Software Requirements Specification

for

Express Parking

Version 1.0 approved

Prepared by Aamna Bhatti, Jonathan Blumenfield, Jaz Mani, Ashish Ranjan, Satinder Sikand, and Haider Shoaib

Lassonde School of Engineering

November 9th, 2020

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
All Members	11/14/20	Final Draft	1.0

Introduction

1.1 Purpose

In the city of Toronto, there are 7500 single space parking meters, which need to be regularly visited to collect parking fares. This is a task that can be made faster, more efficient and less expensive. The purpose of this product is to minimize the number of parking meters that need to be visited to collect money, as well as to help parking enforcement workers monitor payments.

1.2 Document Conventions

The term "User" is an inclusive term, which refers to both customers and parking enforcement officers: the two types of users in the system.

"Customer": uses system to pay for parking spaces

"Parking Enforcement Officer": Monitors payments for parking spaces

1.3 Intended Audience and Reading Suggestions

This document is intended for the following audiences.

1. *Developers,* who need a strong understanding of the software requirements in order to design and implement the application
2. *Toronto Parking Authority,* who will help with the elicitation and verification of the software requirements
3. *Users*, including both parking enforcement officers and parking customers, may use this document to understand the various features in this system and learn how to use it

1.4 Product Scope

This product will be concerned with the details and management of parking payment. The application does not replace the need for officers to randomly visit parking spaces, as that requires additional infrastructure that our project is not able to incorporate. For example, a camera system. The system will only be used for single parking meters, and will not replace the ticket system that exists for the remaining 10,000 parking spaces in Toronto.

For customers, the application provides a fast, convenient, and easy way to pay for parking, allowing them to pay for a space without using a parking meter. For the customers sake, none of the functionality of the original parking system has been replaced, besides the ability to pay with cash.

For the parking enforcement officers, the application will provide a way of monitoring parking spaces and their payments. As mentioned above, this application will bypass the need for parking officers to visit parking meters to collect money.

1.5 References

Project Document

2. Overall Description

2.1 Product Perspective

This product is an optimization to the current parking meter system in Toronto where customers manually pay for parking using coins. The product will allow customers to pay for single parking spaces online, while the Toronto Parking Authority personnels do not have to collect money from each parking meter. The online payment system and the manual parking meters accomplish the same tasks, however it is more convenient for users to pay online for a parking space beforehand, and it is easier for the Toronto Parking Authority to collect payments online rather than physically collecting money from each meter.

2.2 Product Functions

- *Customers* can select a parking space(s) and pay for it
- *Parking enforcement officers* can add or remove parking spaces
- *Parking enforcement officers* can see if an occupied parking space has been paid for or not

2.3 User Classes and Characteristics

Customers (Primary)
The customers are the most common user base. They can make a payment for a parking space of their desire.

Toronto Parking Authority (Primary)
The Toronto Parking Authority manages the system through parking enforcements. They are able to add and remove parking spaces, as well as see if an occupied parking space has been paid for.

System Administrators (Primary)
The system administrators update the status of the user payment for the parking space. They are able to add and remove parking enforcement officers from the system. Finally, they change the *customer's* payment status from unpaid to paid.

Payment Services (Secondary)
The payment services do not use the system directly, but they are the ones who process payments with banks when users make their payments online.

Database (Secondary)
The database holds user information such as login information for verification when a user logs into the system, as well as when a user creates a new account.

2.4 Operating Environment

The express parking system will be programmed in the Java environment. It will be able to run on WIndows, MacOS, and Linux operating systems.

2.5 Design and Implementation Constraints

User Interface
This system is required to have a GUI interface. Classes should be implemented and tested to make sure that the system functions as it should. Currently this system will be programmed in Java, and in the future it can be implemented in an mobile app, which is not in the current scope of this project.

Payment Management
For the purposes of this project, payments should be verified by the name on the customer's credit card (i.e. the name on the card must match the customer's name).

Server Connection
Currently, the system will not be communicating with an external server or third party service.

