compareCity = strtolower($address->getCity());
                similar_text($compareCity , $searchInput, $percentageCity);
                if($percentageCity > $thresh){
                    $addresses[] = $address;
                    continue;
                }

                // Compare search query to street name
                $compareStreetName = strtolower($address->getStreetName());

                // Check for street and avenue in compareStreetName
                if(strcmp(substr($compareStreetName, -6),"street")==0
                          || strcmp(substr($compareStreetName, -6),"avenue")==0){
                    $compareStreetName = substr($compareStreetName, 0, -6);
                } elseif (strcmp(substr($compareStreetName, -2),"st")==0
                          || strcmp(substr($compareStreetName, -3),"ave")==0) {
                    $compareStreetName = substr($compareStreetName, 0, -3);
                }

                // check for house number and remove
                $compareStreetNameNoNumbers =
                        trim(str_replace(range(0,9),'',$compareStreetName));

                similar_text($compareStreetName , $searchInput, $percentageStreetName);
                similar_text($compareStreetNameNoNumbers ,
                          $searchInput, $percentageStreetNameNoNumbers);
                if($percentageStreetName > $thresh
                          || $percentageStreetNameNoNumbers > $thresh){
                    $addresses[] = $address;
                    continue;
                }
                ...
```

5. Self-Check on Best Practices for Security

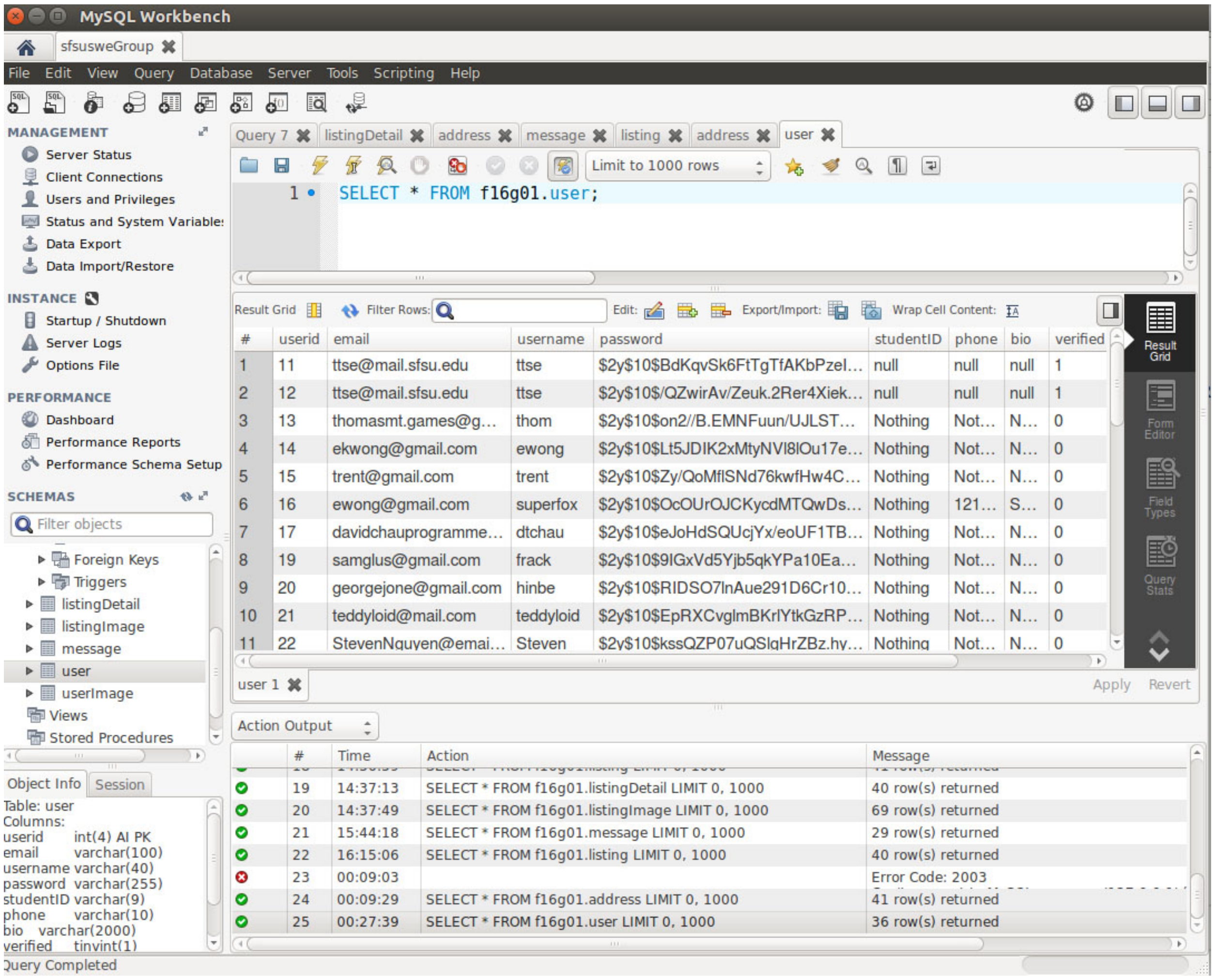
5.1 Major Assets Being Protected

The following user information is being protected:

- Name
- Password
- Email address
- Listing address

5.2 Encryption of Passwords in the Database

User passwords are encrypted before being sent to the database. Below is a screenshot showing the encrypted user passwords in the database:



```
SELECT * FROM f16g01.user;
```

#	userid	email	username	password	studentID	phone	bio	verified
1	11	ttse@mail.sfsu.edu	ttse	$2y$10$BdKqvSk6FtTgTfAKbPzel…	null	null	null	1
2	12	ttse@mail.sfsu.edu	ttse	$2y$10$/QZwirAv/Zeuk.2Rer4Xiek…	null	null	null	1
3	13	thomasmt.games@g…	thom	$2y$10$on2//B.EMNFuun/UJLST…	Nothing	Not…	N…	0
4	14	ekwong@gmail.com	ewong	$2y$10$Lt5JDIK2xMtyNVI8lOu17e…	Nothing	Not…	N…	0
5	15	trent@gmail.com	trent	$2y$10$Zy/QoMflSNd76kwfHw4C…	Nothing	Not…	N…	0
6	16	ewong@gmail.com	superfox	$2y$10$OcOUrOJCKycdMTQwDs…	Nothing	121…	S…	0
7	17	davidchauprogramme…	dtchau	$2y$10$eJoHdSQUcjYx/eoUF1TB…	Nothing	Not…	N…	0
8	19	samglus@gmail.com	frack	$2y$10$9IGxVd5Yjb5qkYPa10Ea…	Nothing	Not…	N…	0
9	20	georgejone@gmail.com	hinbe	$2y$10$RIDSO7lnAue291D6Cr10…	Nothing	Not…	N…	0
10	21	teddyloid@mail.com	teddyloid	$2y$10$EpRXCvglmBKrlYtkGzRP…	Nothing	Not…	N…	0
11	22	StevenNguyen@emai…	Steven	$2y$10$kssQZP07uQSlgHrZBz.hy…	Nothing	Not…	N…	0

	#	Time	Action	Message
✓	19	14:37:13	SELECT * FROM f16g01.listingDetail LIMIT 0, 1000	40 row(s) returned
✓	20	14:37:49	SELECT * FROM f16g01.listingImage LIMIT 0, 1000	69 row(s) returned
✓	21	15:44:18	SELECT * FROM f16g01.message LIMIT 0, 1000	29 row(s) returned
✓	22	16:15:06	SELECT * FROM f16g01.listing LIMIT 0, 1000	40 row(s) returned
✗	23	00:09:03		Error Code: 2003
✓	24	00:09:29	SELECT * FROM f16g01.address LIMIT 0, 1000	41 row(s) returned
✓	25	00:27:39	SELECT * FROM f16g01.user LIMIT 0, 1000	36 row(s) returned

