System Design Document:
Car Rental Service

Group 4

CSCI 4050: Software Engineering
Instructor: Krys Kochut

Osama Mansour, Stephen Patton, Vincent Lee, and Minh Pham

Table of Contents

1. Introduction

1.1 Purpose of the system

The purpose of the system is to support an online car rental service for young drivers. Customers can sign up, log in, search for cars, reserve cars, return cars, and pay fees. The system is designed to have near continuous uptime and support every conceived need the customer will have.

1.2 Design goals

Ease of use. We want our customers to have no difficulties understanding our website. The design will be implemented in such a way that potential customers can easily use the service from the beginning.

High availability. The more customers we have registered with the system, the more potential rentals the system can handle. It is valuable to have the system ready to handle many concurrent users and to do so with minimal downtime.

Scalability. As the company grows, so will the databases. The implementation design now should be scalable to accommodate a growing user base.

Ease of adding new vehicles and locations. The system is designed in hopes that the business will grow. With that in mind, it must be simple, expedient, and secure to add new locations, vehicles, and vehicle type categories. Full time administrators will be hired to manage these.

1.3 Definitions

Vehicle type: a category of vehicle within the system where all vehicles in that categories share certain features (i.e. 7 seats), and share a common price. Examples: minivan, luxury car

1.4 Reference

Document No.	Document Title	Date	Author
1	Object-Oriented Software Engineering, Using UML, pattern and Java	2010	Bernd Bruegge, Allen H.Dutoit
2	Operating System Concepts with Java	2010	Silberschatz, Galvin, Gagne
3	Database System, An Application-Oriented Approach	2006	Kifer, Bersntein, Lewis

HyperText Markup Language (HTML): the main markup language for creating web pages and other information that can be displayed in a web browser

JavaBeans: reusable software components for Java – classes that encapsulate many objects into a single object (the bean)

JavaScript (JS): an interpreted computer programming language which implementations allow client-side scripts to interact with the user, control the browser, communicate asynchronously, and alter the document content that is displayed

JavaServer Pages (JSP): a technology that helps software developers create dynamically generated web pages based on HTML, XML, or other document type

MySQL: open source, multiuser database management system known for its reliability and efficiency

1.5 Overview

This document has been developed by Group 4 for the Car Rental Service Project. This document was developed from the above resources and is intended to satisfy all the customer requirements, objectives and expectations. This document also covers the current system architecture and our proposed architecture and subsystem service

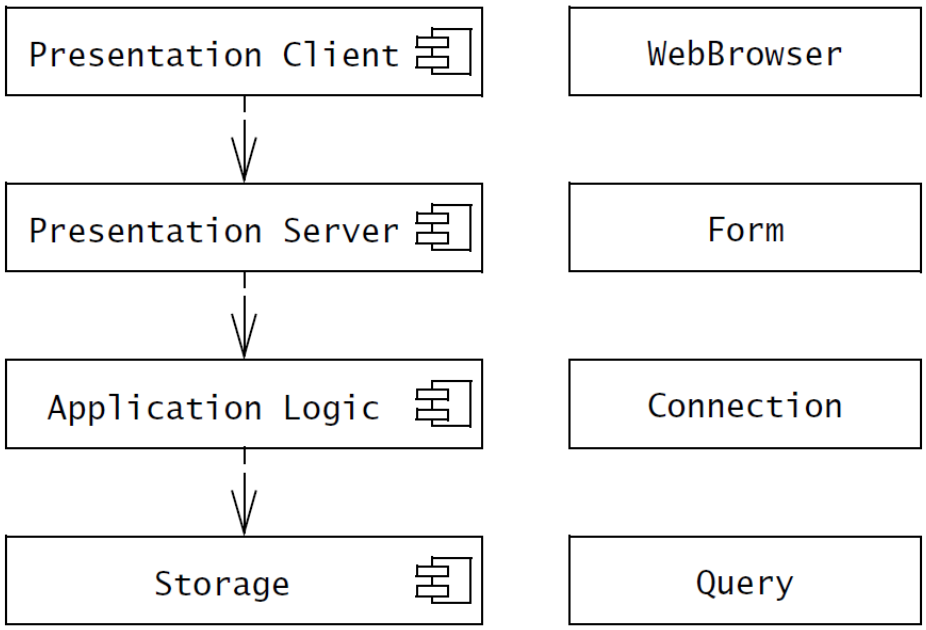
The proposed architecture section details the subsystems of our implementation. It will provide information about how the system will handle subsystem decomposition, hardware/software mapping, persistent data, access control and security, global software control, and boundary conditions.

