**GRENT**

System Design Document

13.12.2019

Ahmet Can Terlikçi

Duygu Genç

Tunç Bora Tamsan

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)

SYSTEM DESIGN DOCUMENT[1]

# Introduction

In the process of designing the VehicleRentingSystem, we worked on defining the design goals which includes the concepts ease to use, well defined interface, user-friendliness, end-user, performance and security.

In ease to use goal, we stated that SystemUser must be able to perform renting in short amount of time.

In well-defined interface goal, we stated that the interface components on page and page’s layout design must be responsive and attention-drawing to SystemUser.

In User-Friendliness goal, we stated that the webpage must be visual, support images and include text based info which made/displayed in a way that is aesthetically pleasing to SystemUser.

In end-user goal, the SystemUser must not allowed to display unavailable vehicles.

In performance goal, the system must respond to SystemUser and give feedback in a short amount of time.

In security goal, SystemUser must not be allowed to access other SystemUser’s personal information.

We also selected and designed an architecture pattern to be followed in the implementation process, this architecture pattern is called 3-Tier Architecture. In our implementation 3-Tier architecture is consists of Interface, Application Logic and Storage layers. The interface layer is consists of the web browser that users interact. The Application Logic layer is consists of the main web server of the GRENT which realizes all the user operations. Finally, the Storage layer is consists of the relational database server of the GRENT.

## Purpose of the System

GRENT is a simple web-based car renting system which allows to the users renting different types of cars in specific countries and cities through the authorized offices of the company.

## Design Goals

* 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

**Layout Design:** The design of panels on a specific web page as a whole.

**Auto-scaled pages:** Responsive pages. That means that, they are responsive to different screen sizes and resolutions.

**Legacy System:** An old system which wants to build a new one on top of it without throwing it to the garbage.

## References

The websites which we used as examples in the GRENT:

* www.enterprise.com.tr
* www.europecar.com

# Current Software Architecture

Since GRENT is not a legacy system, we will briefly examine the architectures that using to develop websites commonly today. Although there are infinite number of website designs in the market, the 3-Tier arhitectural style is pretty common today. 3-Tier is a layered architectural style and basicly consists of the layers: Interface, Application Logic and Storage. Layers can be named different but their functions is pretty much same. Interface layer is consists of the web browser which is commonly used today to browse in the web and display websites. Application Logic or Middleware is consists of the system’s itself which is a web server. The web server may be change website to website but its functions is the same: operating the functions of the users. Finally, the storage layer is consists of the storage components of the system. It may be flat files or a relational database. 3-Tier have many advantages such as Scalability, Security and Data Integrity. It is scalable because the application servers can be deployed on many machines. It offers Data Integrity because the middleware provides a validation for the data. Only valid data can be send to the storage tier. It is secure because users can’t reach the database directly, the middle tier controls and validates the users and also there is only one central server which controls all the functions of the system.

In addition, peer-to-peer systems may be used for different purposes with 3-Tier arhictural style on the websites.

# Proposed Software Architecture

## Overview

In the system decomposition of the GRENT, we basically followed the classic 3-Tier architectural style. 3-Tier archiectural style is composed of three different layers which each of them on a specific machine. In the GRENT, we applied the same style. There are three layers which are Interface, Application Logic and Storage. Interface layer consists of the GrentClient component which is the Web Browser that implements the interface of the GRENT. Second layer, the Application Logic is consists of the Web Server that we use in our implementation. It is a composition of GrentServer, UserManagement, VehicleManagement and OfficeManagement components. Storage layer is the subsystem that consists of GrentStorage components which is responsible from the database and storage management of the GRENT.

