VehicleRentingSystem

System Design

SYSTEM DESIGN DOCUMENT[1]

15.12.2019

Ahmet Can Terlikçi

Tunç Bora Tamsan

Duygu Genç

Bora Arseven

Prepared for

SE301 Software Engineering



Table of Contents

[1. Introduction 1](#_Toc433996772)

[1.1. Purpose of the System 1](#_Toc433996773)

[1.2. Design Goals 1](#_Toc433996774)

[1.3. Definitions, Acronyms, and Abbreviations 1](#_Toc433996775)

[1.4. References 1](#_Toc433996776)

[2. Current Software Architecture 1](#_Toc433996777)

[3. Proposed Software Architecture 1](#_Toc433996778)

[3.1. Overview 1](#_Toc433996779)

[3.2. System Decomposition 1](#_Toc433996780)

[3.3. Hardware Software Mapping 2](#_Toc433996781)

[3.4. Persistent Data Management 2](#_Toc433996782)

[3.5. Access Control and Security 2](#_Toc433996783)

[3.6. Global Software Control 2](#_Toc433996784)

[3.7. Boundary Conditions 2](#_Toc433996785)

[4. Subsystem Services 2](#_Toc433996786)

[5. References 2](#_Toc433996787)

SYTEM DESIGN DOCUMENT[1]

# Introduction

## Purpose of the System

In this document what we aim is to present a detailed description of the design process of VehicleRentingSystem. And this design process consisted of architectural features such as subsystem decompositions, hardware/software mapping, database management etc. to show the developers how the functional, non-functional requirements defined on previous RAD documents should be implemented and which architectural decisions taken by architects to create a consistent and responsive web-based car renting system of VehicleRentingSystem.

## Design Goals

The system is designed to any kind of users including a visitor to view the available, rentable cars by selecting corresponding filtering options and

* Ease of Use:

A person which is never use any vehicle renting system before, must be able to learn how to the system by simply renting a vehicle from VehicleRentingSystem in a short amount of time which is considered about 15 minutes.

* Well-defined interfaces:

The system should be implemented as a responsive website in terms of its layout design, the auto-scaled pages, visibility of the various interface elements on page should be achieved in a way to draw user’s attention quickly, eg. The search panel should be placed on the main page (non func req Usability 4)

* User-friendliness

The search result must be displayed with corresponding images of vehicles which will make the whole system as aesthetically pleasing to user.

Search results should be displayed up to 10 results per page and displayed with textual vehicle information such as baggage capacity, the number of seats etc alongside with visual representative images. This way, user can recognize and identify the vehicles easily and it’ll help to reduce the time in making decisions for user.

* End-User:

After searching, the unavailable vehicles such as vehicles with “damaged” physical status must not be displayed in advance to prevent user to rent this type of vehicles.

* Performance:

The search function in the system must be display a result at most 30 seconds to the user.

VehicleRentingSystem must send e-mails to the users in 1 minutes after user completed the renting process.

* Security:

The credit card/payment information belong to users are not stored the company’s database and users personal information is not shared through any company or cooperation or whatsoever and cannot be viewed by any end-user or visitors to protect the user’s information.

## Definitions, Acronyms, and Abbreviations

## References

# Current Software Architecture

Describe the architecture of the system being replaced**. If there is no previous system**, this section can be replaced by **a survey of current architectures for similar systems**. The purpose of this section is to make explicit the background information that system architects used, their assumptions, and common issues the new system will address.

# Proposed Software Architecture

Documents the system design model of the new system.

## Overview

Present a bird’s-eye view of the software architecture and briefly describes the assignment of functionality to each subsystem.

## System Decomposition

Describe the decomposition into **subsystems and the responsibilities** of each. **This is the main product of system design.**

## Hardware Software Mapping

Describe how subsystems are assigned to hardware and off-the-shelf components. It also lists the issues introduced by multiple nodes and software reuse.

## Persistent Data Management

Describe the persistent data stored by the system and the data management infrastructure required for it. This section typically includes the description of **data schemes, the selection of a database, and the description of the encapsulation of the database**.

## Access Control and Security

Describe the user model of the system in terms of an access matrix. This section also describes security issues, such as the selection of an authentication mechanism, the use of encryption, and the management of keys.

## Global Software Control

Describe how the global software control is implemented. In particular, this section should describe how requests are initiated and how subsystems synchronize. This section should list and address synchronization and concurrency issues.

## Boundary Conditions

Describe the start-up, shutdown, and error behavior of the system. (If new use cases are discovered for system administration, these should be included in the requirements analysis document, not in this section.)

# Subsystem Services

Describe the **services provided by each subsystem**. Although this section is usually empty or incomplete in the first versions of the SDD, this section serves as a reference for teams for the boundaries between their subsystems. The interface of each subsystem is derived from this section and detailed in the Object Design Document.

# References

The following is an example of listing a book in this section. Check the text to see how it is cross referenced (The whole document is based on [1]).

1. Bruegge B. & Dutoit A.H.. (2010). *Object-Oriented Software Engineering Using UML, Patterns, and Java*, Prentice Hall, 3rd ed.