The subsystem services section proposes a basis for interface creation. It helps clarify boundaries among subsystems.

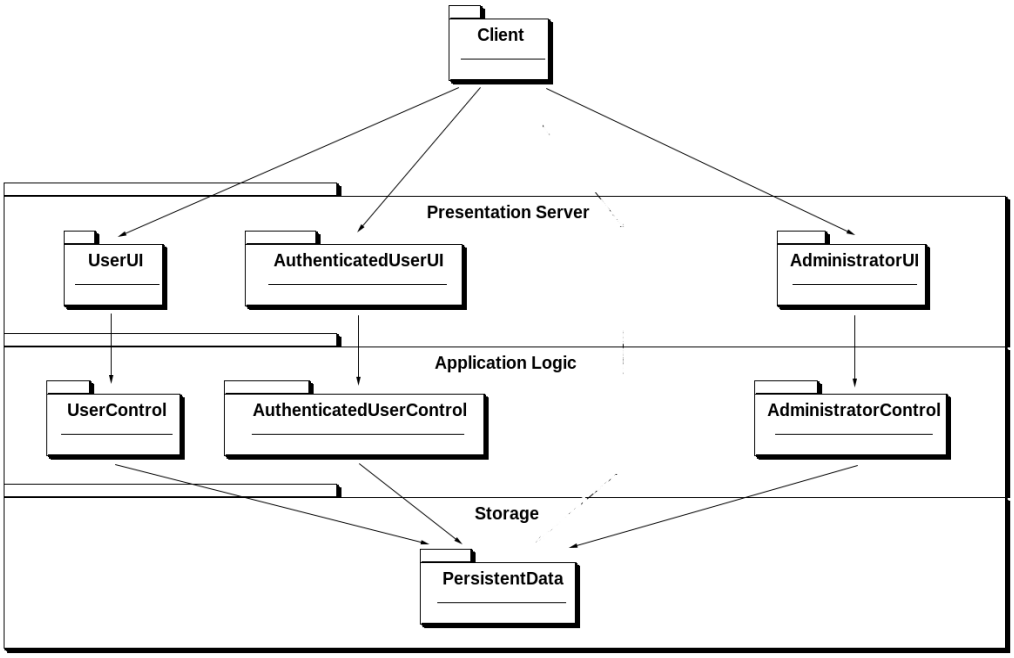
2. Proposed software architecture

2.1 Overview

The car rental service, You Drive, consists of a four-tier architectural style. This style consists of the presentation client, presentation server, application logic, and storage layers. The presentation client layer is located on the user's machines, while the presentation server layer is located on a cluster of servers. The application logic layer includes all control and entity objects, and the storage layer realizes the storage, retrieval, and query of persistent objects.


Presentation Client
WebBrowser
Presentation Server
Form
Application Logic
Connection
Storage
Query

2.2 Subsystem decomposition



2.2.1 Presentation Layer – UserUI Subsystem

Operation	Returns	Arguments(s)	Description
logIn		username password	Verifies the user identity and log the user in or display message.
register		username password socialSecurityNumber firstName lastName address telephone creditCard	Registers a AuthenticatedUser with the given credentials and information or displays an error message.
locationSearch	List<Location>	keyword	Searches the available locations in the

			system.
vehicleSearch	List<Vehicle>	keyword	Searches the vehicles in the system.

2.2.2 Presentation Layer – AuthenticatedUserUI Subsystem

Operation	Returns	Arguments(s)	Description
logout			Log the user out.
ViewMyInfo	Customer		Displays all personal information of the current logged in Customer.
ViewAvailbileCars	List<Vehicle>	carName carLocation	Displays the Cars available for rental by location.
ViewMyRentalHistory	List<RentalHistory>		Displays a current logged Customer rental history.
ViewMyBillingInfo	Customer		Displays Current logged Customer billing report.
ViewMyCarRental	CarRental		Displays Customer past and present car rentals.
EditMyInfo	Customer	firstName lastName address telephoneNumber	Allows Customers to edit their information as needed.

