Bilkent University

Department of Computer Engineering

Senior Design Project

Project name: DAOS (Dealership Assistance and Optimization System)

Low Level Design Report

Metehan Kesekler, Kerem Tuna Özlü, Mustafa Said Sarı, Sarp Tuğberk Topallar

Supervisor: Uğur Doğrusöz

Jury Members: M.Mustafa Özdal and Mehmet Koyutürk

Progress Report

February 12, 2018 This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

1. Introduction

When dealerships are given by companies to third parties, two of the most essential concerns are legality and profitability for such a dealership. Both of these issues can be addressed by selecting the appropriate location for the dealership. Finding such proper locations manually is very troublesome, because one must obtain information about the legal restrictions and regulations about the area. These restrictions and regulations may differ from province to province, even sometimes municipalities within the same province enforce different regulations. In addition to that, profitability potential is very hard to measure and may be affected by many factors such as weather, time of the day, seasonal activities of the locals, shopping demographic of the area etc. Even if these potential factors are successfully be identified initially, they may also change later on. Luckily, all these problems are dependent of the location of the dealership and using a geographic information system (GIS) can be helpful to solve all of them.

The issue of legality is very area sensitive and different constraints must be met for different types of businesses. For example, liquor and tobacco shops cannot be nearer than 100 meters to schools, dorms or sanctuaries by law. Therefore, having such sensitive places as points of interest (POI) in DAOS and display them on the actual map would certainly be very beneficial for companies trying to pick up the right venue for the regulatory assessment.

When the profit potential of a dealership is considered, the area can be manually inspected. But manual inspection can be made for a limited amount of time and therefore may not be enough to cover the overall trend for the area. Sending a team to different cities around the country for field surveys is also very costly. DAOS will present the data about spending potential of the area such as number of residents in the area and the foot traffic of the streets. Data provided by DAOS can be interpreted to become aware of the trends about the area even it could be used for making suggestions about possible business opportunities since the information about the area is accumulated in the database.

After the dealership becomes operational, its performance must be continuously observed. Because, the habits of the customers are not static and businesses are very sensitive to competition. So, a very profitable location for a dealership may not be still profitable so monitoring the operations of that dealership can be very important for making a decision about allocating more resources there or closing it entirely. Monitoring the sales of different branches only provides information about quantity, but considering branch sales

according to their local shopper profile will give their actual performance metric for evaluation.

Our solution, DAOS with its enhanced user interface, interactive and color coded maps and broad capabilities about different types of businesses, provides the answers for the following questions “Can this dealership be opened here?”, “Should this dealership be opened here?” and “Is this dealership still profitable?”

1.1 Object Design Trade-Offs

1.1.1 Usability vs Functionality

User interface and ease of use is important in order to prevent confusion of the users because DAOS is willing to reach any kind of user from dealership owner to highly educated executives of investor company. The more functions are added to DAOS, the harder for users to take advantage of it. There will be more buttons, more settings and more choices appearing on the screen with every single functionality.

1.1.2 Reliability vs Time

DAOS will make critical suggestions to investors, which means that assistance of DAOS should be a reliable. Reliable data must be true and relevant. Composition of reliable data is problematic and time consuming.

1.1.3 Availability vs Maintainability

DAOS should be available any time when users want to use it. DAOS should be precise and available any time investors need suggestions or want to follow their existing investments. As a trade-off maintenance and repair of the system become harder because shutting off the servers is not an option while maintenance.

1.1.4 Security vs Cost

DAOS collects data from users including dealership locations, sales data and company info. Thus, it is crucial for our system to ensure security and keep the information of the company secure. For security, DAOS relies on encryption. Relatedly, the security introduces monetary, time, and labor cost.

1.2 Interface Documentation Guidelines

The detailed interface documentation outline is provided below:

Class: Description of the class
Attributes: attribute type and attribute name
Methods: method name, parameters and return value

1.3 Engineering Standards

In the reports, UML [1] design principles are used in the description of class interfaces, diagrams, scenarios and use cases, subsystem compositions, and hardware-software components depiction. UML is a commonly used standard that allows simpler description of the components of a software project. With standard UML models we were able to represent the system structure, software components, and functionalities. The reports follow IEEE citation guidelines [2] for he references. IEEE is also a commonly accepted engineering standard.

