CMSC 447: Software Engineering I

UMBC ParkingPal
System Requirements Specification

Customer:
Katherine Gibson

Team: UMBC Construction Workers
Abbie Minor
Constantin Koehler
Braxton Dubin
Naomi Schumacher
Sarah Kirby

9/22/16

UMBC ParkingPal

System Requirements Specification

Table of Contents

1. Introduction

This section provides an overview of what is included in the SRS document as well as an overview of the product, UMBC ParkingPal. A list of references used while working on the application and documentation is included.

1.1 Purpose of This Document

The purpose of this document is to give a detailed description of the requirements for the UMBC ParkingPal web application. It includes the purpose of the product, scope of the product, function system requirements including multiple use cases and tests for the use cases, non-functional system requirements, deliverables, open issues, and a contract between the customer and development team. The intended audience is the development team and the customer. The document is to be presented to the customer for approval and used by the development team for reference in application completion.

1.2 References

[1] Wilson, K (2016). Chapter 4 - Requirements Engineering [Powerpoint Slides]. Retrieved from Blackboard.

[2] Wilson, K (2016). SRS Template [Word Document]. Retrieved from Blackboard.

1.3 Purpose of the Product

The idea for UMBC ParkingPal was sparked by the daily struggle that students and faculty who commute to UMBC face. The immense flow of cars entering and leaving UMBC can cause people to search for a parking spot for upwards of 30 minutes if they do not make it to campus early enough. UMBC ParkingPal seeks to solve that problem by allowing users to find places where people are about to leave. A person leaving can sell their spot to someone who is looking for one. Buyers can search by lot, and after the purchase is made the buyer receives more detail regarding the exact location and what car they are looking for to exchange the spot.

1.4 Product Scope

The UMBC ParkingPal web application contains several main features (use cases) which include registering or logging in, selling parking spaces, searching for parking spaces, purchasing and exchanging the parking space, and canceling a purchase or listing. See Figure 1 below for a visual representation of how the buyer and seller can interact with the application.