2.2.3 Presentation Layer – AdministratorUI Subsystem

Operation	Returns	Arguments(s)	Description
logout			Log the user out.
ViewMyInfo	Administrator		Displays all personal information of the current logged in admin.
ViewVehicles	List<Vehicle>	carName carLocation	Displays the Cars available within the Car rental system.
ViewRentalHistory	List<RentalHistory>	customerName	Displays a rental history for all customer by name.
ViewBillingHistory	List<BillingHistory>	customerName	Displays all Customer History Billing reports.
ViewCarRentals	List<CarRental>	customerName	Displays All Customer Car rental reports.
EditVehicleInfo	Vehicle	make model tag mileage last_serviced condition	Edit Vehicle Information.
EditCustomerInfo	Customer	socialSecurityNumber firstName lastName address telephone	Allows Admin to edit customer information both adding and removing Customers.

2.2.4 Application Logic Layer – UserControl Subsystem

Operation	Returns	Arguments(s)	Description
logIn	boolean	username password	Verifies the user identity and log the user in or display message.
register	boolean	username password socialSecurityNumber firstName lastName address telephone creditCard driverLicense	Registers an AuthenticatedUser with the given credentials and information or displays an error message.
locationSearch	List<Location>	keyword	Searches the available locations in the system.
vehicleSearch	List<Vehicle>	keyword	Searches the vehicles in the system.

2.2.5 Application Logic Layer – AuthenticatedUserControl Subsystem

Operation	Returns	Arguments(s)	Description
logOut			Log the user out.
GetMyInfo	Customer		Returns customer personal information of the current logged in Customer.
GetAvailbileCars	List<Vehicle>	carName carLocation	Returns the Cars available for rental by location.
GetMyRentalHistory	List<RentalHistory>		Returns a current logged Customer rental

			history.
GetMyBillingInfo	Customer		Returns Current logged Customer billing report.
GetMyCarRental	CarRental		Returns Customer past and present car rentals.
GetMyInfo	Customer	firstName lastName address telephoneNumber	Returns Customer information.

2.2.6 Application Logic Layer – AdministratorControl Subsystem

Operation	Returns	Arguments(s)	Description
logOut			Log the user out.
GetMyInfo	Administrator		Returns all personal information of the current logged in admin.
GetVehicles	List<Vehicle>	carName carLocation	Returns the Cars available within the Car rental system.
GetRentalHistory	List<RentalHistory>	customerName	Returns a rental history for all customer by name.
GetBillingHistory	List<BillingHistory>	customerName	Returns all Customer History Billing reports.
GetCarRentals	List<CarRental>	customerName	Returns All Customer Car rental reports.
GetVehicleInfo	Vehicle	make model tag	Return Vehicle Information.

		mileage	
		last_serviced	
		condition	
GetCustomerInfo	Customer	socialSecurityNumber	Returns customer information for editing.
		firstName	
		lastName	
		address	
		telephone	