1.4 Definitions, Acronyms, and Abbreviations

DAOS: Dealership Assistance and Optimization System
GIS: Geographical Information Systems
POI: Point of Interest
UI: User Interface
API: Application Programming Interface
Client: The part of the system the users interact with
Server: The part of the system responsible from logical operations, scheduling, and data management

2. Packages

Our system is composed of 4 packages: View, Controller, Application, and Data packages. View and Controller packages are in Client subsystem in the Presentation package while the Application and Data packages are in Server subsystem.

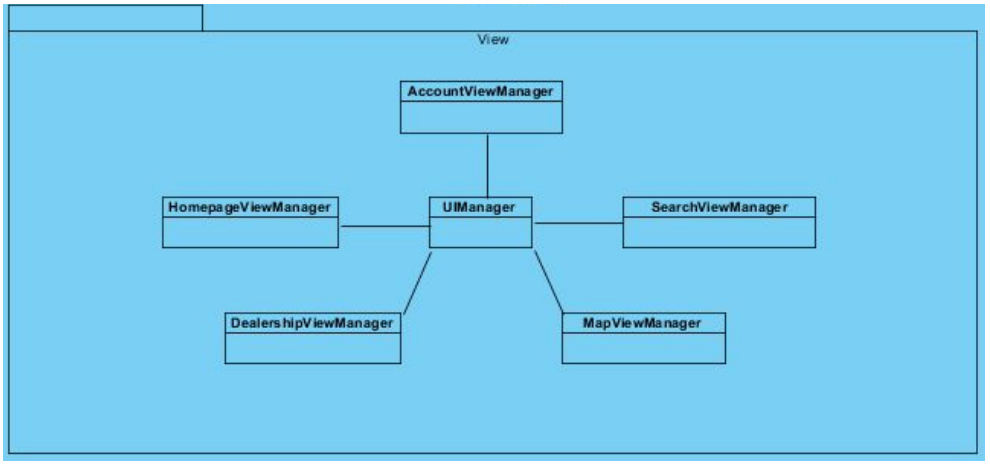
2.1 Client

The client corresponds to the desktop application of our system. The client is the presentation layer of our system. The user creates an account or logins to the system via the client. The client requests login access from the server. The client manages the account of the user, the preferences and searching. The user specifies the filters to see the current or past performance of the dealerships.
Client subsystem includes Presentation Tier and the Presentation Tier has View and Controller subsystems. View subsystem is responsible for all the user interface operations. Controller subsystem manages the interaction between the client and the server and controls the operations within the client. It collects the data from the application and sends it to server. It also requests the required data such as performance report or suggestion requests.

2.1.1 Presentation

Presentation package contains two subpackages which are view and controller.

2.1.1.1 View



UIManager: Main User Interface class that manages other UI classes.

AccountViewManager: Class for arranging the account view.

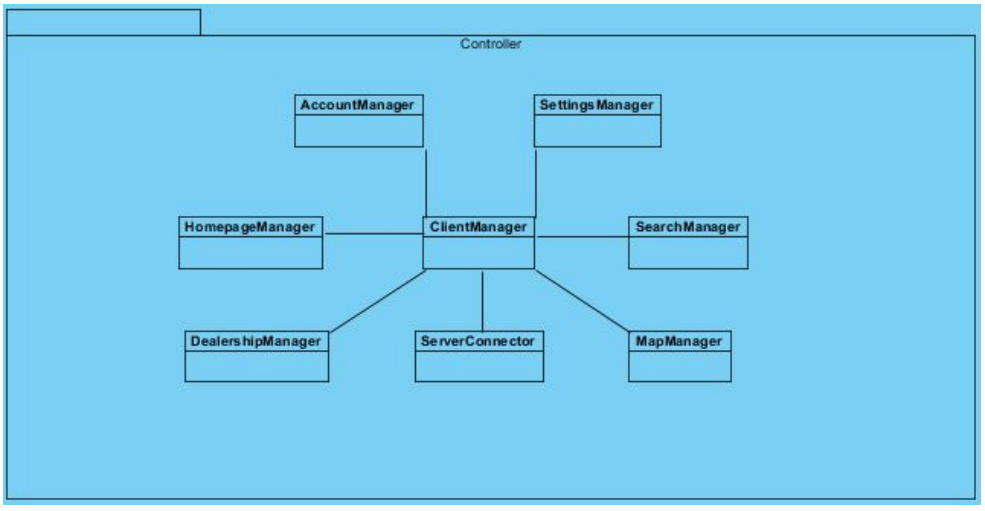
SearchViewManager: Class that manages dealership and region searching and also showing the filters.