```
Table: user
Columns:
userid      int(4) AI PK
email       varchar(100)
username varchar(40)
password  varchar(255)
studentID varchar(9)
phone       varchar(10)
bio   varchar(2000)
verified    tinyint(1)
```

5.3 Input Data Validation

The following user input is checked for validation on the front-end using jQuery Validation Plugin v1.15.0:

- Listing details:
    - Price
    - Address (whether it is provided; not whether it is a valid address)
    - Whether or not the full address should be revealed
    - Number of bedrooms and baths
    - Listing Type
- Search query input
    - For example, if a user attempts to search by ZIP code but accidentally enters six digits instead of five, he is prompted with a message that asks him to enter a valid ZIP code.

Login credentials are checked for validation on the back-end in the Users controller using calls to the database.

6. Adherence to Original Non-Functional Specs

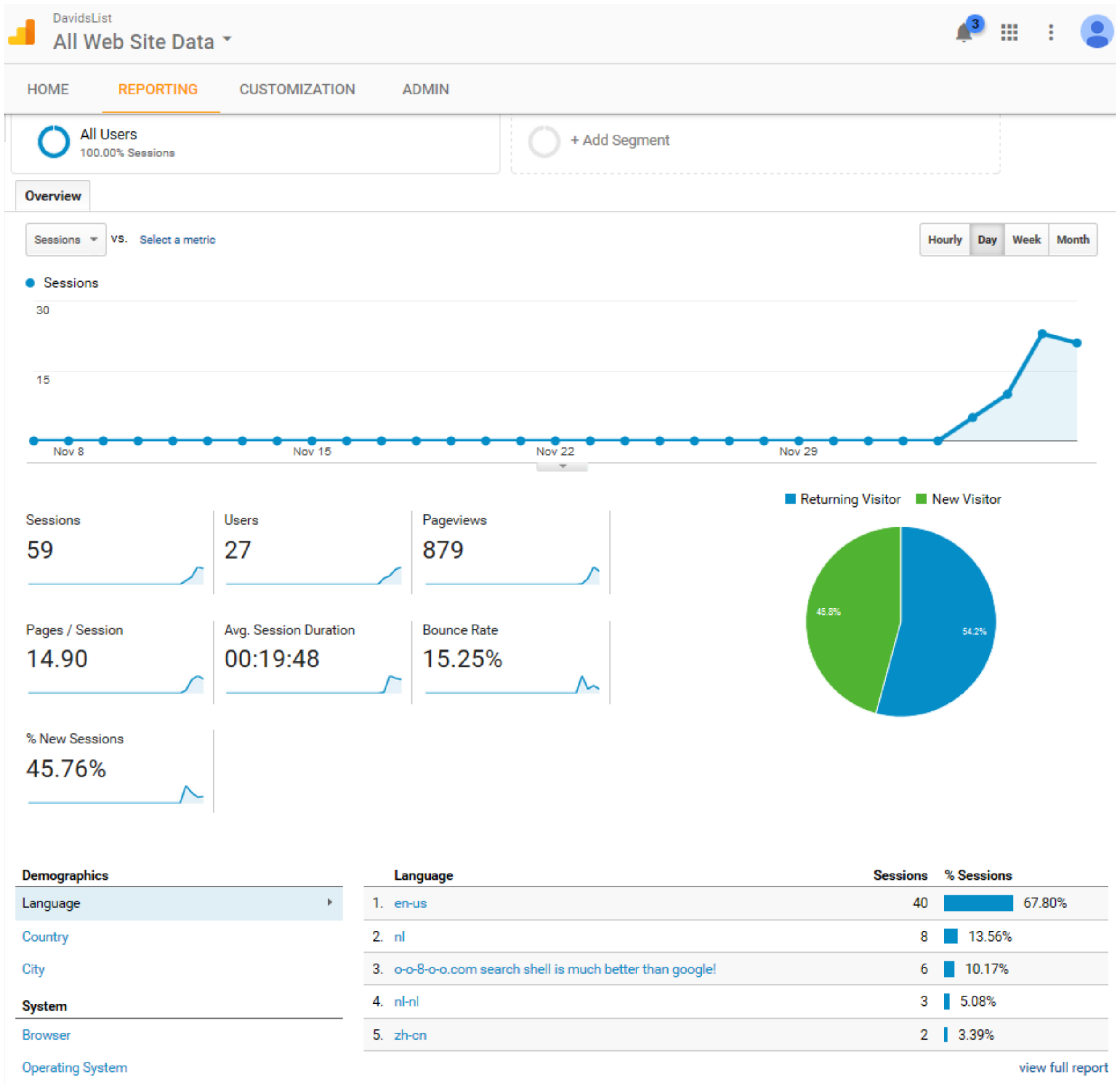
Below is the status of all original non-functional specs:

	Non-Functional Spec	Status
1	Application shall be developed using class provided LAMP stack	DONE
2	Application shall be developed using pre-approved set of SW development and collaborative tools provided in the class. Any other tools or frameworks shall be explicitly approved by Marc Sosnick on a case by case basis.	DONE
3	Application shall be hosted and deployed on Amazon Web Services as specified in the class.	DONE
4	Application shall be optimized for standard desktop/laptop browsers, and shall render correctly on the two latest versions of all major browsers: Mozilla, Safari, Chrome. It shall degrade nicely for different sized windows using class approved programming technology and frameworks so it can be adequately rendered on mobile devices	DONE
5	Data shall be stored in the MySQL database on the class server in the team's account	DONE
6	Application shall be served from the team's account	DONE
7	No more than 50 concurrent users shall be accessing the application at any time	DONE
8	Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users	DONE
9	The language used shall be English.	DONE
10	Application shall be very easy to use and intuitive. No prior training shall be required to use the website.	DONE
11	Google analytics shall be added for major site function	DONE

12	Messaging between users shall be done only by class approved methods to avoid issues of security with e-mail services	DONE
13	Pay functionality (how to pay for goods and services) shall be simulated with proper UI, no backend	REMOVED BY MANAGEMENT
14	Site security: basic best practices shall be applied (as covered in class)	DONE
15	Modern SE processes and practices shall be used as specified in the class, including collaborative	DONE
16	The website shall prominently display the following text on all pages “SFSU/FAU/Fulda Software Engineering Project, Fall 2016. For Demonstration Only”. (Important so as to not confuse this with a real application)	DONE

7. Google Analytics

DavidsList uses Google Analytics to keep track of website usage. Below is a screenshot of Google Analytics showing the usage of DavidsList's homepage:



Sessions	59
Users	27
Pageviews	879
Pages / Session	14.90
Avg. Session Duration	00:19:48
Bounce Rate	15.25%
% New Sessions	45.76%

	Language	Sessions	% Sessions
1.	en-us	40	67.80%
2.	nl	8	13.56%
3.	o-o-8-o-o.com search shell is much better than google!	6	10.17%
4.	nl-nl	3	5.08%
5.	zh-cn	2	3.39%