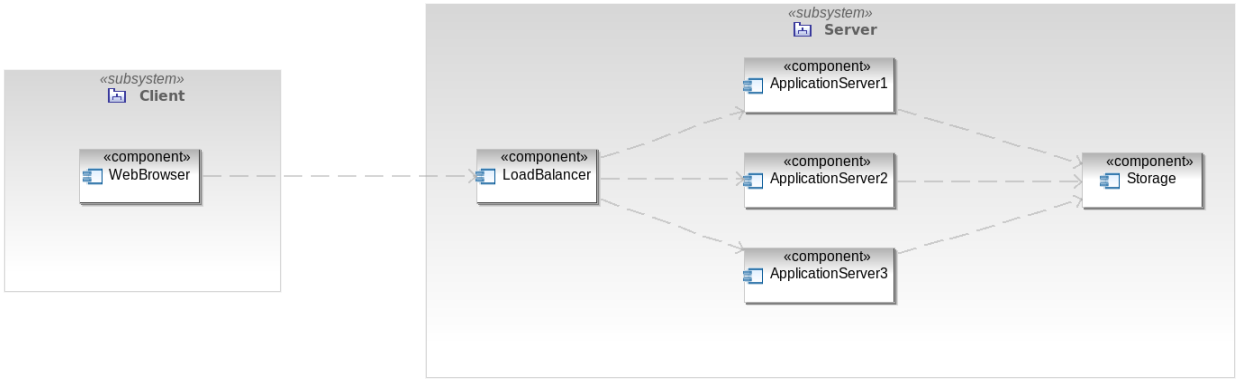
2.3 Hardware/software mapping

The four-tier architectural style is realized in hardware as two subsystems: Client and the Server. The Client subsystem contains the Presentation Client while the Server subsystem contains the Presentation Server, Application Logic, and Storage. An additional component, the Load Balancer distributes network traffic across a number of servers in the subsystem. In overview, the Client connects with a Load Balancer which routes the user's request to one of the three Application Server's which then communicates with the centralized storage to finish processing the request.

The presentation client is a web browser which is a software application for retrieving, presenting and traversing information resources from the server subsystem. The web browser also is capable of executing JavaScript (JS) which is a client scripting language that communicate asynchronously, and can alter the document content that is displayed in the web browser.

The presentation server and application logic layers reside on one of the three Application Server's. The interface is realized with HTML styled with CSS. The HTML is wrapped in JavaServer Pages (JSP) which allows dynamically generated web page creation. The JSP pages use the Java programming language, which is deployed on a servlet container Tomcat. Inside the Tomcat server, are JavaBean's which contain the application logic or connection required to communicate with the storage system.

The storage system consists of a MySQL Server relational database management system (RDBMS). This persistent storage is accessed by one of the three application servers providing abstraction from the system ensuring security, and data integrity.



2.4 Persistent Data Management

We will be using the MySQL platform to develop our databases. MySQL contains a relational relationship between objects and ID associated with them which allows us to access information for modification, querying, and deletion. Below we have list the persistent objects and explained important attributes that are accessed and their interactions within the car rental system.

2.4.1 Persistent Objects

- **CustomerUser**: contains the attributes identifying a customer within the Car Rental System
    - Is Referenced by **BillingInfo**, which contains relating information needed to view user billing
    - Is Reference by **RentalHistory**, which contains relating information needed to display a historical account of user car rentals
- **Administrator:** contains information relating the an administrator within the Car Rental System
    - The Administrator can add/delete/modify all and any types of information within the database system within including all persistent objects

2.4.2 Storage Strategy

We will be using the MySQL relations database for our storage needs with the Car rental System.

- MySQL is a free, accessible relational database system that has little barriers in configuration
- MySQL is lightweight, and quick for querying and modifying databases with in an optimized manner
- MySQL is also widely used database with basic standards established for many types of Qualified Database Administrators, Analyst, and Designers.

2.5 Access and Control Security

	User	Cars	Rental	Billing Info	Rental History
User (Non-Authenticated)	none	search	none	none	none
CustomerUser	search rent	search	rent	View	View
AdministrativeUser	Create Update Delete Search	Create Update Delete Search	View Update	View Update	View Update

	Car Type	Customer Rentals	Admin Info	User Defined Attribute Names	User Defined Attribute Values
User(Non-Authenticated)	none	none	none	none	none
Customer	none	none	none	saved	saved
Administrator	Create Delete	View Update	View Create Update Delete	saved	saved

2.6 Global Software Control

The system will be implemented with Servlets and JSP approach, the *procedural driven control* is ideal for maintaining a simple path oriented way of controlling data flow within the server side of the system. Rental Transactions and Threaded processes in implementation will also follow procedural driven logic processes.

An Example of Client/Server Interaction:

The available server will wait for a request via HTTP to the servlet based inside the web server, and on execution of the procedural request a thread will be created with the following constraints

- If any number of rental transactions are being performed then availability statuses need to be updated in order to reduce concurrent rental of non-existent vehicles.
- If transactions are successful, the user should receive a message pertaining to their request be it a rental, billing information, car information request, object manipulation, or other system components.
- If transaction fails, system should revert back to old state to avoid irreversible data. The user shall also receive an appropriate message pertaining to failure.

