CMSC 447: Software Design and Development

**UMBC Parking Pal**

**Testing Report**

Customer:

Katherine Gibson

Team: UMBC Construction Workers

Abbie Minor

Constantin Koehler

Braxton Dubin

Naomi Schumacher

Sarah Kirby

UMBC Parking Pal

Testing Report

**Table of Contents**

[**1. Introduction**](#_1xgl85n68i7v) **2**

[1.1 Purpose of This Document](#_3cymfnq9l0x2) [2](#_3cymfnq9l0x2)

[1.2 References](#_ow5qhhkacny) [2](#_ow5qhhkacny)

[**2. Testing Process**](#_ic0tx91ooltb)[**2**](#_ic0tx91ooltb)

[2.1 Description](#_fl538cqa79tv) [2](#_fl538cqa79tv)

[2.2 Testing Sessions](#_o3kdu8hcbdgt) [3](#_o3kdu8hcbdgt)

[2.3 Impressions of the Process](#_ine52nv43ypy) [3](#_ine52nv43ypy)

[**3. Test Results**](#_510nt3k3tq35)[**3**](#_510nt3k3tq35)

[**Appendix A – Agreement Between Customer and Contractor**](#_l1ihjqbw7bu)[**10**](#_l1ihjqbw7bu)

[**Appendix B – Team Review Sign-off**](#_6q8tke53gd53)[**11**](#_6q8tke53gd53)

[**Appendix C – Document Contributions**](#_hf77dgs9axja)[**12**](#_hf77dgs9axja)

# 1. **Introduction**

## 1.1 Purpose of This Document

The purpose of this document is to outline the testing methods that we followed in order to inspect our webapp for any possible flaws. This document is intended for use by members of the UMBC Construction Workers as well as our client to help bring attention to faults or otherwise overlooked portions of our application.

## 1.2 References

The UMBC ParkingPal System Requirements Specification

The UMBC ParkingPal Code Inspection Report

# 2. **Testing Process**

Outlined below is the testing process we followed to evaluate our webapp as well as any other related information or thoughts regarding the process.

## 2.1 Description

Before starting a formal testing session, we made sure to continually assess our project as we worked on it to ensure that there is as few issues as possible when we reached the formal testing stage. These informal sessions were done individually on a page-by-page basis, depending on what the member was working on at the time. If any bugs were found in these sessions, the member would fix them as they came across them. Doing this prevented any unnecessary problems further down the road as the code became more complex.

Our formal testing process was done in conjunction with the code inspection process. While we looked through the actual code of our webapp, we navigated through the pages to make sure everything we had coded performed as intended. We did not follow a strict standard; rather, we simply tested for any possible defects that could have shown up in our web application one-by-one. Unlike the informal testing session however, we wrote down any defects that we came across and opted to fix them after testing the remaining Use Cases. This made it much easier to streamline the process so that we weren’t trying to fix bugs in the middle of testing.

## 2.2 Testing Sessions

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Date | Location | Time Started | Time Ended | Who Performed the Test | Which Use Cases Covered |
| 11/29/16 | UMBC | 9pm | 10pm | Abbie | Register  Login |
| 11/29/16 | Home | 7pm | 8pm | Constantin | Registration  and Navigation when not logged in |
| 11/29/16 | UMBC | 5pm | 6pm | Naomi | List a Spot  Search for a Spot |

## 2.3 Impressions of the Process

The testing process is centered around making sure the use cases were satisfied. The testing process was both efficient and effective. Based on the use cases, test conditions were created and used to make sure the system requirements were fulfilled. The quality of the program definitely increased after repairs were made.

The best modular units of the program are the Welcome screen and the Homepage. The Welcome screen is what all users see before logging in. It is important for this page to be free of defects because all users must use this part of the web app. The home page must also be defect-free. The user must return to the home page in use cases. The worst modular unit of the program is the implementation of the student database. Many parts of the web app must pull and push information from the student database. Adding this functionality will take the most amount of time so currently there are still several defects in the code.