2.6 User Documentation

Eclipse Java IDE installation guide: https://www.eclipse.org/downloads/packages/installer

2.7 Assumptions and Dependencies

- The system does not replace current parking meters
- The system administrator already exists in the system and has a master login
- Customers are allowed to book more than one parking space (with a limit)
- Parking enforcement officers must login as a customer in order to book a parking space for themselves

3. External Interface Requirements

3.1 User Interfaces

On the system there should be different login and user options with three different appearances for three different user bases, that being one screen for customers, one screen for the system administrator, and another screen for the Toronto Parking Authority who will act as an administrative role. The customer's features will belong to those who log in and want to authorize payment for a parking bay. Once they log into the system, they should see a standard layout with a set of options to initiate commands. The customer should be directed to a new screen where they can select a location to park by inputting the designated parking space number into a text box.

After this, there should be three options, one to go back to the previous screen if they enter the wrong input, an option for the user to add up to two more additional parking spaces to pay for, and a final option to check-out. If the customer decides to check out, they should be taken to a screen where they can enter their payment information (ex. credit card number, expiry date, etc.), and be able to pay for all parking spaces purchased.

On the interface for the Toronto Parking Authority, There must be a way for the user to easily access their features, such as creating a space for them to manage the amount of parking spaces available by adding or removing available spaces to park.Other features for the parking authority enforcers should be to view any user's parking space request and to be able to cancel a user's erroneous space request. With these features, the parking enforcement user needs to be able to see if an occupied space is paid for or not.

The system administrator interface should provide the user with a screen to log in using an ID and password, and cannot be accessed without these requirements. After logging into the system, the administrator should be able to choose to add or remove parking enforcement officers to the system, and they should be able to manually change the status of parking space payments to "paid".

3.2 Hardware Interfaces

The software can either be supported on a laptop or desktop with operating system support for Mac, Windows, or Linux.

3.3 Software Interfaces

The system should be integrated and implemented with the Java programming language on Eclipse, and ultimately should be able to run on such versions of Eclipse on Windows 10 or previous versions, as well as Mac OS X and Linux. No external downloads should be required other than for the usage of the web app itself. Messages that come into the system would be in the form of notifications such as firing events when certain errors are encountered by the user, such as trying to add an occupied space, and trying to pay for more than 2 parking spaces.

3.4 Communications Interfaces

Security Issues: Customers should not be able to access the Transit Authority UIs, or have any of their permissions, and in addition, all credit card and personal information must be implemented safely and securely. A password system should be put in place to ensure there is no way for an ordinary user to be able to access the features made exclusive for the Toronto Parking Authority, and to ensure that customers cannot easily access other user's credit card information or accounts.

4. System Features

4.1 Manage Parking Enforcement Officer

Priority: high

4.1.1 Description

This feature allows a *system administrator* to add or remove *parking enforcement officers* from the system. When an officer is added, an unique ID is assigned.

4.1.2 Stimulus/Response Sequences

- The system will ask information such as first name, last name, and email address is required to add a new *Parking Enforcement Officer*
- A *Parking Enforcement Officer* can be removed by the administrator if their email address exists in the system

4.1.3 Functional Requirements

4.1.3-REQ-1: The system must assign a unique ID to each new parking enforcement officer added
4.1.3-REQ-1: The system must verify the parking enforcement officer exists in the system before removing an officer
4.1.3-REQ-3: The system must verify a new parking enforcement officer's ID does not exist in the system already
4.1.3-REQ-4: The system must store a new parking enforcement officer's registration information for future authentication purposes

4.2 Customer Registration

Priority: high

4.2.1 Description

This feature allows *customers* to register a new account in order to use this application.

4.2.2 Stimulus/Response Sequences

- New *customers* will be displayed with a button such as "REGISTER", leading to a separate window, where once clicked, the *customer* can enter their first name, last name and email along with a password.

4.2.3 Functional Requirements

4.2.3-REQ-1: The system must accept entries with duplicate first and last names
4.2.3-REQ-2: The system must allow unique email addresses when registering
4.2.3-REQ-3: The system must store a new user's login information for future authentication purposes

4.3 User Login

Priority: high

4.3.1 Description

This feature allows the user to sign in, in order to access the application.

