SYSTEM DESIGN DOCUMENT[1]

# Introduction

Previously in RAD document, project’s application domain and system analysis model were presented.

SDD document reports the transformation of the analysis model to system design model. SDD documents contains solution domain which is proposed and specified, design goals, subsystem decomposition, strategies and the definitions of subsystems and interfaces. Mainly, SDD portrays a virtual system that includes all of the specifications and requirements in RAD, and will create a service in boundaries between subsystems and interfaces.

## Purpose of the System

The purpose of the system is providing an efficient platform for users, which are customers, admin, managers, guest. For customers and guest, the main functionality that aimed is to search convenient trips to the places that want to travel and buy their tickets easily. For admin and managers, the main functionality that aimed is to editing the backup information or settings quickly, correctly and with ease. Further details can be found in RAD Document.

## Design Goals

TrustBus android app is a mobile application which users can buy online ticket using their credit card. Customers and guests can use the system for searching the appropriate trips and buy the tickets. Admin and managers can use the system for editing some backup information such as changing a trip’s departure hour. Therefore, this system should provide certain constraints, functional and non-functional requirements. For example, the customers which use TrustBus application should be able to see their tickets which they bought, the managers should be able to delete a trip.

Some of the important requirements and constraints are:

• Multiple Users

The system should support tasks that are performed by multiple users at a time, supplying

each with the necessary information at the appropriate time.

• Usability

Since the end-user will be using the system while performing work, it is essential for the system to be intuitive and easy to use. For example the user should be able to understand the procces.

• Scalability

The system must be scalable in terms that it can support many users communicating

orretrieving information at the same time.  
• Reusability of Code

To minimize implementation time and improve efficiency, each part of the system has

been designed as a component.  
 • Understandability

User should be able to understand the components of the system and can use them without any instructions.

• Reliability

System should be reliable which means it should take the stress of the systems and not make any failure.(A measure of success with which the observed behavior of a system confirms to the specification of its behavior)

• Location-Transparency

Server might itself be distributed, but provides a single "logical" service to the user

• High Performance

Client optimized for interactive display-intensive tasks; Server optimized for mobile CPU-intensive operations

• Flexibility

User interface of client supports a variety of end devices (From Android KitKat to latest version).

**Functional Requirements:**

-Managers can add new trips or edit current trips.

-Users can delete their account with approvel from admin.

**-**Admins can edit user credentials.

-Users can list their latest tickets.

-The system should give gift ticket when 10 ticket bought from one customer.

-Guests can list trips without any login and can buy ticket with few information.

**Non-Functional Requirements:**

-The system can never down, also at payment can never stop or cause error.

- The system must be running %100 of the time when buying and also when listing trips. The system can never crash.

- The system must allow more than 1000 parallel user. 1000 user must be able to use the system at same time. System must return response immediately.

- The system must be changeable and easily to maintenance.

**Constraints:**

-The implementation language is Java.

-The project is mobile-based.(Android)

## Definitions, Acronyms, and Abbreviations

SDD - System Design Document  
 RAD - Requirement Analysis Document

## References

RAD is taken as a reference.

Kamil Koç and Metro Ulaşım mobile applications are taken as references on this project.

# Current Software Architecture

Current software architecture styles for our projects , similaryly most of the android apps serving for a purpose like ticket buy based, not using client server architecture style.

# Proposed Software Architecture

In this project, we use Model-View-Controller Architectural System. Model subsystems maintain domain knowledge and does not depend on any view or controller subsystem (Entity Objects). View subsystems shows the system to the user (Boundary Objects), and Controller subsystems manage the sequence of interactions with the user (Control Objects). We used MVC model because, in our system, entity objects and data will be in Model, and Controller can be called bridge.

Since the project is mobile and java based, MVC is a great fit.

## Overview

The design has 14 subsystems. Customer interface, Guest interface, Admin interface and Manager interface are the subsystems for “Presentation Layer”. Login subsystem, Buy Ticket subsystem, Registration subsystem, Edit Bus Schedule subsystem, Manage Booking subsystem, Trip Info subsystem, Payment subsystem, Manage Account subsystem and Update User Info subsystem are the subsystems for “Business Layer.” Lastly, Data Access subsystem is the subsystem for “Data Layer”.

**Admin Interface :** Provides fuctionalities that are only usable for admins.Under the interface there are subsystems for admin.

**Guest Interface :** Provides subsytems that are only usable for guest user type.

**Manager Interface :** Provides subsytems that are only compatible with manager user type.

**Delete User Account Subsystem** : Provides services for user to delete their account.Needs confirmation from admin.

**Data Access Subsystem :** Provides services for getting and pushing data from Database.

(Diğer subsystem ve interface tanımları gerekli. 3.2 deki tanımları alabilirsiniz)

**Buy Ticket Subsystem:** Provides services for customers to buy ticket on system.

**Edit Bus Schedule Subsystem:** Provides services for admin and manager to update, add or delete trips on system.

**Manage Booking Subsystem:** Provides services for admin and manager to edit and delete bookings on system.

**Login:** Provides for any user enter the application with their own account.

Login needs e-mail and password.