## System Decomposition

gök, metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

*Fig 1.1: The Subsystem Decomposition of the GRENT (UML Component Diagram).*

Following by the 3-Tier architectural style, GRENT has 3 different Layers as subsystems which named Interface, Application Logic and Storage. Interface subsystem consists of the GrentClient component which is the Web Client that implements interface elements of the GRENT for users. Application Logic subsystem consists of the GrentServer, UserManagement, VehicleManagement and OfficeManagement components. In general, it is the main web server of the GRENT which responsible for all user operations. In detail, GrentServer is the web server component that uses and manages all services in the system. UserManagement component is responsible for all operations related to the User, Renting, Chart and also Discount. In use case context, it realizes the Login, ChangeThePassword, SearchVehicles, ChangeTheSearch, FilterTheVehicles, RegisterTheWebsite, UseDiscount, RentVehicle, ExtendReturningDate, ViewTheRentingInformation, ChangePersonalInfo, ChangeReturningOffice, UpdateTheRenting, Add/Remove/UpdateOfficeUser use cases. In the other hand, VehicleManagement component is responsible for all operations related to the Vehicle. It realizes the use cases AddVehicle, RemoveVehicle and UpdateVehicle. Finally, the OfficeManagement component is responsible for all operations related to the Office. It realizes the use cases AddOffice, RemoveOffice and UpdateOffice. The storage layer is consists of the GrentStorage component which is responsible for the all functions related to the database such that adding, removing, updating and validating data.

## Hardware Software Mapping

ekran görüntüsü içeren bir resim

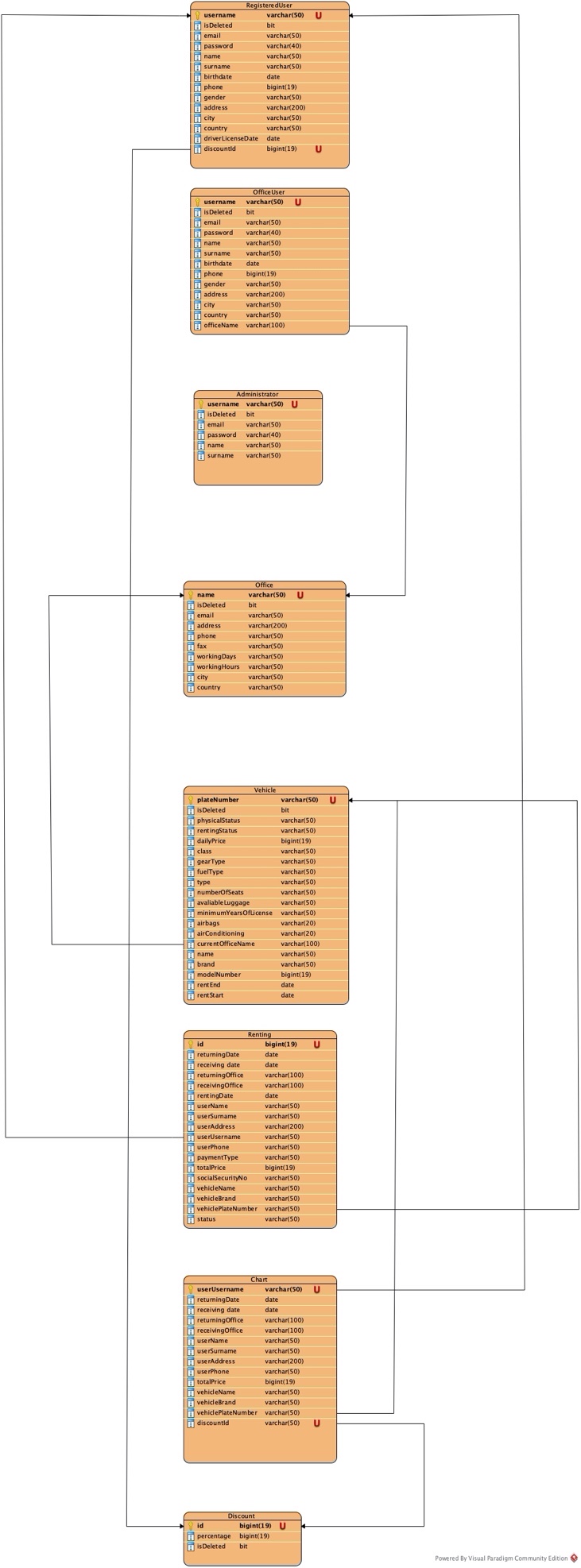
Açıklama otomatik olarak oluşturuldu

*Fig 1.2: Hardware-Software mapping of the GRENT (UML Deployment Diagram).*

By the implementation of 3-Tier architectural style, there are three layers in three different devices(machines) in the GRENT. First device is the UserMachine which consists of the Web Browser that users interact with. Second device is the ServerMachine which consists of the web server of the GRENT and the third device is the DatabaseServer. It consists of the database server of the GRENT.