MapViewManager: Responsible for map views.

DealershipViewManager: Class that is responsible for showing filtered and/or sorted dealerships.

HomepageViewManager: Class that presents all the components of the users' homepages.

2.1.1.1 Controller



ClientManager: Main manager class that manages all operations of other controller classes.

AccountManager: Responsible class for account information.

SettingsManager: Class that will change the settings by the users' preferences.

SearchManager: Class that is responsible for filtering, searching and sorting.

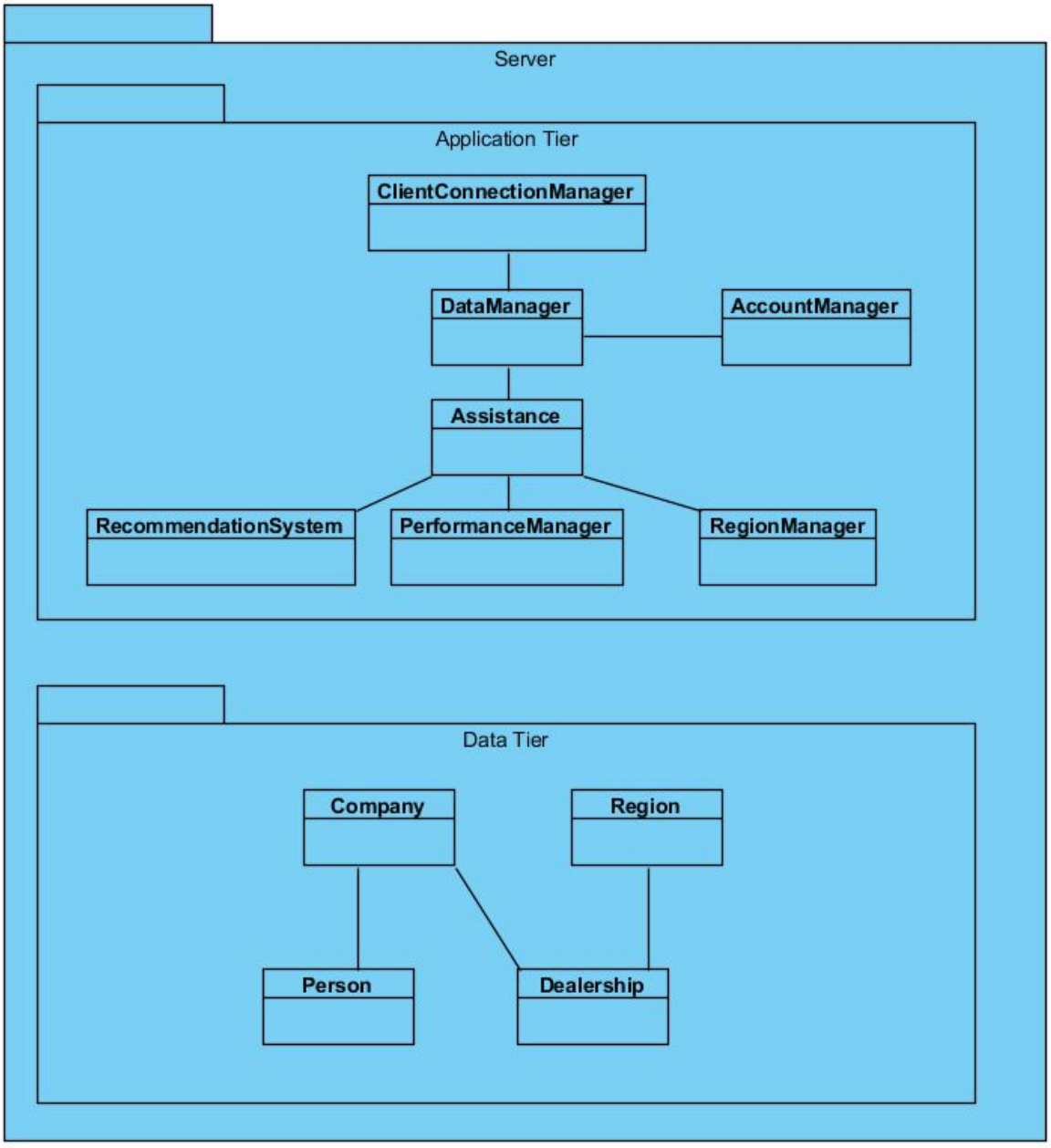
MapManager: Class that is changing the map according to performance and new dealerships.

