Software Requirement Specifications

ParkIT

# 

# Submitted By

Nauman Khan CS-142004

Hummam Shah CS-151137

Maaz Hussain CS-142026

# 

# Supervisor

Engr. Farooq Shibli

# Co-Supervisor

Naushad Tabani

# 

# Fall 2018

# Revision Chart

This chart contains a history of this document’s revisions. The entries below are provided solely for illustration purposes. Those entries should be deleted until the revision/s they refer to have been created.

The document itself should be stored in revision control, and a brief description of each version should be entered in the Revision Control System. A brief description can be repeated in this section. Revisions need not be described elsewhere in the document, unless they explain the document.

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Primary Author(s)** | **Description of Version** | **Date Completed** |
| Draft | TBD | Initial draft created for distribution and review comments | (To be decided)  TBD |
| Preliminary | TBD | Second draft incorporating initial review comments, distributed for final review | TBD |
| Final | TBD | First complete draft, which is placed under change control | TBD |
| Revision 1 | TBD | Revised draft, revised according to the change control process and maintained under change control | TBD |
| Revision 2 | TBD | Revised draft, revised according to the change control process and maintained under change control | TBD |
| Etc. | TBD | TBD | TBD |

# Table of Contents

[**1. Introduction**](#_ijyqvogjn4bi) **5**

[1.1. Purpose of Document](#_dib7eg1uqrle) 6

[1.2. Intended Audience](#_2cib95hco7fn) 6

[1.3. Abbreviations](#_ckgl5jhsmrfg) 6

[**2. Overall System Description**](#_xjbl97duy9jm) **7**

[2.1 Project Background](#_gy8kzz7baxeb) 7

[2.2 Problem Statement](#_jl4rbtb65gbz) 7

[2.3 Project Scope](#_k3ic1ckfekc8) 7

[2.4 Not In Scope](#_5ijsnici9wc) 7

[2.5 Project Objectives](#_mwo9bgye676r) 7

[2.6 Stakeholders & Affected Groups](#_lvhm80z605ez) 7

[2.7 Operating Environment](#_qmvtzp5m0c9v) 7

[2.8 System Constraints](#_b3odawbewzbg) 7

[2.9 Assumptions & Dependencies](#_cpbta77gk84q) 7

[**3. External Interface Requirements**](#_1pbslzdlducs) **8**

[3.1 Hardware Interfaces](#_qzp4tmgkpxj) 8

[3.2 Software Interfaces](#_gy5qmcuejutj) 8

[3.3 Communications Interfaces](#_o56xyxrttspj) 8

[**4. System Functions / Functional Requirements**](#_v3zehylmrnis) **9**

[4.1 System Functions](#_y0nlbzlendg7) 9

[4.1.1 System Attributes/ Nonfunctional Requirements](#_bn6dng7isjtd) 10

[4.2 Use Cases](#_bw9be3hp8vao) 11

[4.2.1 List of Actors](#_4pknb781yve0) 11

[4.2.2 List of Use Cases](#_kvv8ggqiga14) 12

[4.2.3 Use Case Diagram](#_99s02ggvvsre) 12

[4.2.4 Description of Use Cases](#_my2lwmfxpvc2) 12

[**5. Non - Functional Requirements**](#_7x778uz9lcxv) **15**

[5.1 Performance Requirements](#_8akjdnrdp9rh) 15

[5.2 Safety Requirements](#_s4y24rty3x52) 15

[5.3 Security Requirements](#_we4h3by1o9br) 15

[5.4 Reliability Requirements](#_j3w42uxe39n7) 15

[5.5 Usability Requirements](#_7udmsvoe195h) 15

[5.6 Supportability Requirements](#_1fn8js12tao3) 15

[5.7 User Documentation](#_uh8hzhybx3vp) 15

[**6. References**](#_d7fr76tcu8vt) **16**

# 

# 

# 1. Introduction

This section should describe the project and the software product being to be built. No text is necessary between the heading above and the heading below unless otherwise desired.

