ANDREI BRANGA
30432

Online Hotel Booking Application
Supplementary Specification

Version 1.0

Revision History

Date	Version	Description	Author
09/MAY/2018	1.1	Added specification	Andrei Branga

Table of Contents

Supplementary Specification

1. Introduction

The **Supplementary Specification** captures the system requirements that are not readily captured in the use cases of the use-case model. Such requirements include:

Legal and regulatory requirements, including application standards.

Quality attributes of the system to be built, including usability, reliability, performance, and supportability requirements.

Other requirements such as operating systems and environments, compatibility requirements, and design constraints.

2. Non-functional Requirements

In software system engineering, a software requirement that describes not what the software will do, but how the software will do it, for example, software performance requirements, software external interface requirements, design constraints, and software quality attributes. Nonfunctional requirements are difficult to test.

2.1 Availability

Does the system need to be available all the time, and to everyone? Most of the time you can get away with the occasional vacation, but knowing what’s important helps you make good decisions. Quality software ain’t quality if its down when your precious users need it.

2.2 Performance

This one shouldn’t come as a surprise. Quality software has to be fast. Or at least feel fast. “It’s not fast enough” is a battle no wants to get into. It is hard to specify specify what “fast enough” was; talk about moving goalposts. When you think about an app being performant, thing about specifying the following:

2.3 Security

Security covers a lot of areas of the application. From logging in the application, keeping a session open and secure, maintaining a secure connection, to keeping the users’ precious data safe.

2.4 Testability

Testability is a non-functional requirement important to the testing team members and the users who are involved in user acceptance testing. It can be defined as the property that measures the ease of testing a piece of code or functionality, or a provision added in software so that test plans and scripts can be executed systematically.

2.5 Usability

A system can have adequate functionality, but inadequate usability because it is too difficult to use. The purpose of usability requirements is to guard against that. This paper shows six styles for usability requirements seen in practice or recom- mended by experts. For each style we discuss how we can verify the requirements, how we can use them during development, how we elicit the data for the specification, and to what extent the style covers the essence of usability.

3. Design Constraints

This section needs to indicate any design constraints on the system being built. Design constraints represent design decisions that have been mandated and must be adhered to. Examples include software languages, software process requirements, prescribed use of developmental tools, architectural and design constraints, purchased components, class libraries, and so on.

- The main language used for developing the application is C#
- In the application development the .NET Framework is used
- As architectural pattern, the Layered Architecture was chosen for the development process
- MVC will be used as the main framework for developing the application
- Javascript will be the main language used for client-side events