SW Engineering CSC648/848 Section 01 Fall2016

Gator Lodge

Team 7

Lijie Zhou (Team Lead)
Anna Sever(Tech Lead)
Theofanis Koutoulas(Front End)
Ivan Marchenko(Full Stack)
Logan Figgins(Back End)
Hao Xian Zheng(Back end)

"Milestone 4"

12/09/16 Version: 01
12/11/16 Version: 02

1) Product summary

GatorLodge is a secure and user-friendly web app that helps SFSU students rent housing. Our app has the following features:

- Users can view listings without registration. If users are interested in posting their apartment or contact the landlord, they can then register and log in.
- Registered users can post their apartment on GatorLodge, together with the apartment description, address, images, etc.
- Registered users can send and receive messages from other registered users on GatorLodge.
- All users are able to see the approximate location on Google map directly on GatorLodge. Users are also able to see the distance from the location to SFSU.
- Registered users have a personalized page which lists all the apartments he saved.

User-friendly design and security makes our application stand out of similar product on the market. Our app's interface works on different devices (e.g: laptop, iPad, cellphone, etc) and is compatible for different web browsers. Further, we follow the best design principle to make sure users have an intuitive experience. Our app also cares about user privacy and data security. Our app uses HTTPS. We also use JQuery validation to sanitize data from users.

Our app can be found at https://www.sfsuswe.com/~f16g07/

Select ONE major function (NOT login or registration) to be tested for usability. We recommend search or data upload.

Test Objectives:

The objectives of testing the search function are for the user to be able to find an apartment to rent within their criteria. The search function should be easy to use and shouldn't leave the user lost or confused with functionality or looking for functionality. An SFSU student shall be able to use GatorLodge search to find a place to live. Should the student be on a budget they should be able to narrow results by price and also use other filters to help aid in their apartment search. The student shall also be able to find the search function easily without question, and be prompted with details of what to search for. (A City, Area Code, or Address).

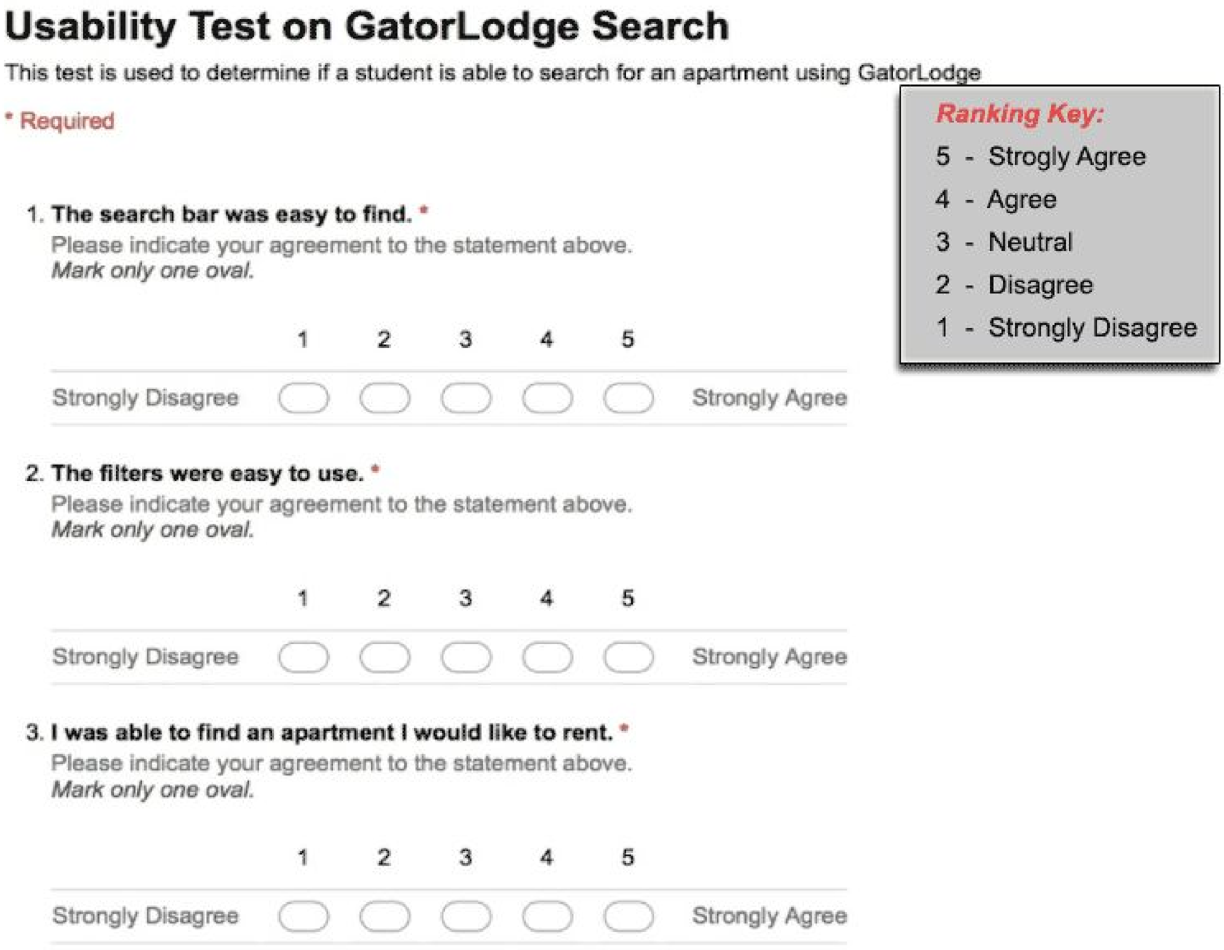
Test plan:

The system will be presented to the user while at the homepage. In order to isolate testing to the search function the system will not require the user to register or log in. This will make testing for both a guest user and a registered user possible in order to cover a wider array of use cases as a guest user is equally as able to browse apartments as a registered user.

A SFSU student is the intended user for the search usability test and will be tasked with searching for an apartment using GatorLodge. A student will have basic technology skills and will be familiar with search functions such as Google. The intended user will be looking for a place to live during their education period. The user will have to find an apartment they would like to rent. GatorLodge provides filters to aid the user in their search by narrowing the results. The system should provide enough feedback and remain responsive enough for the user to quickly and painlessly find their desired future apartment. The student shall be able to find the apartment without being confused or be forced to ask any questions. It shall be simple and straightforward for the user. The user shall be able to find an apartment in a timely fashion as to fit the criteria of the website being simple for the user.

The user will begin on the homepage and shall be asked to find an apartment. The user will use the homepage to set their search location. The system will then take the user to a search results page and populate apartments for the user to browse. These results will be shown in a very organized way as to be visually pleasing for the user. Should the user have a budget they will have the option to filter results based on a price range. The student shall also be granted the option to select through multiple filters to make the right choice for their apartment. The user shall also feel more informed on the location they picked, giving them the feel of accomplishment during their test. To consider the test complete the user will have to select an apartment they are interested in renting.

URL of System to be tested: https://sfsuswe.com/~f16g07/team07/proto



Usability Test on GatorLodge Search

This test is used to determine if a student is able to search for an apartment using GatorLodge

* Required

Ranking Key:

5 - Strogly Agree
4 - Agree
3 - Neutral
2 - Disagree
1 - Strongly Disagree

1. **The search bar was easy to find.** *
   Please indicate your agreement to the statement above.
   Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

2. **The filters were easy to use.** *
   Please indicate your agreement to the statement above.
   Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

3. **I was able to find an apartment I would like to rent.** *
   Please indicate your agreement to the statement above.
   Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

Figure 1 Usability Test

3) QA Test Plan

Test Objectives:

Objective is to verify that search feature functions properly and conforms to functional requirements outlined in the feature specification document. This feature is a must have component of the application. QA must certify this feature in order to move on with our next production release. Criteria for successful certification: no blocker, critical, major bugs.

