<Project Name>

System Design

<Version>

<Date>

<Your Name>

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]

The System Design Document (SDD) is written after the initial system decomposition is done, and updated throughout the development. SDD describes 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.

SDD is used to define interfaces between teams of developers and serve as a reference when architecture-level decisions need to be revisited. The audience for the SDD includes the project management, the system architects (i.e., the developers who participate in the system design), and the developers who design and implement each subsystem.

# Introduction

Provide a brief overview of the software architecture and the design goals. It also provides references to other documents and traceability information (e.g., related requirements analysis document, references to existing systems, constraints impacting the software architecture).

## Purpose of the System

Viatorem is a website to buy bus tickets. It has an easy to use interface, it is easy to manage for admins. Also, it provides direct communication chance with an officer. Viatorem is a replacement alternative for websites such as kamilkoç.

## Design Goals

Usability: Viatorem is easy to understand and use for ordinary internet users. Anyone who occasionally uses internet should not have any problem with using Viatorem.

Reliability: User password is not displayable even for admins and payment information is not shown anywhere except purchase page.

Performance: Registered users can buy as many tickets as they want.

Supportability: Admin should be able to organize trips and campaigns without any problem.

Implementation:

Legal: This software ins regulated under the DDNB(first letters of our names) general public license.

## Definitions, Acronyms, and Abbreviations

Viatorem is designed with using php, html, and mysql.

Admin: Admin is a user who is responsible from the system.

Visitor: A user who is not logged in to system.

Registered User: A user who io logged in to system.

Viatorem: A website where you can buy and reserve bus tickets from a city to another.

Officer: Officer is an employee whose job is answering mails sent through “contact us” section.

Interface: Interface is the sight of website which users see.

Html: Html is a programming language.

Php: php is a programming language.

MySQL: MySql is a database management system.

Subsystem: Subsystem is collection of classes that are closely related to each other.

## References

* KamilKoç
* Metro Tourism
* Biletall.com

# Current Software Architecture

Especially across the country, there are many different, advanced applications that offer the bus ticket purchase/reservation function, which is the main purpose of our application. An example of these applications is given below.

[www.kamilkoc.com](http://www.kamilkoc.com)

[www.metroturizm.com](http://www.metroturizm.com)

[www.obilet.com](http://www.obilet.com)

The main subsystem of these applications is to purchase or reserve bus tickets. Advanced and large applications such as these cater to a wide range of users, accommodating a range of system requirements. In addition to the technical requirements, they have a long operational requirement. Our application, on the other hand, does not appeal to a large audience as much as the applications mentioned, but to smaller audiences. Since it is a web-based application, the 3-Layered artchitecture structure is used as software architecture pattern. The system has got four actors; admin, registered user, visitor and officer who will contact with the registered users. Our system has a design that meets the technical requirements expected by the users and minimizes the problems that may arise.

# Proposed Software Architecture

At out system, users will be able to search for trips to their destination, select the one that suits them, and purchase or reserve. They can cancel these transactions if they want. It will not endanger users for safety reasons. User data privacy is extremely important and will keep the data as secure as possible. Users will be able to cancel the transactions they perform within the specified time limits, such as cancellation of purchased tickets and cancellation of the reservation. The user interface is successfully simple to understand and does not make the user difficult to use the application. Due to the creation of a fast data exchange channel between the client-side and the server-side, the user is intended to avoid time-consuming problems when using the application. Our system opens a window to the users to solve the problems encountered while using the system as soon as possible. Users are provided with a service to communicate any complaints or suggestions to the system administrator. Ticket information received by the user will be shown to the user through the application at any time. In this way, the user will benefit if they forgets the information.

## Overview

The system will have a layered architecture and there will be 3 layers; there are interface , application and data. Users will be interacting with the system by using the interface layer that designed in simple way to use, application layer will have all the functions of the system, for example buying/reservation of a ticket, interact with the officer, cancellations etc. and data layer will have all the information about the users, tickets, trips, messages etc.