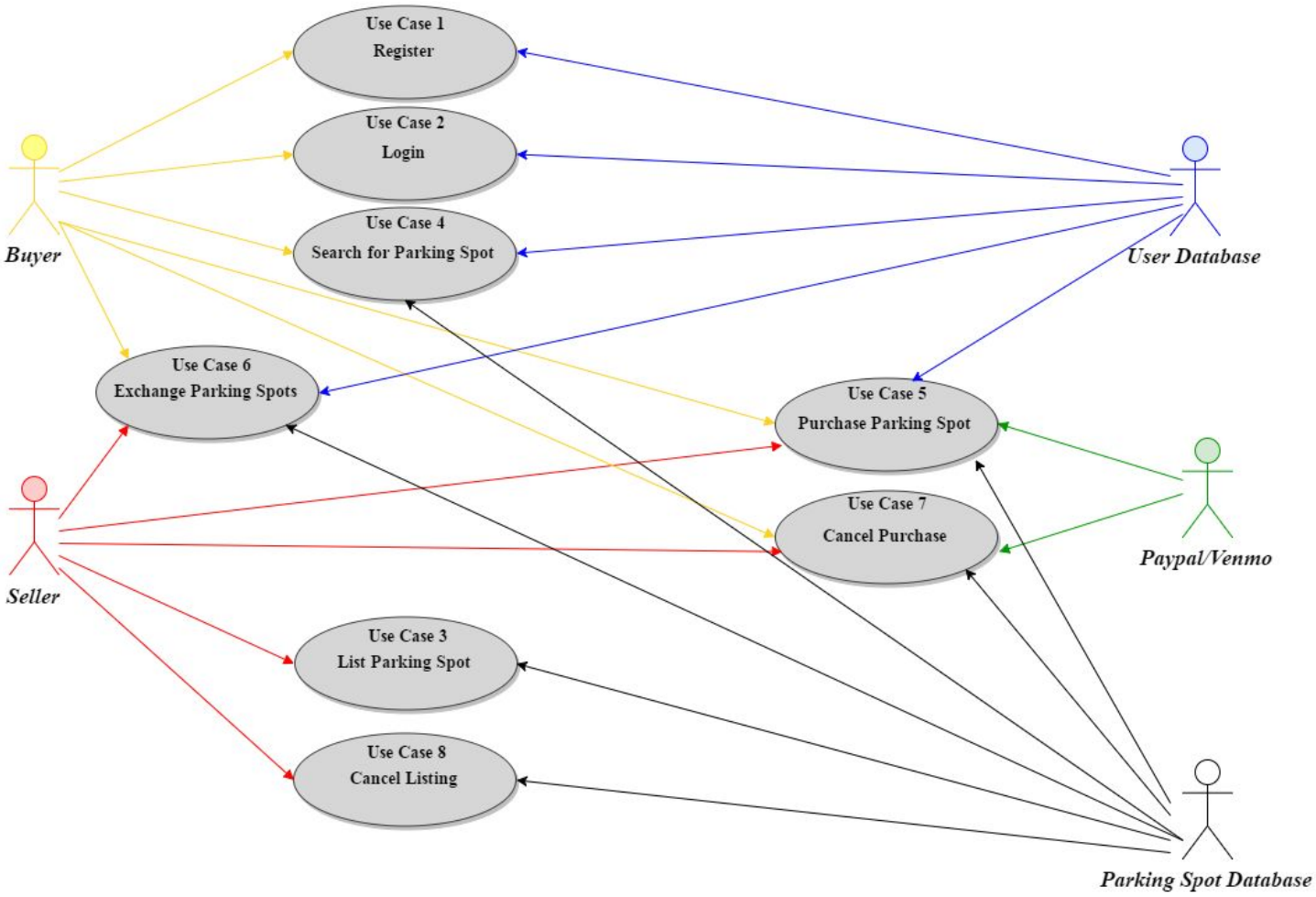

Figure 1: Use case diagram including use case numbers.

2. Functional Requirements

Detailed below are the functions the application must perform. These include the inputs, how the system should react to such inputs, and the outputs.

2.1 Use Cases

This section considers each use case in depth. Refer to Figure 1 for the overall view and interaction of all use cases.

2.1.1 Use Case 1

Name: Register
Summary: The user shall go to our web application and, using their UMBC email, will be able to register for the application.

Priority: 5
Preconditions: User must have a valid UMBC email.
Postconditions: The user will have an account for ParkingPal.
Primary Actor: Unregistered student
Secondary Actor: User Database
Trigger: User wishes to use our service
Main Scenario:

1. The user clicks on the register button on the website
2. The user will be asked to use their myUMBC e-mail
3. The user will create a password for their account
4. The user will be prompted to enter their parking pass type, vehicle information, and contact information.
5. The user will be asked to confirm their details.
6. The user will be asked to agree to the terms of the service and the UMBC campus parking policies.

Extensions:

2. a) User attempts to use invalid UMBC email or non-UMBC email: error message displayed.

3. a) User does not follow password specifications (if limitations on passwords implemented): error message displayed and user redirected to 3.

6. a) If the user does not agree to the terms they will not be able to register.

Open Issues:
Are there specifications for password length and characters included?

2.1.2 Use Case 2

Name: Login
Summary: The user shall go to our web application and, using their UMBC email and password created during registration, will be able to login to the application.
Priority: 5
Preconditions: User must have a valid UMBC email and be registered on the application.
Postconditions: The user will be logged in to ParkingPal.
Primary Actor: User with a ParkingPal account
Secondary Actor: myUMBC login system, User Database
Trigger: User wishes to use our service
Main Scenario:

1. The user clicks on the logon button on the website

2. The user will be asked to login using their myUMBC e-mail

Extensions:

2.a) Invalid credentials are entered: display login error message.

b) The user can ask for a password reset.

Open Issues:

Will the user have a limited number of attempts/login failures before the account locks?
Will there be a password reset feature and how will that be carried out?

2.1.3 Use Case 3

Name: User lists their parking spot
Summary: The user shall go to our web application and list their parking space for sale.
Priority: 5
Preconditions: The user must be logged on.
Postconditions: The parking space the user listed will now be listed as for sale.
Primary Actor: The user who is selling their parking space
Secondary Actor: User Database, Parking Spot Database
Trigger: User wishes to list their parking space as for sale
Main Scenario:

1. The seller will select "sell a spot" on the homepage.
2. The seller will select the lot they are in either from the map or drop-down menu.
3. The seller shall select the row they are parked in from the lot mapt.
4. The seller shall select a price from the drop down menu.
5. The seller shall enter the time they are leaving.
6. The seller shall confirm their listing details and list the spot.

Extensions:

3. a) If the seller already has an active listing, they will not be able to sell unless they delete their current listing.

Open Issues:

How does the user select their spot? Options include selecting from a map with numbered spots, dropping a pin on a map, or pre-populated options to narrow down where the parking spot is located.

2.1.4 Use Case 4

Name: User Searches for parking spot

Summary: The user shall go to our web application and view which parking spaces are for sale.
Priority: 5
Preconditions: The user must be logged on.
Postconditions: The user shall see which parking spaces are available for purchase.
Primary Actor: The user who is searching for parking spaces
Secondary Actor: The user selling their parking space, User Database, Parking Spot Database
Trigger: User wishes to park on campus
Main Scenario:

1. The user selects 'buy a spot.'
2. The user will select the lot they want from the drop down menu or from the map. They may also select "all lots." This will select all lots for their permit type.
3. The user will view a page with all the listings; they can select a specific lot via radio buttons.
4. The user can sort listings by ascending or descending time, price, and lot.