ServerConnector: Class for handling interaction between the client and the server.

DealershipManager: Class that is responsible of details of dealerships.

HomepageManager: Class that decides the information on the homepage

2.2.Server

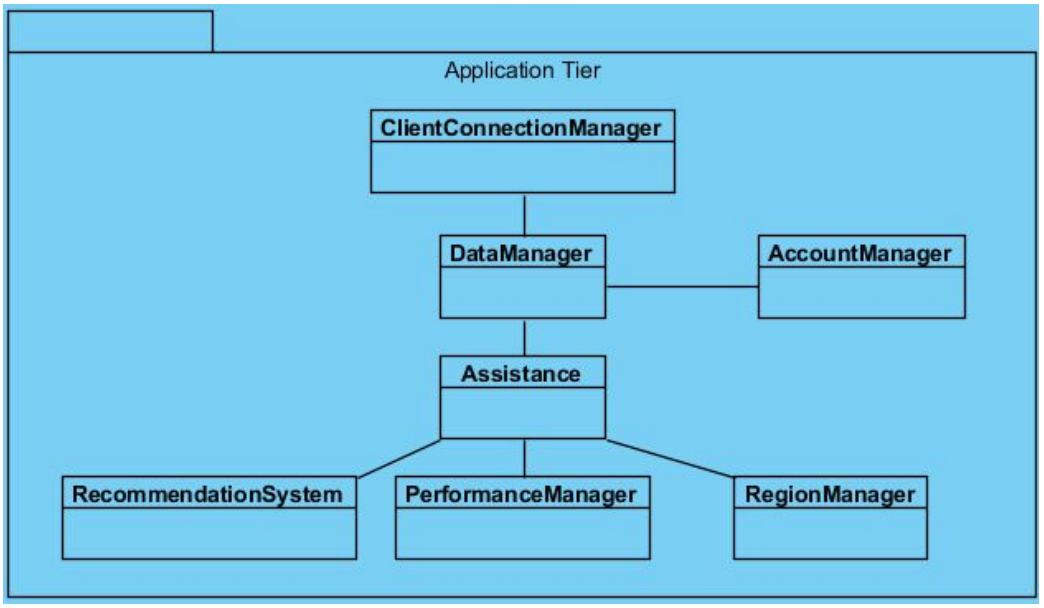


Server is the part of our system where the crucial operations such as showing performance and dealership location suggestions are handled. The client sends data of the user to server. Server contains all data of all users. Furthermore, server is responsible from collecting dealership and region details from Google Maps. Server continuously analyzes the performance data to improve the recommendation system. Server has crucial functions such as showing performance details and suggesting a location for a new dealership. When the

user requests a new suggestion, the client sends the filters for the suggestion to server. The server creates the best possible suggestion and sends it back to the client.

Server includes two layers, Application Tier and Data Tier. Application Tier is the main operational layer that handles all performance and suggestion functions. Application Tier also interacts with the client to respond to the requests of the clients. Data Tier includes the Database Management Subsystem. This subsystem represents the database where the region, dealership and account information are stored.

2.2.1.Application



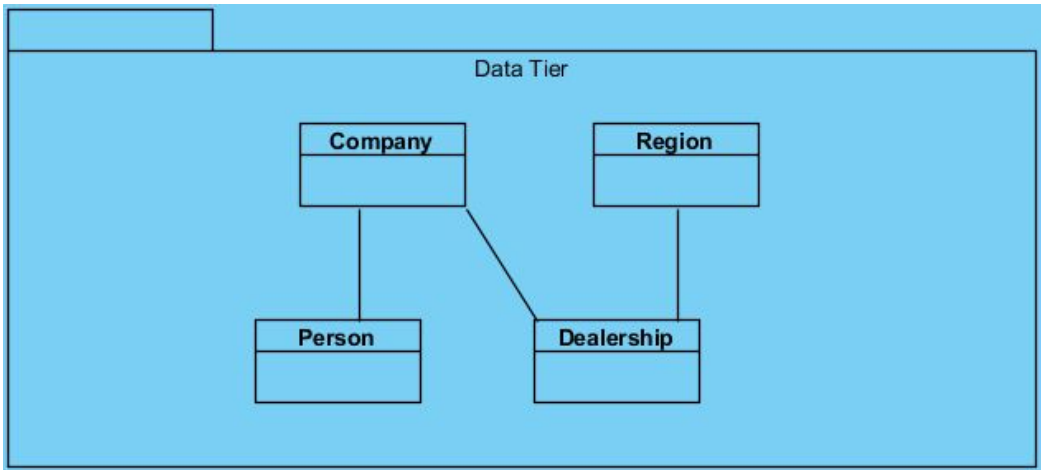
ClientConnectionManager: Every client is using a determined area in our server. ClientConnectionManager's job is to separate all the users and keep their information safe from others.