## 1.1. Purpose of Document

## 1.2. Intended Audience

The purpose of this project is to …

The problem statement should be brief, comprising of no more than 50 words

## 1.3. Abbreviations

Provide definitions or references to all the definitions of the special terms and acronyms used within this document.

# 2. Overall System Description

## *2.1 Project Background*

## As per the research there are very less or zero parking spaces available in Karachi.

## *2.2 Problem Statement*

## Nowadays parking is a serious issue in Pakistan as there are very less parking slots available for people to park their cars and due to which people are forced to park their cars on the streets which cause unnecessary traffic jams on the streets. People find difficulties in finding proper secure parking slots. First of all it takes too much time to find the slot and there is so no guarantee that you will get your car without any damage.

## *2.3 Project Scope*

*In Scope*

* Simple user interface
* Find parking slot
* Leave parking slot
* Add parking slot/Delete parking slot
* Renter’s record (one who is renting out his/her garage)
* Driver’s/Car’s record (who is parking his/her car).

*Out Scope*

* Not covering any other vehicle except cars and bikes (only small vehicles).

## *2.4 Not In Scope*

## *2.5 Project Objectives*

* To provide safe and secure parking slot.
* We are providing a way to earn easy money for those who have vacant parking space.
* Avoiding unnecessary traffic jam due to illegal parking on the streets.

## *2.6 Stakeholders & Affected Groups*

## *2.7 Operating Environment*

## *2.8 System Constraints*

## *2.9 Assumptions & Dependencies*

We are assuming that there are large number of peoples who have vacant garage at least for some time period as most people go to work on their cars leaving the garage empty for 8 to 9 hours and a lot of people don’t have cars so there garage are always available.

* Our application is depending on the number of peoples willingness to rent out their garage
* On first it’s available on android platform only.

# 3. External Interface Requirements

## 3.1 Hardware Interfaces

Define hardware interfaces supported by the system, including logical structure, physical addresses, and expected behavior.

## 3.2 Software Interfaces

Name the applications with which the subject application must interface. State the following for each such application: name of application, external owner of application, interface details (only if determined by the other application).

## 3.3 Communications Interfaces

Describe communications interfaces to other systems or devices, such as local area networks.

## 

# 4. System Functions / Functional Requirements

## 4.1 System Functions

This section may contain

• end user, operator, support, or integration functions,

• performance requirements,

• design constraints, • programming language, and • interface requirements.

System functions are descriptions of what a system is supposed to do. They should be identified and listed in logical cohesive groups, with their category (priority) assigned. These system functions will be identified as a result of the requirement gathering process conducted with the client. However, in some cases, prior to the development of the Functional Specifications the requirements may already have been listed in a document: if this is so then a reference to the document may suffice.

To verify that some X is indeed a system function; it should make sense in the following sentence:

The system should do <X>

The table below gives an example of how system functions can be listed:

▪ The Functions column gives a brief one-line description of the required functionality.

▪ The Category column refers to the status of the functionality for the proposed system. The options for the Category are defined below.

▪ The Attribute column defines the system characteristics. The Details and Constraints column specifies the conditions within which the attribute is applicable. Section 1.12 defines the default Attributes and the related Constraints. In case, the default conditions are to be over-ridden then the conditions can be defined in this table.

Table 4.1: Function Categories

|  |  |
| --- | --- |
| **Function Category** | **Meaning** |
| Evident | Should perform, and user should be cognizant that it is performed. |
| Hidden | Should perform, but not be visible to users. This is true of many underlying technical services, such as save information in a persistent storage mechanism.  Hidden functions are often missed during the requirements gathering process. |
| Frill | Optional; adding it does not significantly affect cost or other functions. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ref # | Functions | Category | Attribute | Details & Boundary Constraints |
| R1.1 | Record the underway sale – the items purchased | Evident | System Response time | Price listing within 3 seconds  Availability agreement in less than 10 sec |
| R1.2 | Reduce inventory quantities when a sale is committed | Hidden | Concurrent user load |  |
| … | … | … |  |  |

## 

### 4.1.1 System Attributes/ Nonfunctional Requirements

System attributes are nonfunctional system qualities – such as ease of use. System attributes are characteristics of the system; they are not functions.

