

UMBC ParkingPal Administrator Manual

Customer:

Katherine Gibson

Team: UMBC Construction Workers

Abbie Minor

Constantin Koehler

Braxton Dubin

Naomi Schumacher

Sarah Kirby

UMBC Parking Pal
Administrator Manual

Table of Contents

1. Introduction	3
1.1 Purpose of This Document	3
1.2 References	3
2. System Overview	3
2.1 Background	3
2.2 Hardware and Software Requirements	3
3. Administrative Procedures	4
3.1 Installation	4
3.2 Routine Tasks	4
3.3 Periodic Administration	4
3.4 User Support	5
4. Troubleshooting	5
4.1 Dealing with Error Messages and Failures	5
4.2 Known Bugs and Limitations	5
Appendix A – Team Review Sign-off	7
Appendix B – Document Contributions	8

1. Introduction

1.1 Purpose of This Document

The purpose of this document is to detail the administrative procedures and the general functionality of this webapp. This document is intended for a potential administrator of the UMBC Parking Pal system. The document includes hardware and software requirements to run the web app as well as instructions for administrative tasks like installment and maintenance of the system. Finally, it includes troubleshooting tips for known bugs in the system.

1.2 References

The UMBC ParkingPal System Requirements Specification

The UMBC ParkingPal Code Inspection Report

The UMBC ParkingPal System Design Document

2. System Overview

The following gives a brief background and then details hardware and software requirements.

2.1 Background

The system administrator should read the references from Section 1.2 before proceeding. This will give the administrator a general understanding of the system before attempting any Sys Admin tasks.

The system administrator should also need to make sure the system is being used properly by all users. They would be responsible for manually deleting any listings that happen to be in the database past their listed time. Weekly server maintenance would be required to patch system vulnerabilities. In the event of a server crash, the system administrator would have to ensure that no users in the middle of a transaction lost their money.

2.2 Hardware and Software Requirements

The Webapp must be hosted on a local Linux server and must have enough processors for the administrator to run asynchronous tasks seamlessly. It is also required that the server must also have enough storage to hold the database. The server must have Meteor installed to monitor the database and perform maintenance on the site. MongoDB is included in the installation of Meteor.

3. Administrative Procedures

The following are the required and recommended administrative tasks, and instructions to perform the tasks.

3.1 Installation

The webapp must be hosted on a local Linux server. The only installation necessary is the installation of Meteor. To install Meteor, verify that the server is connected to the internet. Then, open a terminal and type the command “curl <https://install.meteor.com/> | sh”. The installation will begin and the administrator can view its progress in the terminal.

3.2 Routine Tasks

There are two routine tasks that we recommend to run UMBC Parking Pal. First, we recommend routinely monitoring the server performance and memory usage to ensure that the website is up and running without any performance issues. If performance or memory usage is maxed out, the administrator may need to upgrade or add additional servers.

Second, we recommend monitoring the database of listings. There is a known bug in the system where listings do not always expire correctly, so the admin should check the listings daily to make sure that this functions properly. To perform this task, the admin must use the Mongo shell. To open the Mongo shell, the Sys Admin should navigate to the main folder where the web app files are stored. Open a terminal here, and type “meteor mongo”. In the mongo shell, type “use meteor” then “db.parkingSpaces.find({});”. All of the parking spaces stored in the database will then be displayed and the admin can then check the listing times to make sure that none have times that should have expired.

If there are any incorrect times, the admin can remove them by using the command “db.parkingSpaces.remove({userName: <users name>})” in the Mongo shell. The admin user’s name entered must correspond to the incorrect listing. We recommend performing this task at midnight every night to ensure that the database will be clear for the following day. It is possible to automate this process, but there is not currently a system in place to do so.

3.3 Periodic Administration

We recommend periodically performing system backups of the server. We also recommend checking the Meteor framework for updates, as well as all of the various packages we use. To accomplish this, stop the instance of Parking Pal and inside that terminal prompt, run “Meteor update” to update the Meteor library. Meteor is a new, up-and-coming framework that should be

updated frequently. To update the packages that our application relies upon, run “meteor npm update” in terminal.

3.4 User Support

There is not currently a user support system in place, but there is an FAQ page on the website. If the administrator wishes to do so, they may add contact information to the FAQ page of the website for user support. They can then monitor this contact and update the FAQ with common questions and troubleshooting information.

4. Troubleshooting

The following section details how to deal with error messages and failures before explaining known bugs and system limitations.

4.1 Dealing with Error Messages and Failures

When first installing Meteor, the Sys Admin may be presented with several error messages or failures. Before trying to run our application, ensure that Meteor is installed from the Meteor website and that all packages are installed to run it. To install these packages, navigate to our folder and in a terminal run a “meteor npm install” command. This will download packages using the node package manager and install them.

Error Messages on the client-side/front-end can be seen by opening the inspect tool on the web browser. There the administrator can see what errors are being returned to the user. They may also embed Console.log statement inside of the client Javascript to aid debugging by printing to the console. If they are debugging server-side code or are encountering failures on the back-end, error messages, warnings, and debugging print statements can be found in the terminal window designated to run the web app.

4.2 Known Bugs and Limitations

Currently, there are two known bugs in the system. First, when a listing is purchased, the buyer and seller information does not appear on the listing card as expected. This occurs in the client folder in the ParkingSpaces.js and MyParkingSpace.html files. Attempts were made to fix the bug and from inspection it appears the fix should work, however it does not. We believe we are unable to fix it due to unfamiliarity with Meteor and time constraints. It would be optimal if this type of issue could be researched and fixed immediately, since the site will become unusable as buyers and sellers have no means of communicating with one another.


Additionally, as previously discussed in Routine Tasks, there is a bug in the expiration of listings. In the in the init.js file in the server folder and parkingSpaces.js file in the collections folder, is the location of the code dealing with expiration of listings. This functionality was working at one point in time, but became non-functional. We were unable to determine the cause of the code no longer working. Additionally, the expired spots would still appear on the listings page, making it cluttered and unruly. It would positively impact user experience for the bug to be addressed and would ease the duties of the system administrator by alleviating the need to clear out the database daily. Given more time and resources, this bug could and should be fixed.

Appendix A – Team Review Sign-off

All members of the team have reviewed this document and agree on its content and format. 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: 12/8/2016


Comments:

Constantin Koehler _____  _____ Date: 12/8/2016

Comments:

Braxton Dubin _____  _____ Date: 12/8/2016

Comments:

Naomi Schumacher _____  _____ Date: 12/8/16

Comments:

Sarah Kirby _____  _____ Date: _12/8/16_

Comments:

Appendix B – 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: 25%

Contribution: Known Bugs and Limitations

Constantin Koehler: 25%

Contribution: Dealing with Error Messages and Failure, Periodic Administration

Braxton Dubin: 14%

Contribution: Purpose, Hardware and Software Requirements, Background

Naomi Schumacher: 1%

Contribution: Proofreading

Sarah Kirby: 35%

Contribution: Purpose, Hardware and Software Requirements, Administrative Procedures