4.3.2 Stimulus/Response Sequences

- Users will be displayed with a button such as "SIGN-IN", leading to a separate window, where once clicked, the user can enter their login information

4.3.3 Functional Requirements

4.3.3-REQ-1: A user must be registered or added before logging in
4.3.3-REQ-2: The system must authenticate a user's login information before granting access to the application

4.4 Book a Parking Space

Priority: high

4.4.1 Description

This feature allows a *customer* to select which parking space, and how long they want to book it for.

4.4.2 Stimulus/Response Sequences

- The *customer* will be displayed with a button such as "BOOK SPACE", leading to a separate window, where once clicked, the *customer* is prompted to enter the parking space number, booking time, and their license plate number.

4.4.3 Functional Requirements

4.4.3-REQ-1: *Customer* must be registered and logged-in before booking a parking space

4.4.3-REQ-2: *Customer* must select which parking space they are booking
4.4.3-REQ-3: If the *customer* selects a parking space which is occupied, they are presented with an error message and must select a new space
4.4.3-REQ-4: *Customer* must enter how long they want to book said parking space for
4.4.3-REQ-5: The system allows a *customer* to book up to three parking spaces
4.4.3-REQ-6: Each booked parking space receives a unique booking ID
4.4.3-REQ-7: The system must display an error message if the parking space the customer booked is occupied in the system

4.5 Cancel a Parking space

Priority: high

4.5.1 Description

This feature allows a *customer* to cancel a parking space which they previously booked, bookings can only be canceled before they expire.

4.5.2 Stimulus/Response Sequences

- The *customer* clicks on a button labeled "CANCELLATIONS" and enters their booking number

4.5.3 Functional Requirements

4.5.3-REQ-1: *Customer* must be registered and logged-in before cancelling a parking space
4.5.3-REQ-2: *Customer* must enter the booking ID which is associated with their name
4.5.3-REQ-3: The cancellation can only go through if the time of cancellation is before the booking expiry time

4.6 Payment

Priority: high

4.6.1 Description

This feature allows the *customer* to pay for their currently booked parking space.

4.6.2 Stimulus/Response Sequences

- The *customer* will be displayed with a button such as "PAY", leading to a separate window, where once clicked, the *customer* is prompted to enter the parking space number they wish to pay for.
- *Customer* will be be shown total amount to be paid in a window
- Once confirmed, *customer* can proceed to enter payment information
- The *customer* confirms the transaction and the payment takes place
- The *System Administrator* changes the payment status from unpaid to paid once the *customer* payment is confirmed

4.6.3 Functional Requirements

4.6.3-REQ-1: *Customer* must be registered and logged-in before making a payment
4.6.3-REQ-2: *Customer* must have entered additional information such as which parking space they are booking before making payment
4.6.3-REQ-3: The system must accept different forms of payment (ex: Paypal, credit, debit, etc.)
4.6.3-REQ-4: The system must automatically timestamp each payment confirmation
4.6.3-REQ-5: The system automatically starts a countdown till expiry once payment is confirmed
4.6.3-REQ-6: The system allows a customer to pay for multiple parking space bookings (if any) cumulatively
4.6.3-REQ-7: The system must authenticate *customer* payment information before proceeding with confirmation of parking space

4.7 View Bookings

Priority: medium

4.7.1 Description

This feature shows a *customer's* current parking booking. Information such as expiry time and payment status can be viewed.

4.7.2 Stimulus/Response Sequences

- After logging in, *customers* will be displayed with a button labeled "VIEW BOOKINGS", leading to a separate window displaying all their bookings.
- *Customers* can select and open any of their current booking to display additional information like expiry time and payment status
- *Parking enforcement officers* can enter customer information and view their booking details

4.7.3 Functional Requirements

4.7.3-REQ-1: Only authorized users, such as *parking enforcement officers*, can view any *customer's* booking details
4.7.3-REQ-2: *Customers* can view their parking space booking information, including expiry time
4.7.3-REQ-3: The system must notify the *customer* when their parking space booking is expired

4.8 Manage Parking Spaces

Priority: high

4.8.1 Description

This feature allows the *parking enforcement officer* to manage parking spaces by adding or removing them.