# 3. **Test Results**

**Testing Suite**

Each of the following tests is for a specific use case corresponding to those defined in the SRS. We describe the purpose of each test, and the prior inputs necessary to perform the test. Each test has a set of valid and invalid inputs, their expected outputs, and boundary cases if applicable.

|  |  |
| --- | --- |
| **Use Case:** | **Login** |
| Purpose of test: | To test the ability to log in with UMBC credentials. |
| Prior Inputs: | None |
| Valid Input: | Valid UMBC email, eg. “skirb@umbc.edu” |
| Expected Output for Valid Input: | Redirect to personal home page if the user is registered  Redirect to registration page if the user is not registered |
| Invalid Input: | Non-UMBC email, eg. “[skirb@gmail.com](mailto:skirb1@gmail.com)”  Invalid UMBC email, eg. “@umbc.edu”  Valid UMBC email with an incorrect password. |
| Expected Output for Invalid Input: | Login fails, redirect to default splash page |
| Boundary Cases: | A UMBC email for an alumni will still allow a person to log in. |

|  |  |
| --- | --- |
| **Use Case:** | **Register** |
| Purpose of test: | To test the ability to register a new user. |
| Prior Input: | Must be logged in |
| Valid Input: | Phone number in the form “555-555-5555”  Car make “Toyota”  Car model “Rav4”  Color “Grey”  License Plate “XYZ123” |
| Expected Output for Valid Input: | User added to database, redirected to personal home page |
| Invalid Input: | Invalid phone number, i.e. “5555555555”, “555”, etc.  No input for make, model, color, or license plate. |
| Expected Output for Invalid Input: | Request to re-enter information. |
| Boundary Cases: | The form will accept any input for car make, model, color, and license plate. The user could enter car make: “Toy”, model: “!!”, color: “white\*” and still register successfully. |

|  |  |
| --- | --- |
| **Use Case:** | **List a Spot** |
| Purpose of test: | To post a parking space to sell |
| Prior Input: | Logged in and registered |
| Valid Input: | Price between $1 and $5,  Hour between 1 and 12,  Minute between 0 and 59,  Selected “AM” or “PM”,  Lot Number between 1 and 29. |
| Expected Output for Valid Input: | Redirects to personal home page with active listing displayed. |
| Invalid Input: | Price below 1 or above 5,  Hour below 1 or above 12,  Minute below 0 or above 59,  No selection for AM/PM,  No selection for Lot Number dropdown |
| Expected Output for Invalid Input: | Error message requests to re-enter parking information. |
| Boundary Cases: | Price $1 or $5, Hour 1 or 12, Minute 0 or 59 |

|  |  |
| --- | --- |
| **Use Case:** | **Search for a Spot to Buy** |
| Purpose of test: | To give the user the ability to find a parking space on campus |
| Prior Input: | Must be logged in and registered |
| Valid Input: | User selects any of the filtering options, or ordering options, or none. |
| Expected Output for Valid Input: | All spots that meet filtering requirements are listed in the order specified. |
| Invalid Input: | N/A |
| Expected Output for Invalid Input: | N/A |
| Boundary Case: | If there are no listings to display, the page will be blank. |

|  |  |
| --- | --- |
| **Use Case:** | **Purchase a Spot** |
| Purpose of test: | To ensure that a spot can be purchased |
| Prior Input: | Must be logged in and registered. There must be listings displayed on the search page. |
| Valid Input: | User selects “buy” on a listing. |
| Expected Output for Valid Input: | User clicks on the “Buy” button and confirmation modal appears. If they select “Buy” then the spot is purchased and the user is redirected to their personal home page. This purchase is now listed on their page. If they press “Cancel” in the modal then they go back to the listings page. |
| Invalid Input: | N/A |
| Expected Output for Invalid Input: | N/A |
| Boundary Cases: | N/A |

|  |  |
| --- | --- |
| **Use Case:** | **Leave a Rating** |
| Purpose of test: | A rating is left to show the reliability of seller |
| Prior Input: | Must have completed a purchase or a sale |
| Valid Input: | User clicks on the number of stars they want to give the other user. |
| Expected Output for Valid Input: | Rating is updated for the user, page redirects to home page |
| Invalid Input: | No input. |
| Expected Output for Invalid Input: | Default 5 stars is entered. Page redirects to home. |
| Boundary Case: | N/A |

|  |  |
| --- | --- |
| **Use Case:** | **Cancel a Purchase** |
| Purpose of test: | Gives the buyer the opportunity to cancel a purchase. |
| Prior Input: | User is logged in and has a purchase displayed on their home page. |
| Valid Input: | User selects the cancel button on their purchase. |
| Expected Output for Valid Input: | The purchase is canceled and the spot is relisted. |
| Invalid Input: | User selects cancel but they are not within the valid timeslot (it is less than 10 minutes before the time listed) |
| Expected Output for Invalid Input: | An error message is displayed. |
| Boundary Cases: | User cancels exactly 10 minutes before the listed time. |