System attributes have a possible set of Attribute Details, which tend to be discrete, fuzzy, symbolic values of the attribute, such as: response time = psychologically appropriate interface metaphor = graphical, browser-based

Some system attributes may also have Attribute Boundary Constraints, which are mandatory boundary conditions, usually on a numeric range of values of an attribute, such as: response time = five seconds maximum

In this section the Category column indicates whether or not the attribute is critical for the operation of the system.

The Category can take two options:

▪ Optional

▪ Mandatory

|  |  |  |
| --- | --- | --- |
| Attribute | Details and Boundary Constraints | Category |
| Response time | (Boundary constraint) When recording a sold item, the description and price will appear within 5 seconds | Optional |
| Concurrent User Load | A minimum of 10 users connected simultaneously | Mandatory |

## 4.2 Use Cases

Describe the following items:

▪ Actors & use cases

▪ Use case diagrams

▪ High level, essential use cases

No text is necessary between the heading above and the heading below unless otherwise desired.

### 4.2.1 List of Actors

Define the system boundary and list all actors with the use cases.

For example:

Cashier; this person performs all the financial activities

Account Manager; this person supervises all financial activities

### 4.2.2 List of Use Cases

List all the use cases, with a brief description (should not exceed two lines):

### 4.2.3 Use Case Diagram

Create the system level use case diagram

### 4.2.4 Description of Use Cases

Document each use case. This can completed using the tables provided below:

|  |  |
| --- | --- |
| Section: | Main |
| Name: | Buy Item |
| Actors: | Customer, Cashier |
| Purpose: | Capture a sale and its payment. |
| Description: | A customer arrives at a checkout with items to purchase. The cashier records the purchase items and collects a payment. On completion, the customer leaves with the items. |
| Cross References: | Functions: R1.1, R1.2 |

Use Cases: Cashier must have completed the Log In use case. This is a reference to the System Functions as described in Section 1.10 Pre-Conditions Assumption about the state of the system before execution of the operation Successful Post-Conditions State of the system after completion of the operation. Failure Post-Conditions State of the system after completion of the operation.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Typical Course of Events** | | |
|  | Actor Action |  | System Response |
| 1 | This use case begins when a customer arrives at the Point of Sale checkout with items to purchase. |  |  |
| 2 | The cashier records each item | 3 | Determines the item price and adds the item information to the running sales transaction. |
| 4 | … | 5 | … |
| 7 | Customer selects payment type:  a. If cash payment, see section Pay by Cash  b. If credit payment, see section Pay by Credit |  |  |
|  |  | 8 | Logs the completed sale |
|  |  | 9 | Updates inventory levels |
|  |  | 10 | Generates a receipt |
| 11 | Cashier gives the receipt to the customer |  |  |
| 12 | The customer leaves with the items purchased |  |  |

Alternative Course

Step 2: Invalid item identifier entered. Indicate error.

Step 7: Customer could not pay. Cancel sales transaction

Section: Pay by Cash

Typical Course of Events

|  |  |  |  |
| --- | --- | --- | --- |
|  | Actor Action |  | System Response |
| 1 | The customer makes a cash payment |  |  |
| 2 | The cashier records the cash tendered | 3 | Presents the balance due back to the customer, if any. |
| 4 | The Cashier deposits the cash received and extracts the balance owing and gives it to the customer |  |  |

Alternative Courses

|  |  |
| --- | --- |
| Step 1: | Customer does not have sufficient cash, may cancel sale or initiate another payment method. |
| Step 4: | Cash drawer does not contain sufficient cash to pay balance. |

# 5. Non - Functional Requirements

## 5.1 Performance Requirements

## 5.2 Safety Requirements

## 5.3 Security Requirements

## 5.4 Reliability Requirements

## 5.5 Usability Requirements

## 5.6 Supportability Requirements

## 5.7 User Documentation

# 6. References

List of references