**Software Requirements Specification**

**For**

**Motoroid**

**Version 0.1**

**Submitted to**

**Md. Shohel Arman**

**Lecturer**

**Department of SWE**

**Daffodil International University**

**Prepared by**

**S.M. Rabby Hasan**

**Id: 171-35-1910**

**Table of Contents**

**Table of Contents i**

**List of Figures ii**

**1. Introduction 1**

1.1 Purpose 1

1.2 Project Scope 1

1.3 Glossary 1

1.4 References 1

1.5 Overview 2

**2. User Classes and Characteristics 2**

**3.** **Design and Implementation Constraints 2**

3.1 User Interface Technology 2

3.1.1 Programming Language 3

3.1.2 Java Swing 3

**4. Use Case Diagram 4**

**5. Requirement Specification 5**

5.1 Functional Requirements 5

5.2 Performance Requirements 5

5.2.1 Capacity Requirements 6

5.3 Dependability Requirements 6

5.3.1 Reliability and Availability 6

5.3.2 Safety Critical Requirements 6

5.4 Maintainability and Supportability 7

5.4.1 Maintenance Requirements 7

5.4.2 Supportability Requirements 7

5.4.3 Adaptability Requirements 7

5.5 Usability and Human Integrity Requirements 7

5.5.1 Ease of Use Requirements 8

5.5.2 Understand-ability and Politeness Requirements 8

5.5.3 Accessibility Requirements 8

5.5.4 User Documentation 9

5.6 Operational and Environmental Requirements 9

5.6.1 Expected Physical Requirements 9

5.6.2 Requirement for Interfacing with Adjacent System 9

5.6.3 Release Requirements 9

5.7 Legal Requirements 9

5.7.1 Compliance Requirements 9

5.7.2 Standard Requirements 10

**6. Requirement Engineering Process 10**

6.1 Requirement Elicitation Techniques 10

6.1.1 Hold Elicitation Interviews 10

6.1.2 Perform Document Analysis 10

6.1.3 Distribute Questionnaires 10

6.2 Requirement Validation 11

6.2.1 Review the Requirements 11

6.2.2 Test the Requirements 11

6.2.3 Simulate the requirements 11

6.3 Change Management 11

**List of Figures**

Figure 4.1 – Use Case Diagram for Motoroid.

Figure 6.1 – State Diagram of Change Request

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Date | Reason For Changes | Version |
|  |  |  |  |
|  |  |  |  |

**1. Introduction**

A Software Requirements Specification (SRS) is a description of a software system to be developed. The introduction of Software Requirements Specification provides an overview of the entire SRS with purpose, scope, glossary, references and overview of the SRS. Software requirements specification is a rigorous assessment of requirements before the more specific system design stages, and its goal is to reduce later redesign. The software requirements specification document lists sufficient and necessary requirements for the project development.

**1.1 Purpose**

The purpose of this document is to give a detailed description of the requirements for the “Motoroid” software. It will illustrate the purpose and complete declaration for the development of system. It will also explain system constraints, interface and interactions with other external applications. This document is primarily intended to be proposed to a customer for its approval and a reference for developing the first version of the system.

**1.2 Project Scope**

The Motoroid is mobile and web based application which helps people to find their suitable ride for getting rent. It’ll also help the providers to rent their vehicle. The application should be free to download from either a mobile phone application store or similar services.

The vehicle owners can give rent their vehicle by uploading their vehicle info through the application.

**1.3 Glossary**

This subsection contains definitions of all the terms, acronyms, and abbreviations used in the document. Terms and concepts from the application domain are defined.

* SRS- Software Requirements Specification
* SDLC- Software Development Life Cycle
* UI- User Interface

**1.4 References**

IEEE. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.

**1.5 Overview**

The system is designed for both user and providers. Those who need vehicles but hadn’t one they can easily borrow vehicle through this system. And those who want to give rent their vehicle, they can give rent to them who want to rent. But the customer must visit the provider for further procedure. Both of them need to give information as the system required.

**2. User Classes and Characteristics**

There are only three types of user in this system. Only the admin can see both of other two user information. It is the third type of user, the administrator, who is able to initially setup the system, and set the authorization level.

**User:** Those who need to take vehicle in rent are the user. User can only see the vehicle information and providers shop location and also contract number.

**Provider:** Those who want give their vehicle in rent are the providers. They can only give information about their vehicle and shop.