Interface layer will have the AdminInterface, UserInterface and OfficerInterface subsystems.

Application layer will have the Login Subsystem, Registration Subsystem, TripManagement Subsystem, Campaign Subsystem, Communication Subsystem and UserManagement Subsystem. These subsystems will be responsible of login, registration, buy ticket, communication, edit trip, edit account, view trip and edit campaign services.

Data layer will have the TripInfo, TicketInfo, MessageInfo, CampaignInfo and UserInfo subsystems that has got the information about users, tickets, messages ,trips and campaigns.

Admin can be manage to add trip, cancel trip,add campaign, cancel ticket, view all trips.

Registered user and visitor will have the services of viewing trips, buying tickets and canceling tickets. Some of the functions are specific to registered user and visitor separately. In addition to the registered user subsystem, it will be responsible for contacting the officer, editing profile, login and booking services.

## 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

|  |  |  |  |
| --- | --- | --- | --- |
|  | Administration | User | Viatorem |
| Admin | addCampaign ()  addTrip()  cancelTrip()  cancelTicket() | login()  logout() | viewTrip() |
| Registered User |  | login()  forgotPassword()  editProfile()  logout()  checkMailBox() | contactWithOfficer()  giveFeedback()  reserveTicket()  purchaseTicket()  cancelTicket()  viewMyAllTicketst()  seeHelp()  viewTrip()  seeCampaign() |
| User (Visitor) |  | register()  logout() | viewTicketDetail()  purchaseTicket()  cancelTicket()  seeHelp()  viewTrip()  seeCampaign() |
| Officer |  | checkMailBox()  logout() | sendMessage() |

User password is held encrypted in database and not even admin can see it. Credit card information is not held in database. Therefore there is no need for a further encryption.

## Global Software Control

Admin Interface that positioned at Interface Layer initializes login service, communication service, edit trip service that are responsible of adding trip, cancellation trip, view trip service and edit campaign service.

User Interface that positioned at Interface Layer initializes login service, registration service, buy ticket service that are responsible of buying ticket, contains buying ticket, cancellation ticket and reservation ticket which is let only by the registered users, communication service, edit account service that is also let only registered users and view trip service.

Officer Interface that positioned at Interface Layer initializes login service and communication service.

All of these services run on Application layer.

Login Subsystem controls logging into the system and each actor can log in system.

Registration Subsystem controls registration service and only visitors can execute the registration function.

Trip Management Subsystem controls edit trip service that includes adding a trip and cancellation trip and view trip service that allows to admin and user to view trips.

Ticket Management Subsystem controls buy ticket service which is includes buying/reservation ticket and cancellation ticket.

Campaign Subsystem controls edit campaign service that are initializes by only admin.

Communication Subsystem controls communication service which contains function of contacting between user and officer also admin.

User Management Subsystem controls edit account service which allows the registered user to execute some changing on their profile.

Data Layer has Trip Info Subsystem, Ticket Info Subsystem, User Info Subsystem, Message Info Subsystem. Trip Info Subsystem controls the trips’ data, Ticket Info Subsystem controls the tickets’ data, User Info Subsystem controls the user’s data and Message Info Subsystem controls the messages’ data.

## 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

Login Service: Users can log in to the system through this service.

Registration Service: Visitors can register to the system through this service.

Buy Ticket Service: Users can buy or reserve tickets through this service.

Edit Ticket Service: Users can cancel their tickets through this service. Also admin can cancel user’s tickets if it is necessary.

Communication Services: Registered users and officer can communicate with each other through these services. Also Registered users can give feedbacks that are supervised by admin.

Edit Trip Service: Admin can add or cancel trips through this service.

Edit Account Service: Registered users can edit their account information through this service.

View Trip Service: Admin and users can view available trips through this service.

Edit Campaign Service: Admin can add or remove campaigns through this service.

# 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.