Hardware Setup	The entire application needs to be in sync with latest code from repository and deployed on our AWS. DB resides on the same server, already configured, no setup overhead. On the client side, need PC with latest versions of Chrome and Firefox installed.
Software Setup	The entire application needs to be in sync with latest code from repository, development branch. Also, ensure that DB schema is up to date.
Test Environment	Staging environment: https://sfsuswe.com/~lfiggins/team07/proto Data: use existing sample data stored in db
Projected Schedule & Scope of Work	During the final two weeks of the semester, each team member will perform comprehensive manual testing of application components they're responsible for. Point of contacts (POCs) can be found below in the Test Report Summary section. Due to lack of time and resources we are not using automated testing methodologies. Each team member is responsible to perform adequate smoke and regression testing on their components.
Risks & Contingencies	AWS server may go offline as happened earlier. Need to set up alternative QA environment to avoid interruptions.
Detailed Requirements	User must be able to click into search bar, type search text, run search, and that will display search results page with relevant rental listings drawn from DB. The look and feel of the UI must conform to provided design wires. Upon arrival onto search results page, user must be able to filter listings using available filtering configurations. Need to perform positive test by providing a search term corresponding to N records in DB, search results page should then display precisely all N instances, not one less, not one more. Also, need to perform a negative test by providing search terms that are not present in DB. This test must return empty search results.

Test Cases:

#	Title	Description	Expected	Actual	Result
1	Performs search	Enter valid zip code, click 'search' button.	For zip code 94121 exactly 3 items should display in search results page.	Works as expected. Observed 3 items.	P
2	UI look and	Validate that UI is in	Should comply and	Observed UI	P

	feel	compliance with wires.	match visual specs.	as expected.	
3	Security compliance	Ensure that search input will ignore SQL and JavaScript XSS injections.	Enter into search field: <script>alert('foo')</script> and perform search. Verify that javascript popup does not show.	Test passed. No popup shown.	P

(Testing results executed on 2 browsers edge version (Chrome and Firefox) are identical.)

Test Report Summary:

Snapshot (hypothetical sample):

Overall Status : **GREEN**
Overall Progress : **70% complete**
Outlook Forecast : **POSITIVE**

Risks:

1. We anticipate last minute refactorings, hence risk of regression issues.
2. Flu season have arrived, people may become ill.

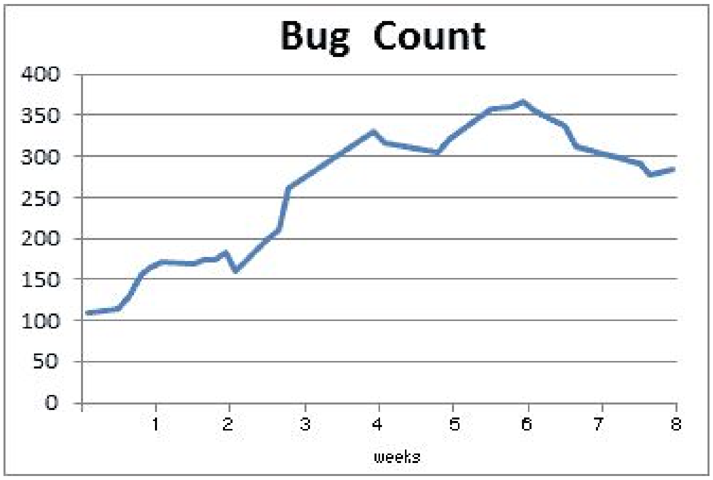


Figure 2 Bug Count

Unresolved Bug Count (hypothetical sample):

Status \ Type	Blocker	Critical	Major	Normal	Minor	Defer	Total
New	1	4	19	94	16	9	**143**
Assigned	1	19	27	20	3	13	**83**
Reopened	0	6	17	19	3	14	**59**
Total	**2**	**29**	**63**	**133**	**22**	**36**	**285**

4) Code Review:

The source code for the Gator Lodge adheres to a coding style similar to that suggested by the php Framework Interoperability Group, formally known as the php Standards Group. Most of the recommendations provided in the PSR-1 and PSR-2 were followed. Class names use the studly caps convention of the form SomeClassName. Method names use camel case and any variable declarations use snake case. Furthermore, method and class blocks curly braces being on the next line in order to improve code readability.