**Payment:** Provides system for payment successful or not so system

can create ticket for customer or guest.

**Update user info:** Provides services for admin to change user profile

from admin account.

## System Decomposition

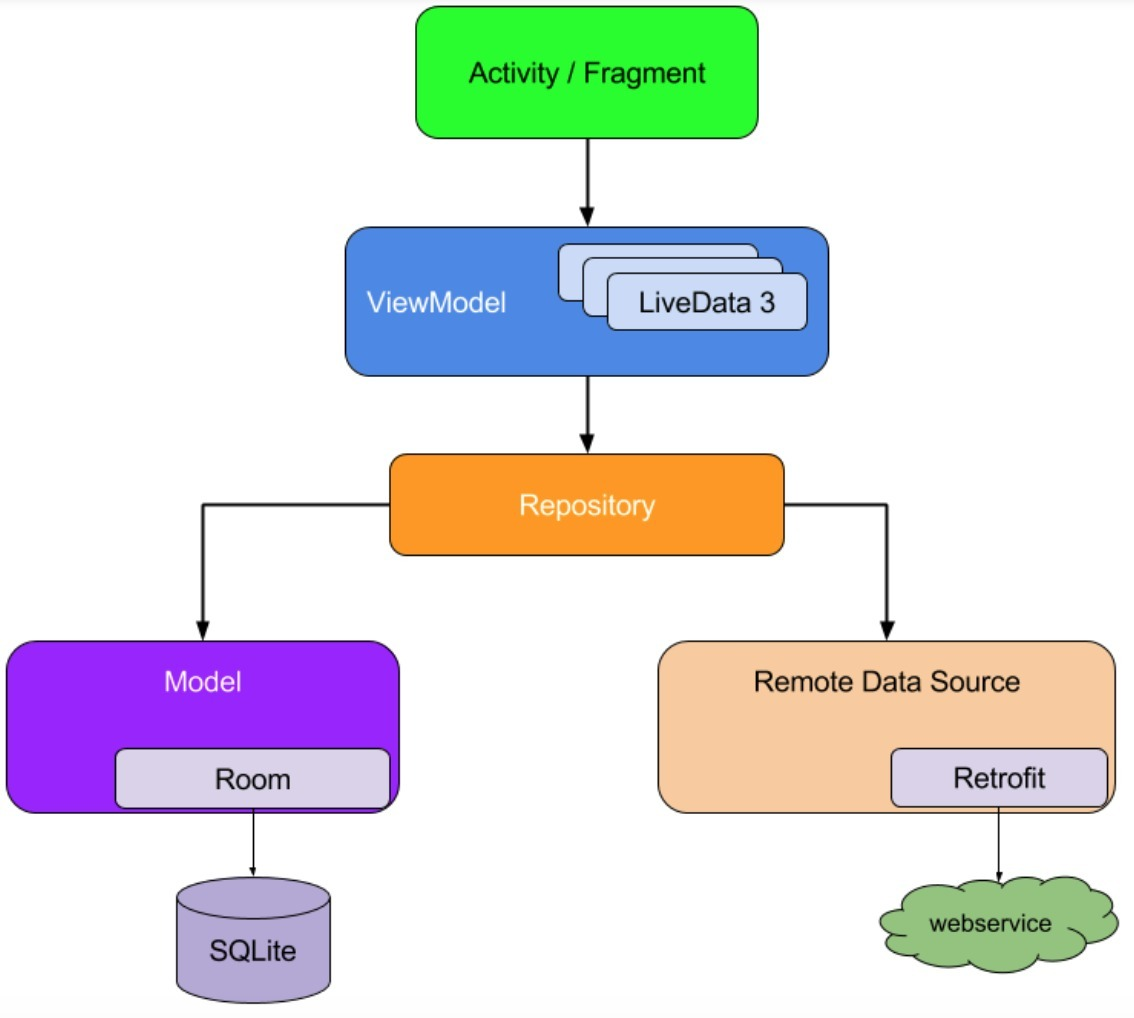
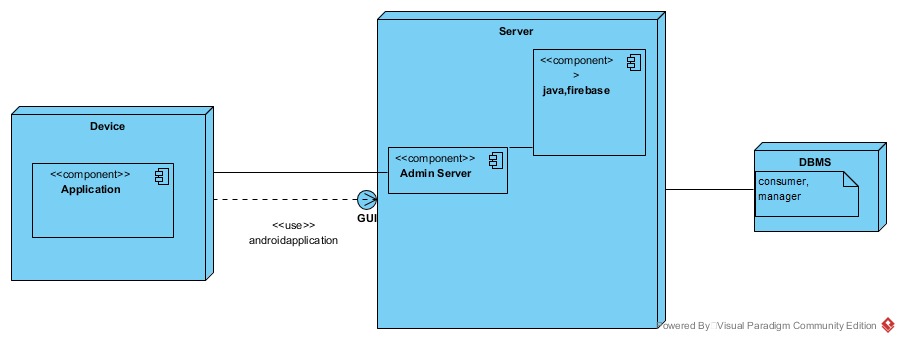
(System decomposition vpp yüklemesi gerekli)