|  |  |
| --- | --- |
| **Use Case:** | **Cancel a Listing** |
| Purpose of test: | Gives the seller an opportunity to cancel a listing |
| Prior Input: | User is logged in and has their listing displayed on the home page. |
| Valid Input: | User selects the cancel button on the listing. |
| Expected Output for Valid Input: | Listing is removed and a message is displayed on the home page |
| Invalid Input: | User selects cancel but they are not within the valid timeslot (it is less than 10 minutes before the time listed) |
| Expected Output for Invalid Input: | An error message is displayed. |
| Boundary Cases: | User cancels exactly 10 minutes before the listed time. |

|  |  |
| --- | --- |
| **Use Case:** | **Edit Account Info** |
| Purpose of test: | To make sure the user is able to change their account information |
| Prior Input: | Must be logged in and registered |
| Valid Input: | Changes to any field, eg. the following:  Phone number in the form “555-555-5555”  Car make “Toyota”  Car model “Rav4”  Color “Grey”  License Plate “XYZ123” |
| Expected Output for Valid Input: | Updates user info in database, redirects to home page. |
| Invalid Input: | Phone Number in the wrong format, eg. “1112223333” or “555”. Blank fields for make, model, color, or license plate. |
| Expected Output for Invalid Input: | Request to re-enter information. |
| Boundary Cases: | The form will accept any input for car make, model, color, and license plate. The user could enter car make: “Toy”, model: “!!”, color: “white\*” and the user’s info will update successfully. |

**Test Results**

The following is a summary of our tests and their results. First we listed our successful tests, which performed as expected. Next, we listed the unsuccessful tests, which did not have the outputs expected. We described the tests, as well as any defects detected and suggested repairs for the defects.

Successful Tests

**Use Case**: Cancel a Listing

**Tester**: Constantin

**In Working Condition?**: Yes

**Description**: From the homepage, the user is able to cancel a listing very easily with the cross icon in the top right. This works as intended.

**Use Case:** Edit Account Info

**Tester**: Constantin

**In Working Condition?**: Yes

**Description**: The user can edit any of their account information except for their UMBC email. When the user submits the form, it updates their stored information in the database.

Unsuccessful Tests

**Use Case**: Login

**Tester:** Abbie

**In Working Condition?**: Partial

**Description**: The actual login authentication works correctly. It differentiates UMBC email addresses from miscellaneous ones and will not let the user login to the webapp if they try to login with a non-UMBC email.

**Defects Detected:** The user is not automatically redirected login page. They are free to peruse through listings and are not limited in their interactions with the application when they should be.

**Suggested Action to Fix Bug:** The site needs to redirect to the Register page after a successful login. Currently it does not, but the fix should be a simple page redirect assignment. It has not been fixed yet due to working on other site issues.

**Use Case:** Register

**Tester:** Abbie

**In Working Condition?**: Partial

**Description**: This page should be redirected to on the first time the user logs in to the site. This is where they will register and enter all of their information. Phone number format, presence of car make, model, and color are all checked for before the user can submit their information. All information submitted on the Register page is saved in the database and appears on the user’s Account Settings Page.

**Defects Detected:** If the user has not yet created an account on our web application, it does not automatically redirect the user to the registration page. At this time, you must manually click on the “Register” tab on the navbar. Eventually it will be taken off of the navbar and the user will be automatically redirected to this page if they are not registered on the site. Because it is tied to the navigation bar at the top at this time, even users that already have an account can navigate to this page. Currently the license plate contains no check as they often vary and users most likely do not know it off the top of their head.

**Suggested Action to Fix Bug:** The page needs to be redirected to automatically after the first login of a new user. This should be a simple redirect fix. It has not yet been fixed or removed from the navbar for ease of testing. It will be removed once the redirect is handled.

**Use Case**: List a Spot

**Tester:** Naomi

**In Working Condition?**: Partial

**Description**: The user is able to list a spot successfully if they enter valid information. They are asked to fix the information if there are any errors with the listing information. This works as intended. However, not all of the information that should be displayed is actually on the listing cards, such as the time the user is leaving.