DataManager: Manager class responsible from obtaining necessary data from the Data Tier. All the data about region, company and dealerships is kept in the database. DataManager collects the data required for the current operation. It also manages data update operations.

AccountManager: Class that manages the accounts of companies. All data of the user accounts are handled by this class. AccountManager interacts with the client and the Data Tier to manage account operations.

Assistance: Class that is responsible for assisting the user. Assistance class analyzes the performance of dealerships and remarks positive or negative events.
RecommendationSystem: Responsible class for DAOS main feature, recommendation. RecommendationSystem will calculate the best possible location for a new dealership.
PerformanceManager: PerformanceManager is responsible for calculating the performance of the dealerships.
RegionManager: Class that is managing the regions. Regions are very crucial part of the system because of the dealerships' nature of getting affected by the environmental changes.

2.2.1.Data



Company: Company is the main customer of DAOS. Companies' information will be held in Company.
Person: Contact person of a company will be held in Person.
Region: Crucial data of a region such as POIs will be held in Region.
Dealership: Dealerships' information will be held in Dealership.

3. Class Interfaces

3.1.Client

3.1.1 Views

class UIManager
Attributes AccountViewManager
Methods updateCurrentPage()

class AccountViewManager
Attributes
Methods showAccountInfo()

class DealershipViewManager
Attributes
Methods showList() updateList() showMoreInfo()

class HomepageViewManager
Attributes
Methods showMap() updateMap() showDealership() updateDealership()

class SearchViewManager
Attributes
Methods updateFilter()

class MapViewManager
Attributes
Methods updateMap()

3.1.2 Controllers

class DealershipManager
Attributes dealershipList
Methods addDealership(Dealership d) removeDealership(Dealership d) listDealership(filterList fl) addFilter(String s) showMoreInfo(Dealership d) recommendDealership(filterList fl) Getters and setters

class AccountManager
Attributes name mail address
Methods changePassword(int oldPass, int newPass) Getters and setters

class ClientManager
Attributes currentPage
Methods changeCurrentPage() Getters and setters

class SettingsManager
Attributes
Methods changeLanguage()

class HomepageManager
Attributes map dealershipList
Methods Getters and setters

class SearchManager
Attributes filterList
Methods search(filterList fl) Getters and setters

class ServerConnector
Attributes
Methods

class MapManager
Attributes map
Methods Getters and setters

3.2.Server

3.2.1 Application

class ClientConnectionManager
Attributes private DataManager dataManager private Assistance assistance private List<Request> request private List <Response> response
Methods private RequestData processRequest(Request request)

class DataManager
Attributes private List<RequestData> requestData private List<AssistanceData> assistanceData
Methods private AssistanceData verifyId()

class AccountManager
Methods public Credential getCredential(Request request) public Boolean verifyCredential(Credential credential)

class Assistance
Methods private String getRecommendation(AssistanceData assistanceData) private String getPerformance(AssistanceData assistanceData)

class RecommendationSystem
Methods public Region recommendRegion(DealershipType dealershipType)

class PerformanceManager
Methods

public int compareDealerships(Dealership dealership) public int analyzeSaleSlopes(Dealership dealership)

class RegionManager
Methods public RegionMetric checkRegionMetric(Region region)

3.2.2 Data

class Company
Attributes private int company_ID private String companyName private String type private List <Dealership> dealership
Methods Getters and setters

class Person
Attributes private String name private String phoneNumber private String Address private String Position

Methods

getters and setters

class Region

Attributes

private String province

private String district

private int postalCode

private int footTraffic

private int avgRent

private int population

private int addressID

private String addressDescription

Geocode geocode(int x, int y)

Methods

getters and setters

class Dealership

Attributes

private Region region

private Person owner

private int dealership_ID

private String type

private int salesAmount

private int salesPercentage

private double costToCompany

Methods
getters and setters
public int calculateSalesAmount()
public int calculateSalesPercentage()
public int calculateCostToCompany()

4. Glossary

DAOS: Dealership Assistance and Optimization System

GIS: Geographical Information Systems

POI: Point of Interest

5. References