**Admin:** Admin can remove any type of user and providers. Admin can also be able to initially setup the system and set the authorization level

**3. Design and Implementation Constraints**

Design and implementation constraints are those that i have used to implement this project make successful. It also describes tool that enables developers and testers to view and interact with the user interface (UI) elements of this application.

**3.1 User Interface Technology**

User interface (UI) is everything designed into a system view that which person’s associates with this system may like the interface of this system.

**3.1.1 Programming Language**

For developing this system i will use Java as a programming language. Java is a general-purpose computer-programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation.

**3.1.2 Java Swing**

Swing is a GUI widget toolkit for Java. It is part of Oracle's Java Foundation Classes (JFC) – an API for providing a graphical user interface (GUI) for Java programs. Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit (AWT). Swing provides a look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform.

**4. Use Case Diagram**

User

Admin provider

Figure 4.1 – Use Case Diagram for Motoroid.

**5. Requirement Specification**

The complete requirement specification based on the elicitation process is described in this section.

**5.1 Functional Requirements**

The Functional Requirements Specification is designed to be read by a general audience. Readers should understand the system, but no particular technical knowledge should be required to understand the document.

|  |  |
| --- | --- |
| **FR-01** | **Member Information** |
| **Description** | This module helps admin to register user and provider. Admin is able to maintain all the information of user and provider |
| **Stakeholders** | Admin |

|  |  |
| --- | --- |
| **FR-02** | **Delete Information** |
| **Description** | Admin can delete any type of information of user and provider |
| **Stakeholders** | Admin |

|  |  |
| --- | --- |
| **FR-03** | **View user and provider details** |
| **Description** | Admin can see any type of information of users and providers. |
| **Stakeholders** | Admin |

**5.2 Performance Requirements**

A requirement that specifies a performance characteristic that a system or system component must possess.

**5.2.1 Capacity Requirements**

The system is able to manage all the information of users and providers.

|  |  |
| --- | --- |
| PR-01 | Initially the system will store 30,000 of users and providers information |
| Description | The information will store in Database |
| Stakeholders | Admin |

**5.3 Dependability Requirements**

The flexibility of current frameworks encourage system architects to enable reconfiguration mechanisms that refocus the available, safe resources to support the most critical services rather than over-provisioning to build failure-proof system. Therefore, these requirements are essentials.

**5.3.1 Reliability and Availability**

In order to support global and smooth operations the system must be available around the clock. On the other hand most services in this system are not mission-critical. Even better the game posting can handle times of downtime as the users usually interact with high- availability from third party website. This system will be able to catch up with their data once it’s up and running again.

|  |  |
| --- | --- |
| DR-01 | The system must be available 24x7. |
| Description | * The system must be available 24 hours in a day. * The system must be updated regularly |
| Stakeholders | Admin |

**5.3.2 Safety Critical Requirements**

There are no specific safety critical requirements.

**5.4 Maintainability and Supportability**

Supportability is the degree to which system design characteristics and planned logistics resources meet system requirements. Supportability is the capability of a total system design to support operations and readiness needs throughout the life-cycle of a system at an affordable cost.

**5.4.1 Maintenance Requirements**

|  |  |
| --- | --- |
| **MS-01** | The system helps to update any information in any time |
| **Description** | The admin can enable to change or update any information in any situation |
| **Stakeholders** | Admin |

**5.4.2 Supportability Requirements**

In order to understand the system’s behavior on a technical support required by the system operator. The reason for reading them might be

* System malfunction has occurred and the system operator has to find the exact point of time when this happened.
* System produces wrong results and the developers must be able to reproduce the data flow through the system.
* Hacker tried to breach the system’s security mechanisms and the system operator must understand what he did.

**5.4.3 Adaptability Requirements**

There are no specific adaptability requirements.

**5.5 Usability and Human Integrity Requirements**

These Requirements define how to meet the physical and cognitive needs of the intended users of your website or application

**5.5.1 Ease of Use Requirements**

The system is easy to use and can easily be understandable.

|  |  |
| --- | --- |
| **UH-01** | The system must be usable for Admin with all associate stakeholders. |
| **Description** | The system indicates the several possibilities that the admin has to go on in using the system. The admin are allowed to undo any of the operation. |
| **Stakeholders** | Admin, User, Provider |

**5.5.2 Understand-ability and Politeness Requirements**

This section describes more requirements of Motoroid system to add more features in future.

|  |  |
| --- | --- |
| **UH-02** | The features of Motoroid system. |
| **Description** | The system is more efficiently ease of use more added features .The system is understand-ability for both user. The system will not use any term that is not specified in this system. |
| **Stakeholders** | Admin |

**5.5.3 Accessibility Requirements**

There are no access requirements beside those that have been outlined in the below:

* AR-01: Login as Admin.
* AR-02: Login as User.
* AR-03: Login as Provider.
* AR-04: Logout as Admin.
* AR 05: Logout as User.
* AR 06: Logout as Provider.