Here below is the email from reviewer to the engineer who coded the search function. The review is given within the code in the commenting format.

Hi xx,

Great job finishing the search function. Having the function work is always the most important thing!

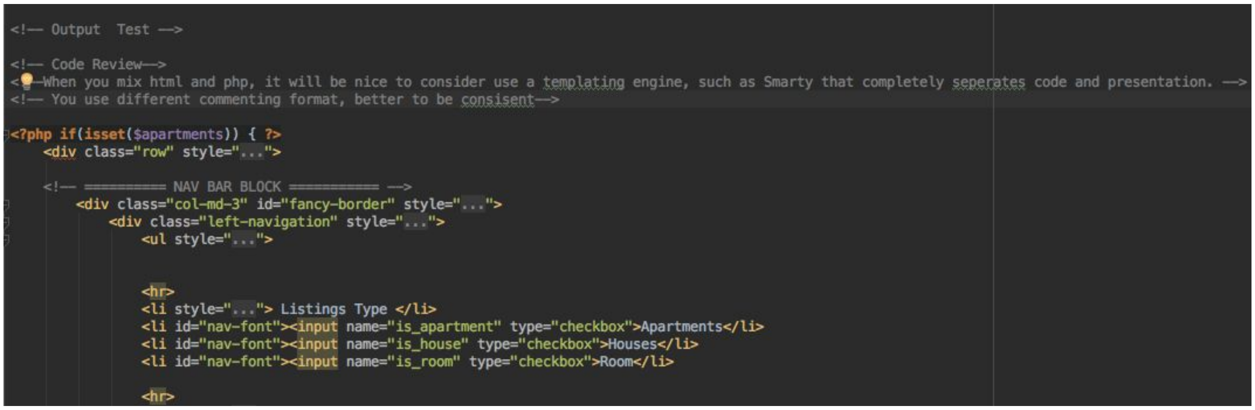
In order to help you follow the best practice and improve, I'm giving you the review below. I can see you did very well in the following areas:

- *Make variable names clear and follow PHP format.*
- *Follow PHP best practice for function/class block formatting.*

Here are a couple of things I think you could improve:

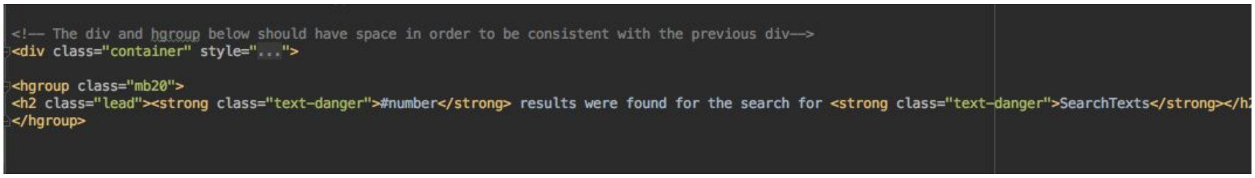
- *Try to use Smarty (a templating engine) to seperate HTML and PHP.*
- *Try to pay attention on HTML block spacing.*
- *Try to use the same commenting format.*

I also included the code snippet that illustrate my points above with review inside. Please take a look and let me know how you think. I hope this helps.



```
<!-- Output  Test -->

<!-- Code Review-->
<!--When you mix html and php, it will be nice to consider use a templating engine, such as Smarty that completely seperates code and presentation. -->
<!-- You use different commenting format, better to be consisent-->

<?php if(isset($apartments)) { ?>
    <div class="row" style="...">

    <!-- ========== NAV BAR BLOCK =========== -->
        <div class="col-md-3" id="fancy-border" style="...">
            <div class="left-navigation" style="...">
                <ul style="...">


                <hr>
                <li style="..."> Listings Type </li>
                <li id="nav-font"><input name="is_apartment" type="checkbox">Apartments</li>
                <li id="nav-font"><input name="is_house" type="checkbox">Houses</li>
                <li id="nav-font"><input name="is_room" type="checkbox">Room</li>

                <hr>
```

