Software Requirements Specification

for

Near By Taxi

Version 1.0 approved

Prepared by

Kopil Das

Md. Abdul Mutalib

17-07-2022

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

Software Requirements Specification (SRS) simply contains all necessary information of project. The purpose of SRS is to give description all abouts of project design, develop and testing the software.

- The purpose of the project is to provide a user-friendly Taxi Service for the transportation.
- The main purpose of this is project is to manage an easy system for transportation.

1.2 Document Conventions

Entire document should be justified.

➔ Convention for main title:

- ♦ Font Face: Times New Roman
- ♦ Font Style: Bold
- ♦ Font Size: 18

➔ Convention for subtitle:

- ♦ Font Face: Times New Roman
- ♦ Font Style: Bold
- ♦ Font Size: 32

➔ Convention for body:

- ♦ Font Face: Times New Roman
- ♦ Font Style: Normal
- ♦ Font Size: 12

1.3 Intended Audience and Reading Suggestions

- ♦ Admin
- ♦ General User
- ♦ All User

1.4 Product Scope

The document only covers the requirements specifications for the Taxi Booking System. This document does not provide any references to the other component of the Taxi Booking System. All the external interfaces and the dependencies are also identified in this document. Taxi Booking System is basically updating the manual ticket booking system into an internet-based application so that the users can know the availability of Taxi and can book from the system. The project is specifically designed for the general people & taxi driver for booking taxi. The project will work as a complete user interface for Taxi Booking process and complete the transition. The project can be easily implemented under various situations. We can add new features as and when we require, making reusability possible as there is flexibility in all the modules. The language used for developing the project is PHP, Java Script, HTML5, CSS 3, AJAX, and all of these are quite advantageous than other languages in terms of performance, tools available, cross platform compatibility, libraries, cost (freely available & open source), and development process.

1.5 References

- PHP: http://www.phptherightway.com/
- HTML5: http://www.w3schools.com/
- CSS3: http://www.w3schools.com/
- JAVA Script: http://www.w3schools.com/
- AJAX: http://stackoverflow.com/
- MySQL: https://www.mysql.com/

2. Overall Description

2.1 Product Perspective

The implementation of NearBy Taxi starts with entering and updating master records like taxi details, taxi route information. Any further transaction like ticket book issue will automatically update the current information's. The proposed NearBy Taxi System will take care of the current near by taxi detail at any location of time. The book issue, book return will update the current book details automatically so that user will
get the update current book details.

2.2 Product Functions

- The main purpose of this project is to provide an online taxi service.
- This software is capable to show taxi information, available taxi , booking taxi and the cost of taxi service.

2.3 User Classes and Characteristics

We have 2 levels of users: -

- User module: In the user module, user will check the availability of taxies.
  - Taxi Order
  - Comment

- Administrative Module: The following are sub module in the administration module.
  - Register User
  - Update User
  - Post Taxi
  - Update Taxies
  - Update, Edit or Delete of any records.

2.4 Operating Environment

The product will be operating in windows environment. The NearBy Taxi system is a website and shall operate in all famous browsers, for a model we are talking Microsoft Internet Explorer, Google Chrome, and Mozilla Firefox. Also, it will be compatible with the IE 6.0. Most of the features will be compatible with the Mozilla Firefox and Opera 7.0 or higher version. The only requirement to use this online product would be the internet connection. The hardware configuration includes Hard Disk: 40GB, Monitor: 15-inch Color monitor, Keyboard: 122 keys. The basic input devices required are keyboard, mouse and output devices are monitor etc.

2.5 Design and Implementation Constraints

Each user will be having an identity number which can be used for the booking taxi. whenever user wish to book a taxi by, the authority will be check both the taxi details as well as the user details and store it in taxi database. In case of retrieval of taxi much of human intervention can be eliminated.

2.6 User Documentation

The product will include user manual. The user manual will include product overview, complete configuration of the used software (such as SQL server), technical details, backup procedure and contact information which will include email address. There will be no online help for the product at this moment. The product will be compatible with the Internet Explorer 6.0 or higher. The databases will be created in the MySQL.

2.7 Assumptions and Dependencies

The assumptions are: -

1) The coding should be error free.
2) The system should be user friendly so that it is easy to use for the users. Software Requirements Specification for NearBy Taxi Page 5
3) The information of all users, taxi info must be stored in a database that is accessible by the website.
4) The system should have more capacity and provide fast access to the database.
5) The system should provide search facility and support quick transactions.

6) The Taxi system is running twenty-four hours a day.
7) Users may access from any computer that has internet browsing capabilities and an internet connection.

The dependencies are: -

1) The specific hardware and software due to which the product will be run.
2) On the basis of listing requirements and specification the project will be develop and run.
3) The end users (admin) should have proper understanding to the product.
4) The system should have the general report store.
5) The information of all users must be stored in a database that is accessible by the library system. 6) Any update regarding the Taxi is to be recorded to the database and the data entered should be correct.

3. External Interface Requirements

3.1 User Interfaces

The software provides good graphical interface for the user. And admin can operate on the system.

- User can comment from the user panel.
- User can view the taxies availability & registry details.
- Admin can View, Edit and Delete everything on the product.
- Software Requirements Specification for NearBy Taxi

Home of user panel:
Figure/Screen shot/Image
 Comment box in user panel

3.2 Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

3.3 Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

3.4 Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

4. System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

4.1 System Feature 1

<Don't really say "System Feature 1." State the feature name in just a few words.>

4.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

4.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

4.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use "TBD" as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:
REQ-2:

4.2 System Feature 2 (and so on)

5. Other Nonfunctional Requirements

5.1 Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

5.2 Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied.>

5.3 Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

5.4 Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

5.5 Business Rules

<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>

6. Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>