Extensions:

3. a) If the user already has an active purchase, they will not be able to buy another spot until the first is canceled.

Open Issues:
How the map is displayed for users to search for spots.
Will the buyer be able to narrow down the search via searching for a specific time frame? (ex: narrow down for a parking spot opening in between 1:30-2:00)

2.1.5 Use Case 5

Name: User purchases parking spot
Summary: The user purchases a spot listed for sale and the seller is paid.
Priority: 5
Preconditions: The user must be logged on and have searched listings already.
Postconditions: The user shall pay for the space and the seller shall be paid.
Primary Actor: The buyer.
Secondary Actor: The seller, PayPal system, User Database, Parking Spot Database
Trigger: User wishes to pay for a space.
Main Scenario:

1. The buyer selects a parking spot from the listings page.
2. The buyer agrees to pay the price for the parking spot and purchases it.
3. The buyer will receive a confirmation with the exact location and details of the car they are looking for, as well as the time window for the exchange and the cancellation policy.

4. The seller will receive a notification that the spot has been sold and will be informed of the time window for the exchange and the cancellation policy.

Extensions:

1. a) Invalid payment occurs. The buyer is notified that the purchase failed and the spot remains listed.

Open Issues:
If an invalid payment occurs, does the buyer return to the purchase page or the home page?
Figuring out how make a mock transaction. (PayPal connection will not be implemented)
The time window is not determined yet.

2.1.6 Use Case 6

Name: Buyer and Seller exchange parking spots
Summary: The buyer and seller will confirm the transaction and exchange parking spots.
Priority: 5
Preconditions: The buyer must have purchased the parking spot.
Postconditions: The buyer will take the seller's parking space.
Primary Actor: The buyer
Secondary Actor: The seller, User Database, Parking Spot Database
Trigger: The buyer has purchased a parking spot
Main Scenario:

1. The seller prepares to leave their parking space.
2. The buyer finds the seller's location.
3. The buyer and seller both acknowledge in the app that the other person is there.
4. The transaction is finalized (verify spot was sold and exchanged).
5. The buyer takes the seller's parking spot.
6. Both buyer and seller will be prompted to leave a rating for each other.

Extensions:

3. a) If the buyer does not arrive at the parking spot within 10 minutes of the agreed time, the seller may leave and report that the exchange did not occur. The buyer will not be refunded.

3. b) If the seller leaves before the time specified or before the 10 minute window is up, and the spot is taken when the buyer arrives, then the buyer can report that the exchange did not occur and will be refunded.

Open Issues:
Figuring out how to pay the seller. i.e Venmo or Paypal, other?
How long should the seller have to wait for the buyer to show up?

2.1.7 Use Case 7

Name: Buyer cancels their purchase
Summary: The buyer cancels their purchase of a parking spot.
Priority: 3
Preconditions: The buyer must have purchased a parking spot.
Postconditions: The buyer will be refunded if the cancellation is successful.
Primary Actor: The buyer.
Secondary Actor: The seller, PayPal, Parking Spot Database
Trigger: The buyer decides to cancel their purchase.
Main Scenario:

1. The buyer logs on to the UMBC ParkingPal system.
2. The buyer selects the purchase from their active purchases.
3. The buyer selects the cancel button.
4. PayPal will refund their purchase.
5. The buyer and seller will be notified of the outcome of the cancellation.

Extensions:

3. a) If the buyer tries to cancel within 10 minutes of the time they are supposed to take the spot, they will receive a notification that they are not eligible for a refund. They will be allowed to continue with the cancellation or keep the spot that they've already bought.

Open Issues:
The time cutoff for when a user can no longer cancel is still undecided.

2.1.8 Use Case 8

Name: Seller cancels their listing
Summary: The seller removes their parking spot listing.
Priority: 3
Preconditions: The seller must have a listed spot.
Postconditions: The listing will be removed from the parking spot database if the cancellation is successful.
Primary Actor: The seller
Secondary Actor: Parking Spot Database
Trigger: The seller decides to cancel their listing.
Main Scenario:

1. The seller logs on to the UMBC ParkingPal system.
2. The seller selects the listing from their active listings.

3. The seller selects the cancel button.
4. The listing will be removed from the database.
5. The seller will be notified of the outcome of the cancellation.

Extensions:

3. a) If the seller tries to cancel a listing that has already been sold, the cancellation will fail.

Open Issues:

None

2.2 Use Case Tests

This section lists the tests that will need to be performed to ensure the use cases listed above are working according to the requirements and expected outcomes.