All Transactions request should uphold this follow of client/server interactions within the system. All user who have made a request and succeed to a rental transaction page should be given their request with 100% accuracy. The system will perform an inventory check before allowing the user into the rental transaction page if not a message will occur notifying the user. System is to minimize any deadlock or race conditions occurring during concurrent transactions.

2.7 Boundary Conditions

2.7.1 Start-up conditions

The System administrator must adhere to the following to execute a correct system startup.

1. The System will begin by starting MySQL Database
2. The System will start the Apache Tomcat web server with Servlet Containers

2.7.2 Shut-down conditions

To prevent partial writes to the database, this shutdown procedure must be followed.

1. The System will shut down by closing the Apache Tomcat Webserver
2. The System will shut down by closing the MySQL Database

2.7.3 System-Failure conditions

Exceptions might occur during the operation of the system, messages are used to announce their presence.

1. *Client-Side Exception:* If invalid data is input to the system with the presentation layer of the system. An Error message is displayed to the client with no system changes edits.
2. *Server-Side Exceptions:* All errors occurring on the server side shall first proceed with a message to the current user with diagnosis options. Dependent upon the users preference the Server-side will execute the following desired process.(Ex: Database Rollbacks or Information restorations)

3. Glossary

Apache Tomcat (Tomcat): an open source web server and servlet container developed by the Apache Software Foundation (ASF). It implements the Java Servlet and the JavaServer Pages specifications from Sun Microsystems, and provides a pure Java HTTP web server environment for Java code to run in.

Application layer: a layer in architectural style that includes all control and entity objects, realizing the processing, rule checking, and notification required by the application.

Attribute: a name paired with a domain (commonly referred to as type or data type).

Cascading Style Sheets (CSS): a style sheet language used for describing the presentation semantics of a document written in a markup language. The most common application is to style web pages written in HTML and XHTML.

Client/Server Architecture: an architectural style in which a subsystem, the server, provides service to instance of the other subsystems called the clients, which are responsible for interacting with the user.

Concurrency: a property of system in which several transactions are executing simultaneously, possibly interacting with each other.

Data integrity: maintaining and assuring the accuracy and consistency of data over its entire life cycle, and is a critical aspect to the design, implementation and usage of any system which stores, processes or retrieves data.

Deadlock: a situation in which two or more competing actions are each waiting for the other to finish, and thus neither ever does. Deadlock can happen in transactional database and operating system.

Four-tier architecture: an architectural style that organizes subsystem into four layers: the interface layers which decomposed into a Presentation Client layer and Presentation Server, the application logic layer, and the storage layer.

Global software control: describes how subsystems synchronize, including listing and addressing synchronization and concurrency issue.

Hardware Software Mapping: a process to realize the subsystems by mapping the objects and subsystem onto the chosen hardware and/or software. This process includes two types: mapping objects and mapping associations.

Interface layer: a layer in architectural style that includes all boundary objects that deal with the user, including windows, forms, web pages, and so on.

Load Balancer: a computer networking software or hardware for distributing workloads across multiple computing resources, such as computers, a computer clusters, network cluster, network links, central processing units or disk drives.

Persistent data: data which has been specifically set by the user. It is never destroyed even when a parameter is hooked up to a source object.

Persistent data management: control the access to persistent data through the menu.

Persistent object: a unique identifier of a record on a table, used as the primary key.

Procedural driven control: ideal for maintaining a simple path oriented of controlling data flow within the server side of the system.

Race condition: where several transactions access and manipulate the same data concurrently and the outcome of the execution on the particular order in which the access takes place, is called a race condition.

Relation: a set of tuples (d_1, d_2, …, d_n) where each element d_i is a member of D_i, a data domain.

Relational database: a finite set of relation, including a set of relation schemas-called database schema, and a set of corresponding relation instances-called database instance.

Relational database management system (RDBMS): a database management system (DBMS) that is based on the relational model.

SQL: Structured Query Language designed for managing data held in a relational database management system.

Storage layer: a layer in architectural style that realizes the storage, retrieval, and query of persistent objects.

Subsystem decomposition: identification of subsystems, service, and their association to each other (hierarchical, peer-to-peer, etc.).

Transaction: the change of the state of the enterprise corresponding to the change of the information stored in the database in real time by program.

**subsystem services not included on this iteration of the SDD*