**Defects Detected:** Not all necessary information is displayed on the listing cards.

**Suggested Action to Fix Bug:** This defect is not very difficult to fix. We just need to tie the valid information to the listing card and display it to the user, preferably on the card itself and not the pop-up specifically for the time of departure.

**Use Case**: Search For a Spot to Buy

**Tester:** Naomi

**In Working Condition?**: Partial

**Description**: The search page has basic functionality. However, since the filter buttons do not work, the page lists all spots in the database and cannot currently be sorted in any other way.

**Defects Detected**: Filter buttons are not functional.

**Suggested Action to Fix Bug**: To fix this bug, we have to implement some Javascript to take all of the listings in the database and, for ease of sorting, put it into an array or container. Afterwards, the list should be much easier to sort and display to the user as necessary. There should also be a way to check if a listing should show up at all, such as a simple boolean that checks to see if that particular listing falls into that category picked by the user or not. For example, if a user is only looking for C lot parking spots, then parking spots that do not apply to this filter should not show up on the search page.

**Use Case:** Purchase a Spot

**Tester:** Constantin

**In Working Condition?**:Partial

**Description:** When looking for parking spaces, the user can purchase a spot by going through a fake PayPal transaction. The user is able to see all of the details of the transaction afterwards; however, the listing does not get deleted from the database and still appears in searches.

**Defects Detected**: The purchased listing is not deleted from the database and still appears in searches.

**Suggested Action to Fix Bug**: We need to add a boolean field to the ParkingSpots database, which will tell whether a listing is active or not. True means active, and the spot should be listed in searches. False would indicate that a spot has been purchased, and should not appear in searches. Edits will also have to be made to the search page to only display listings with a “True” value for this field. The listing will actually be deleted from the database only after the spot exchange is confirmed.

**Use Case**: Leave a Rating

**Tester**: Constantin

**In Working Condition?**: No

**Description**: The user will be prompted to leave a rating when they confirm that the spot has been exchanged. This is not functional yet. There is no confirmation page.

**Defects Detected**: The rating page does not appear after an exchange is confirmed.

**Suggested Action to Fix Bug**: We need to add a button to confirm the exchange of a purchased spot. This button will redirect to the ratings page.

**Use Case**: Cancel a Purchase

**Tester**: Braxton

**In Working Condition?**: No

**Description**: The user can cancel the purchase at any time. Cancelling a purchase will make the user return to the home page.

**Defects Detected**: There is no way to cancel a purchase.

**Suggested Action to Fix Bug**: We need to add a button so that a user can cancel a purchase. This button will have to first check to see if the user is canceling within 10 minutes of the listed time. If it is within 10 minutes, the user cannot cancel. If they are outside that window, then they will be prompted with a confirmation screen. Upon confirming the cancellation, the database will be updated, and the boolean flag for the listing will be set as “true” aka. “searchable” again.

# 

# **Appendix A – Agreement Between Customer and Contractor**

By signing this document all members of the team verify that they were involved in the testing process, and that they agree with the process and the results.

The customer agrees to attend the in-class demo of the final product on either 6 December 2016 or 8 December 2016.

Abbie Minor \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_

Constantin Koehler \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_

Braxton Dubin \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_

Naomi Schumacher \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_

Sarah Kirby \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_

Katherine Gibson \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_

(Customer)

Customer Comments:

# Appendix B – Team Review Sign-off

All members of the team have reviewed this document and agree on its content and format. They agree to discuss possible areas for change with other team members. The comment areas below are to be used to state any minor points regarding the document that members may not agree with.

Abbie Minor \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_

Comments:

Constantin Koehler \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_

Comments:

Braxton Dubin \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_

Comments:

Naomi Schumacher \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_

Comments:

Sarah Kirby \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_

Comments:

# Appendix C – Document Contributions

This section identifies how each member contributed to the creation of this document. The percentages listed are an estimate of the percentage of work each person contributed.

Abbie Minor: 20%

Contribution: Testing Suite, Proofreading

Constantin Koehler: 15%

Contribution: Test Results

Braxton Dubin: 15%

Contribution: Impressions of Process, Testing suite, Proofreading, Test Results

Naomi Schumacher: 30%

Contribution: Introduction, Test Suite, Test Results, Proofreading

Sarah Kirby: 20%

Contribution: Test Suite, Test Results