2.2.1 Use Case Test 1

The first thing we will test is that a user is able to register to the website with their UMBC email. To test this, we will register with our UMBC e-mails and make sure the user database is populated. We will also test it with an invalid UMBC email address and non-UMBC email address.

2.2.2 Use Case Test 2

We will test the user's ability to login after account creation. We will do this by creating accounts and logging in. User ability to reset their password will also be tested.

2.2.3 Use Case Test 3

We must test the listing feature by logging in and going through each step to sell a spot. We will check that the space is added to the listing database. We will test to see that a listing will be removed from the database when it expires.

2.2.4 Use Case Test 4

We will fully test the search feature for buyers. We will run several tests to see that each lot selection leads to a list of listings for that lot. We will test that the "all lots" selection leads to a list of listings for all lots of the user's permit.

2.2.5 Use Case Test 5

We will test our purchase page and confirmation pages, as well as the updates to the active listing for that buyer and the seller's listing. We will also test to see that bought listings do not show up in a search. We will not test the PayPal connection, but we will test our mock implementation for both successful and unsuccessful purchases.

2.2.6 Use Case Test 6

We will run through the entire process of buying and selling a parking spot between two team members to ensure everything is working. We will test the verification for successful exchanges, exchanges where the buyer does not show and exchanges where the seller is not there. We will also test the ratings for users to see that they are updated after an exchange is verified.

2.2.7 Use Case Test 7

We will purchase a spot as a buyer, and run through the process of cancelling with a successful and failed cancellation. We will test that canceled purchases still within the time frame reappear in the search.

2.2.8 Use Case Test 8

We will list a spot as a seller, and run through the process of cancelling with a successful and a failed cancellation. We will test that canceled listings are removed from the listings database.

3. Non-Functional Requirements

This section details the constraints placed on the system, the priority at which the customer and the development team rank them, and how to test that the constraints are met.

NFR #	Priority	Description	Test
1	5	The system shall be secure	Login will be tested with valid and invalid credentials. The system should respond correctly for all tests.
2	3	The system shall be well documented	We will count the number of discrepancies between documents and the final product.

3	5	The system shall be usable	We will have several users test our product and give us a rating on usability.
4	4	The system shall be responsive	We will have users test our pages and give us a rating on page responsiveness. We will also test the time it takes for pages to load.
5	5	The system shall be scalable	We will run tests with increasing database sizes to ensure that the pages maintain the same level of responsiveness.
6	4	The system shall be portable	We will test the webpage on various browsers and devices. It should be viewable on Chrome, Firefox, and mobile devices.
7	5	The system shall be implemented in JavaScript and HTML/CSS	N/A
8	4	The system shall be developed in a text editor of the developer's choice	N/A
9	5	Github shall be used for source control	N/A
10	5	The system shall not violate campus policies	Campus policies relating to parking will be read to ensure that our system does not violate them.

4. User Interface

See "User Interface Design Document for UMBC ParkingPal."

5. Deliverables

Listed below are all of the items to be delivered to the customer. Unless requested otherwise by the customer, all deliverables will be handed to the customer at the conclusion of this project.

Item	Format	Delivery Date	Comments

Systems Requirement Specification	Hard and soft copy	12/6/16	N/A
System Design Document	Hard and soft copy	12/6/16	N/A
User Interface Design Document	Hard and soft copy	12/6/16	N/A
User Manual	Hard and soft copy	12/6/16	N/A
Administrator Manual	Hard and soft copy	12/6/16	N/A
Copies of all Biweekly Status Reports	Hard copy	12/6/16	N/A
Source Code	Soft copy	12/6/16	N/A
Executable Program	Soft copy	12/6/16	N/A

6. Open Issues

- The exact method of payment within the app has not been chosen at the time of writing. Choices currently include PayPal and Venmo.
- Choice of what database for user information to use has not been made. Choices include SQL and Mongo.

Appendix A – Agreement Between Customer and Contractor

By signing this document all members of the team agree to complete the project with the requirements specified. They also agree to provide the deliverables on the dates specified.

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

If future changes need to be made to the requirements, the team members agree to consult with the customer and vice versa. If an agreement is made, it must be made in writing and the requirements documents must be updated with the changes.

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 follow the requirements specified for the development of UMBC ParkingPal. They agree to discuss possible areas for change with other team members and follow the guidelines for making changes to the requirements. 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: 30%
Contribution: Introduction, table of contents, functional requirements, formatting, and proofreading.

Constantin Koehler: 15%
Contribution: Functional requirements and proofreading

Braxton Dubin: 0%
Contribution: Unclear

Naomi Schumacher: 20%
Contribution: Deliverables, open issues, proofreading

Sarah Kirby: 35%
Contribution: Non-functional requirements, functional requirements, appendices, formatting, and proofreading.