To get access to this system or a specific module the system must provide a central authentication mechanism. In order to prevent anyone to exploit stolen all users password must be encrypted in hash process.

**5.5.4 User Documentation**

|  |  |
| --- | --- |
| **UH-03** | The system developer documentation. |
| **Description** | To develop this project we have specified requirement of user documentation. The teams are involved to this project documentation. |
| **Stakeholders** | System Developer |

**5.6 Operational and Environmental Requirements**

This requirements focus on how the users will operate the system, including interfaces and interoperability with other systems. The requirements establish how well and under what conditions the system must perform.

**5.6.1 Expected Physical Requirements**

There are no specific expected physical requirements.

**5.6.2 Requirement for Interfacing with Adjacent System**

There is no specific interfacing with adjacent system requirements.

**5.6.3 Release Requirements**

There are no specific release requirements but in the project schedule section it was described briefly.

**5.7 Legal Requirements**

These requirements consider any violence of rules and regulation and which rules should be followed to maintain this system.

**5.7.1 Compliance Requirements**

There are no specific compliance requirements.

**5.7.2 Standard Requirements**

There are no specific standard requirements.

**6. Requirement Engineering Process**

Requirements engineering refers to the process of defining, documenting and maintaining requirements in the engineering design process. It is a common role in systems engineering and software engineering.

**6.1 Requirement Elicitation Techniques**

Requirement elicitation is the process of collecting and refining stakeholder’s requirements. Projects are garbage-in-garbage-out meaning that poor quality requirements typically lead to project issues and failure.

**6.1.1 Hold Elicitation Interviews**

I hold interviews that can be performed one-on-one or with a small group of stakeholders. They are an effective way to elicit requirements without taking too much stakeholder time because i meet with people to discuss only the specific requirements that are important to this system. Interviews are helpful to separately elicit requirements from members in preparation for workshops where those member of this system come together to resolve any conflicts.

**6.1.2 Perform Document Analysis**

Existing documentation can help reveal how systems currently work or what they are supposed to do. Documentation includes any written information about current systems, business processes, requirements specifications, competitor research. Reviewing and analyzing the documents can help identify functionality that needs to remain, functionality that isn’t used.

**6.1.3 Distribute Questionnaires**

We conduct a survey to collect requirements for this system. Questionnaires are a way to survey large groups of users to determine what they need. Questionnaires are useful with any large user population but are particularly helpful with distributed groups.

**6.2 Requirement Validation**

Validation ensures that the requirements are correct and demonstrate the desired quality that you want from this system. Requirements that seem fine when you read them might turn out to have ambiguities and gaps when to try to work with them.

**6.2.1 Review the Requirements**

Peer review of requirements, particularly the type of rigorous review called inspection, is one of the highest-value software quality practices available. Assemble a small team of reviewers who represent different perspectives and carefully examine the written requirements, analysis models, and related information for defects.

**6.2.2 Test the Requirements**

We tests constitute an alternative view of the requirements. We also conduct writing tests about how to tell if the expected functionality was correctly implemented. Derive tests from the user requirements to document the expected behavior of the product under specified conditions.

**6.2.3 Simulate the requirements**

To simulate the requirements commercial tools are available that we have used to simulate a proposed system either in place of or to augment written requirements specifications. Simulation takes prototyping to the next level.

**6.3 Change Management**

We used a common set of web-based tools for handling change requests and tracking open issues is essential. Change always has a price, so using change management practices to control scope creep is vital in a contract-development situation. We will provide these following issues in change management.

* Evaluate and prioritize defect corrections and enhancement requests
* Dynamically adjust the scope of future releases or iterations
* Evaluate the impact of proposed changes on users and business processes
* Participate in making change decisions

Done

**Committed**

**Removed**

**Approved**

**New**

Removed

Approved by the Moved to the from the

Product owner backlog backlog

Commitment made Work

By the team Stopped

Work Additional Work

Finished Found

Fig 6.1: State Diagram of Change Request