In online bus ticket reservation, our subsystems are Customer interface, Guest interface, Admin interface, Manager interface, Login subsystem, Buy Ticket subsystem, Registration subsystem, Edit Bus Schedule subsystem, Manage Booking subsystem, Trip Info subsystem, Payment subsystem, Manage Account subsystem, Update User Info subsystem and Data Access subsystem. Customer interface contains login service, buy ticket service, payment service, manage account service and trip info service. Guest interface contains registration service and trip info service which includes list trips service. Admin interface contains login service, edit bus schedule service, manage booking service and update user info service. Manager interface contains login service, edit bus schedule service, manage booking service. Login subsystem provides service for all users to log in to the system. Buy ticket subsystem provides service to the customers who have already registered to the system to buy ticket which they searched. Registration subsystem provides service to guest to register to the system. Edit bus schedule subsystem provides service to manager and admin to add, update or delete the trips on the system. Manage booking subsystem provides service to admin and manager to delete or edit the booking which the customers have made on the system. Trip info subsystem provides service to customer and guest to see the trip details which they search. Payment subsystem provides service to customer to make payment on the system. Manage account subsystem provides service to customer to edit or freeze their account on the system. Update user info provides service to admin to update managers’ account information or authorities on the system. Data Access Subsystem; contains all our persistent objects, this part could be called Model of MVC.

## Hardware Software Mapping

The online bus ticket reservation system has three layers for reusability and readability, which are Presentation, Business and data Access layers. Data Access layer will use Google’s Firebase database management system for server connection. Presentation Layer will be the personal smart phone. Since we are writing codes in Java and our machines are compatiple with Android, we are using AndroidStudio on Presentation Layer.

(Data access layer ve business layer bilgileri eksik)



## Persistent Data Management

(Database tabloları gerekli)

There are some constant data stored by the system, which is vital for the system to be of any use, so that the data can outlive a single execution of the system. For this reason, in the online bus ticket reservation system, we store this data in a database. The persistent data recorded are; Users of the system (Customers, Guests, Admin and Managers), and their information. We can store a list of customers' bought tickets in their database of theirs and their personal information. In addition, we can also store in the database the managers' personal information. These are the persistent data that are needed to be stored in database so that the data can outlive a single execution of the system. Different tables from which information must be connected and extracted so they can be easily manipulated by operators such as project and join to give information in the form in which it is desired. Data independence is achieved more easily with normalization structure used in a relational database than in the more complicated tree or network structure. Looking at all these advantages that relational database provides us, we have decided that the relational database is the closest data management substructure for our system. Firebase, which is a relational database management system (RDBMS) will be used to manage and keep the data.

(Table tanımları gerekli)

## Access Control and Security

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Guest** | **Customer** | **Manager** | **Admin** | **Ticket** | **Trip** |
| **Guest** | Register() |  |  |  |  | ListTrips() |
| **Customer** |  | Login()  ManageAccount() |  |  | BuyTicket()  Payment() | ListTrips() |
| **Manager** |  |  | Login()  ManageAccount() |  | ManageBooking() | ListTrips()  EditBusSchedule() |
| **Admin** |  | UpdateUserInfo() | UpdateUserInfo() | Login()  UpdateUserInfo()  ManageAccount() | ManageBooking() | ListTrips()  EditBusSchedule() |

For online bus ticket reservation and buy system, the security is very important. The system should keep the personal information of the customers and the workers of the system which are admin and manager , safe and secure. Hence, the data of the system will be protected. With one weakness of the database, these data can be stolen. To prevent database injections, we will use prepared statements also we will use stored procedures if it is possible for that case and of course we will try to escape all user supplied input.

The system has database connection when a user tries to login. According to user type(customer, admin and manager) system decides which actor will login the site by looking the actor type from the database table. System has three sub database classes for actors which are database; customer, admin and manager. Admin’s webpage logged in will be different due to the security reasons. Each class has specific attributes and methods for each actor. The actors login is controlled with user\_type field in database.

## Global Software Control

The following section will describe the software control implementation. Centralized design will be good for our global software control.

Centralized Design; One control object or subsystem ("spider") controls everything.

Pro: Change in the control structure is very easy

Con: The single control object is a possible

performance bottleneck

**Multithreading:** The service manager uses threads, so that a large number of services is able to use the service manager simultaneously. The service manager also handles asynchronous events within the system.

**Avoiding Deadlocks:** Callbacks must not block the caller. So only status information is updated in the caller. All other internal work within a single service must be done by other worker threads.

**Worker Threads for Each Service:** Each service has an own thread for communication, which communicates with the service manager and other services.

**Callbacks Between Subsystems:**

Like the Trustbus service manager, the services use asynchronous callbacks for inter process communication. Every single service uses internal methods to notify the other subsystems of its own status, its needs and abilities.

## Boundary Conditions

Initialization:

Initialization:

Failure

# Subsystem Services

# References

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

2. Lecture presentations of the course (the presentations were provided by the Instructor who is Emine Ekin).

3. About application design: https://www.kamilkoc.com.tr/, https://www.pamukkale.com.tr/

4. We lookup other porjects from: www.github.com, www.researchgate.net