Figure 3 Inconsistent commenting style; PHP HTML mix



```
<!-- The div and hgroup below should have space in order to be consistent with the previous div-->
<div class="container" style="...">

<hgroup class="mb20">
<h2 class="lead"><strong class="text-danger">#number</strong> results were found for the search for <strong class="text-danger">SearchTexts</strong></h
</hgroup>
```

Figure 4 Spacing issue in HTML, inconsistent with previous block

5) Self-check on best practices for security

Secured Assets:

- User Contact Information
- User Account Password

Confirm that you encrypt PW in the DB: confirmed

Confirm Input data validation (list what is being checked and how)

- Client side plugins for form validation checks for correctly formulated emails, names, and phone numbers
- Server side form validation is done with the use of prepared sql statements. By using standard PDO the system is protected from sql injection.

1. Use HTTPS to encrypt network data: **DONE**
2. Hash user password before storing in the database: **DONE**
3. Use data validation to check/sanitize user input: **DONE**
4. Use approximate location on Google map to protect user privacy:

6) Adherence to original Non-functional specs

List all original non-functional specs as in high level application document published at the very beginning of the class

For each say DONE if it is done (which is expected and required); ON TRACK if it is being done and you are sure it will be completed on time; or ISSUE and then explain it. Note: you must adhere to all original non-functional specs as published in the original high level specification document. Failure to do so may cause reduced grade

1. System shall respond visually within 5 seconds. **DONE**

2. File size in no time shall exceed 2 Mbytes. **DONE**

3. Users with high-school diploma, after 1 hour training, shall complete the task in 5 minutes with no more than 2 errors. **ON TRACK**

4. Each WWW page shall have official company logo in upper left corner. **DONE**

5. The following user data shall be collected and stored; the data shall be used ONLY for record keeping. **DONE**

6. Application shall be developed using class provided LAMP stack. **DONE**

7. 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**

8. Application shall be hosted and deployed on Amazon Web Services as specified in the class. **DONE**

9. 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. **ON TRACK**

10. Data shall be stored in the MySQL database on the class server in the team's account. **DONE**

11. Application shall be served from the team's account. **DONE**

12. No more than 50 concurrent users shall be accessing the application at any time. **DONE**

13. Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users. **ON TRACK**

14. The language used shall be English. **DONE**

15. Application shall be very easy to use and intuitive. No prior training shall be required to use the website. **ON TRACK**

16. Google analytics shall be added for major site functions. **ON TRACK**

17. Messaging between users shall be done only by class approved methods to avoid issues of security with e-mail services. **ON TRACK**