4.8.2 Stimulus/Response Sequences

- Authorized *parking enforcement officers* will be displayed a button labeled “MANAGE PARKING”, leading to a separate window displaying all parking spaces.
- *Parking enforcement officers* then can select any parking space and view information such as *customer* space requests and current user booking (if any)
- With buttons such as “ADD SPACE”, “REMOVE SPACE”,“CANCEL REQUEST”, “GRANT REQUEST”, parking enforcement officers can add, remove, cancel, and grant requests respectively

4.8.3 Functional Requirements

4.8.3-REQ-1: The user must be an authenticated *parking enforcement officer.*
4.8.3-REQ-2: *Parking enforcement officers* must verify a parking space is vacant before removing it from the system
4.8.3-REQ-3: The system must have a minimum of one parking space
4.8.3-REQ-4: *Parking enforcement officers* must verify requested parking space is vacant in the system before granting requests

4.9 Change Payment Status

Priority: high

4.9.1 Description

This feature allows the *systems administrator* to update *customers'* payment status.

4.9.2 Stimulus/Response Sequences

- *System administrator* will be displayed with a button labeled “CHANGE PAYMENT STATUS”, leading to a separate window, where they can enter *customer* first name, last name, email address and parking space number to change their payment status to “paid”.

4.9.3 Functional Requirements

4.9.3-REQ-1: The user must be an *systems administrator.*
4.9.3-REQ-2: The system must verify the *customer’s* existence before changing their payment status.
4.9.3-REQ-3: The system must verify the *customer’s* occupancy of the parking space before changing their payment status
4.9.3-REQ-4: The system must verify the *customer’s* payment of the parking space before changing their payment status

5. Other Nonfunctional Requirements (Satinder)

5.1 Performance Requirements

Traffic
As this is a parking based application, this application will have specific times where there are large loads of traffic/requests coming in (e.g. morning office rush, etc.), and must be able to handle incoming large loads of traffic and handle them efficiently, without being overwhelmed and shutting down, leading to a denial of service. Response times should be reasonable, most communications should happen within a few milliseconds to at most a few seconds (3 max).

Payment
A user should be able to pay for their parking space and allotted time, so there should be viable options of payment considering (paypal, e-transfer, etc.). As payment is a rather sensitive and worrisome matter, users should be asked to make sure that they wish to use a certain payment method, pay a certain amount, and so on. On our end, the payment should take a few seconds (not including time needed for third parties to confirm the amount and method of payment).

Platforms
The application should be able to run on both MacOS and Windows platforms, and it's design should reflect that. Design should be intuitive to understand, as we do not want to make the process of payment and selection take too much time, becoming an inconvenience for users. Response timings should be similar across platforms. No function in any platform should take more than a few seconds (3 at most).

The reason for having 3 seconds at most for any function's time constraint, is because most computers can complete the steps of the program in a matter of milliseconds. Communication with other components (such as databases) brings in a latency, which most users experience in other applications for at most a few seconds. In our case, 3 is a reasonable amount of time.

5.2 Safety Requirements

As there is a large overlap between Safety and Security, please refer to Section 5.3.

5.3 Security Requirements

Payment
As payment is a rather sensitive issue, we must ensure that no identifying information is passed as raw text data (e.g. passwords and credit card numbers). Safekeeping of sensitive information and other usages of data should adhere to the Privacy Act and MFIPPA.

Login/User Identifying Information
The user's information (address, email, password) must not be stored and communicated in plain text format. As attackers/malicious individuals could try and get this information for malicious purposes.

Sanitizers
In general, user input (or places that the user can affect) cannot be trusted, and must always be sanitized or have some sort of restriction in place to prevent malicious input.

5.4 Software Quality Attributes

Usage
The application should be easy to use and should have an intuitive interface. *Parking enforcement officers* and regular *customers* should have different features available to them (more on this can be found in the use cases section of this document). Another essential feature should be that the program can run on different devices (web-browsers) regardless of platform. These things should be considered before launching the application for production.

5.5 Business Rules

Regional Guidelines
Since the application scope is limited to Toronto parking spaces (falling under the Ministry of Transportation), it must adhere to the Privacy Act and be MFIPPA compliant.

6. Other Requirements

N/A

Appendix A: Glossary

MFIPPA: Municipal Freedom of Information and Protection of Privacy Act