In the implementation of the project, we used a Java EE framework, JavaServer Faces. Corresponding framework is implemented with a Tomcat web server in the project. Also, for the storage, we used a relational MySQL database server.

## Persistent Data Management

****

*Fig 1.3: Entity Relationship Diagram of the GRENT database server.*

In the implementation of data storage, we used a relational database rather than flat files due to high amount of data and the high number of views which uses the same data source. Also, we chose the MySQL database server because of the familiarity of team members. The detected objects in the class diagram that must be persisted in the system are RegisteredUser, OfficeUser, Administrator, Office, Vehicle, Renting, Discount and Chart objects which can be seen in the Fig 1.3, as a E2R diagram. In the selection of persistent objects, we omitted the inheritence between users. Therefore, we don’t realized the objects like User or SystemUser in the database server. This approach provided us less number of tables in the database which means lower join cost in the database queries also the less implementation work for inheritence between users and role determination.

In the system, we realized the encapsulation of the database with a DatabaseManager interface which can be seen in the Fig 1.1. This type of management provides security and prevents code recurrences.

## Access Control and Security

The user authorization in the GRENT is specified below as a access control matrix.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| OBJECTS  ACTORS | Vehicle | Renting | Chart | Office | RegisteredUser | OfficeUser | Visitor |
| Visitor | search()  filterVehicles()  listVehicles() |  |  |  |  |  | register() |
| RegisteredUser | search()  filterVehicles()  listVehicles() | extendReturning  Date()  changeReturningOffice()  viewRenting()  rentVehicle() | useDiscount()  addRenting() |  | changePersonalInfo() |  |  |
| Administrator | removeVehicle()  addVehicle()  updateVehicle() |  |  | updateOffice()  addOffice()  removeOffice() |  | removeOfficeUser ()  addOffice  User()  updateOffice  User() |  |
| OfficeUser |  | updateRenting() |  |  |  |  |  |

## Global Software Control

In the GRENT we used a Java EE framework to realize the system which is a Event-Driven control flow. Although this is the case, our system also based on threads. Threads are accumulated from the Java EE framework and Apache Tomcat server so it is a automatic process that we don’t implemented in any way.

## Boundary Conditions

In the RAD document, we pretty much completed the exception handling related to software fault and specified their use cases but we did not mention start-up and shutdown behaviours of the system.

Start-up: Administrator logs in to the GRENT Tomcat Server environment, checks the server logs to see if there is a crash or planned shutdown in the server. If there is unusual crash, the Administrator starts the server with the last saved data by the server to avoid data loss. Else, Administrator starts the server as usual from beginning.

Shutdown: Administrator logs in to the GRENT Tomcat Server environment and activates the shutdown function of the system. Since it is a planned shutdown, server shutdown itself directly, without persist any data to the database.

# Subsystem Services

The services provided by each subsystem is specified in the Fig 1.1 as a UML Component Diagram. The provided services may not complete but it is almost stable and the remaining services will be detailed in the ODD document of the project.

For now, total number of 4 components provides services in the system which are UserManagement, VehicleManagement, OfficeManagement and GrentStorage. Now, we will examine provided services by each component.

UserManagement provides the services of Authentication, UserManager, RentManager, ChartManager, DiscountManager and AdminManager. Authentication service realizes the function of login. UserManager realizes the functions for the RegisteredUser which are renting vehicles, updating personal data and using discount, and for the OfficeUser which are updating renting informations of other RegisteredUsers and updating personal information. AdminManager service realizes the functions of the Administrator which are updating/adding/removing Vehicles, Offices and OfficeUsers. RentManager service offers the functions that receiving rentings, creating rentings and updating rentings. ChartManager offers the functions of receiving the chart information and applying discounts to the selected item. DiscountManager realizes the function that validating the discount ids.

VehicleManagement component provides the service of VehicleManager which offers the functions of creating, deleting, updating and receiving Vehicles.

OfficeManagement component provides the service of OfficeManager which realizes the functions that creating, deleting, updating and receiving Offices.

GrentStorage provides the DatabaseManager service which is responsible for the connecting to database, querying to database, and adding, deleting or updating information on the database.

# References

1. Bruegge B. & Dutoit A.H.. (2010). *Object-Oriented Software Engineering Using UML, Patterns, and Java*, Prentice Hall, 3rd ed.
2. https://online.visual-paradigm.com/diagrams/tutorials