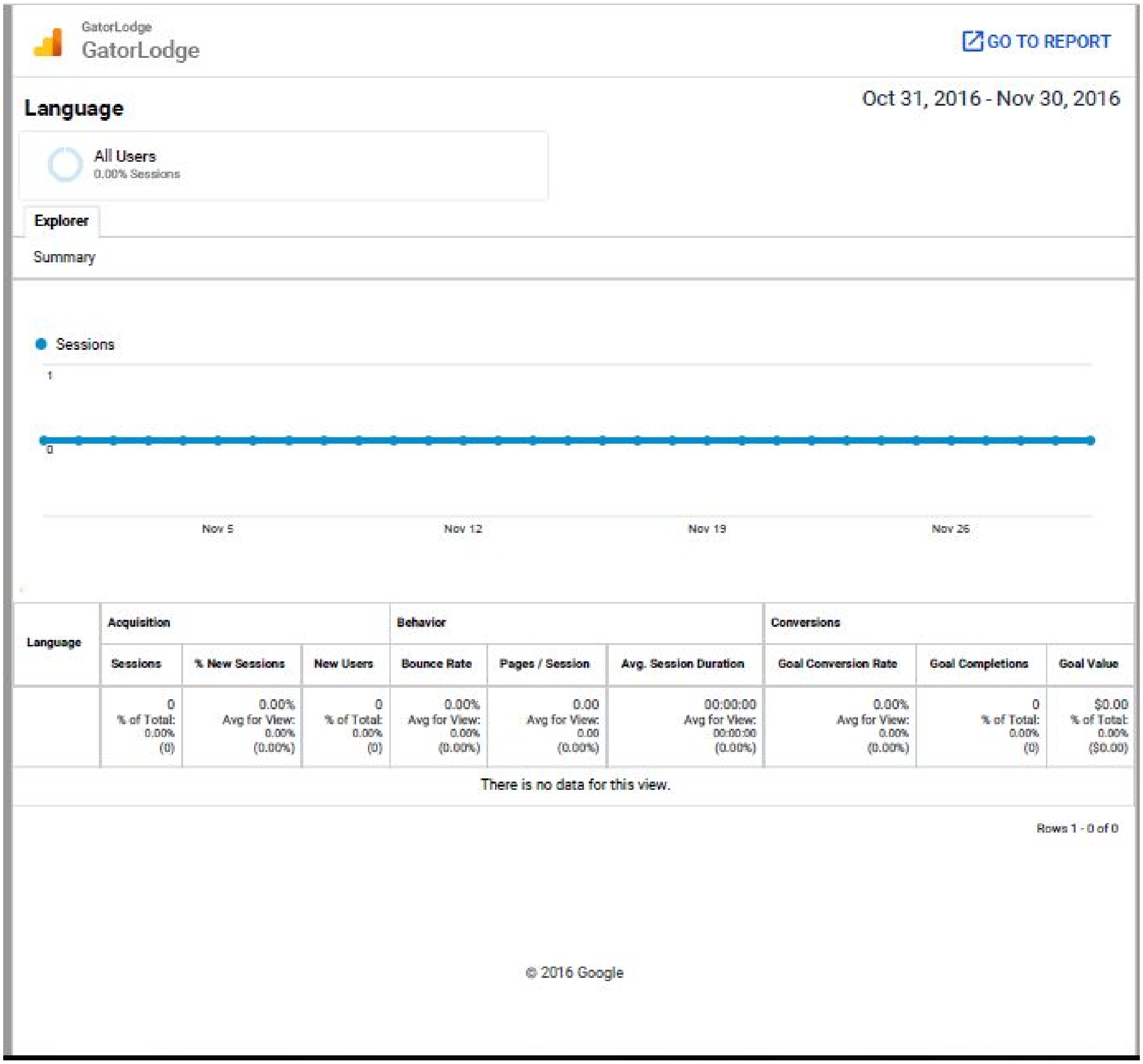
18. Pay functionality (how to pay for goods and services) shall be simulated with proper UI, no backend. **OUT OF SCOPE - removed by management**

19. Site security: basic best practices shall be applied (as covered in the class) **ON TRACK**

20. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development, and only the tools and practices approved by instructors **DONE**

21. 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) Provide screen shot of Google analytics showing usage of your home page



Language

Oct 31, 2016 - Nov 30, 2016

All Users
0.00% Sessions

Language	Acquisition			Behavior			Conversions		
	Sessions	% New Sessions	New Users	Bounce Rate	Pages / Session	Avg. Session Duration	Goal Conversion Rate	Goal Completions	Goal Value
	0 % of Total: 0.00% (0)	0.00% Avg for View: 0.00% (0.00%)	0 % of Total: 0.00% (0)	0.00% Avg for View: 0.00% (0.00%)	0.00 Avg for View: 0.00 (0.00%)	00:00:00 Avg for View: 00:00:00 (0.00%)	0.00% Avg for View: 0.00% (0.00%)	0 % of Total: 0.00% (0)	$0.00 % of Total: 0.00% ($0.00)

There is no data for this view.

Rows 1 - 0 of 0

© 2016 Google

Figure 5 Google Analytics