<Online Ticket System>

System Design

<1.0>

<11.12.2016>

HASAN MURAT AKPOLAT

ERTAN GÖKYILDIRIM

EMRE YILMAZ

MELİS KILIÇ

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

Online Ticket System provides an effective and online way to users to purchase ticket. Users do not need to go to ticket distributor by using this system. They can buy any ticket by using Online Ticket System. Moreover, Online Ticket System has included some services to clients, and admins. Clients can offer new events to customers. Clients add events ,time period of events and event type to these events. Admin add new places, regions and event types to the system for clients. Finally, purpose of the system is to provide efficient way to customers to purchase ticket, and purpose of the system is to provide efficient way to clients to add new events and purpose of the system is to provide efficient way to admin to manage system.

## Design Goals

Online Ticket System is an web-based ticket platform for users and clients. Therefore, it should provide some constraints, functional and non-functional requirements. For example, on users’ side, Online Ticket System design should provide that customers can see all events in any region, customers can select any date and then they can search all events by using this date. In addition, our important goal is to purchase ticket to users; this is main and important goal of our design. Also, our design should provide that clients can add new events to the system and they can add event type of an event, price and time period to these events by using Online Ticket System. Moreover, the other goals of our design Online Ticket System should besafety, should accept upgrades and should be implemented on Netbeans platform, In addition, “Php” should be used as the programming language. For Database the system should use “MySQL”. User Interface should be web-based.

## Definitions, Acronyms, and Abbreviations

RAD: Requirement Analysis Design

SDD: System Design Document

HTTP: The Hypertext Transfer Protocol

UI: User Interface

PHP: Hypertext Preprocessor

MYSQL: My Structured Query Language

## References

[**http://blog.slickedit.com/2007/05/how-to-write-an-effective-design-document/**](http://blog.slickedit.com/2007/05/how-to-write-an-effective-design-document/)

[**https://en.wikipedia.org/wiki/Software\_design\_description**](https://en.wikipedia.org/wiki/Software_design_description)

# Current Software Architecture

Nowadays, Web-based and mobile ticket applications are very popular. In Turkey, “biletix.com” is one of the most popular and largest online ticket system. With this kind of ticket systems, the user selects the region that where s/he lives and selects the best event for their purpose or their budget.

In our Online Ticket System, Users can search for events for free with signing in. Also Users select an event to buy a ticket; the system makes sure that the user is registered to the system. If not, the user fills the needed forms to register the system. After the registration, user can but his/her tickets easily and s/he can make the payment with the system. Also, users can check their own past/future events with their informations. A Client is a member too, who can register their events with their information. Clients can also change the information and the pricing of the event. Also The other functions of the client are add time period of event, add category to event and add seats of the event. In addition, admin can approve or reject the client registrations and also Admin can add new place, region, and event type into the system. In addition Admin can modify the event information and cancelation of events. Also Admin can update the places which are found into database.

Current system is being made by using PHP and HTML. Also, for database creation, MySQL Workbench is used. In all current online ticket systems, flow of events works in this way. However, details about the system can be vary from one to another such as which architecture the system use or is the system supported by any hardware equipment etc.

# Proposed Software Architecture

Documents the system design model of the new system.

## Overview

Present a bird’s-eye view of the software architecture and briefly describes the assignment of functionality to each subsystem.

## System Decomposition

Describe the decomposition into subsystems and the responsibilities of each. This is the main product of system design.

## Hardware Software Mapping

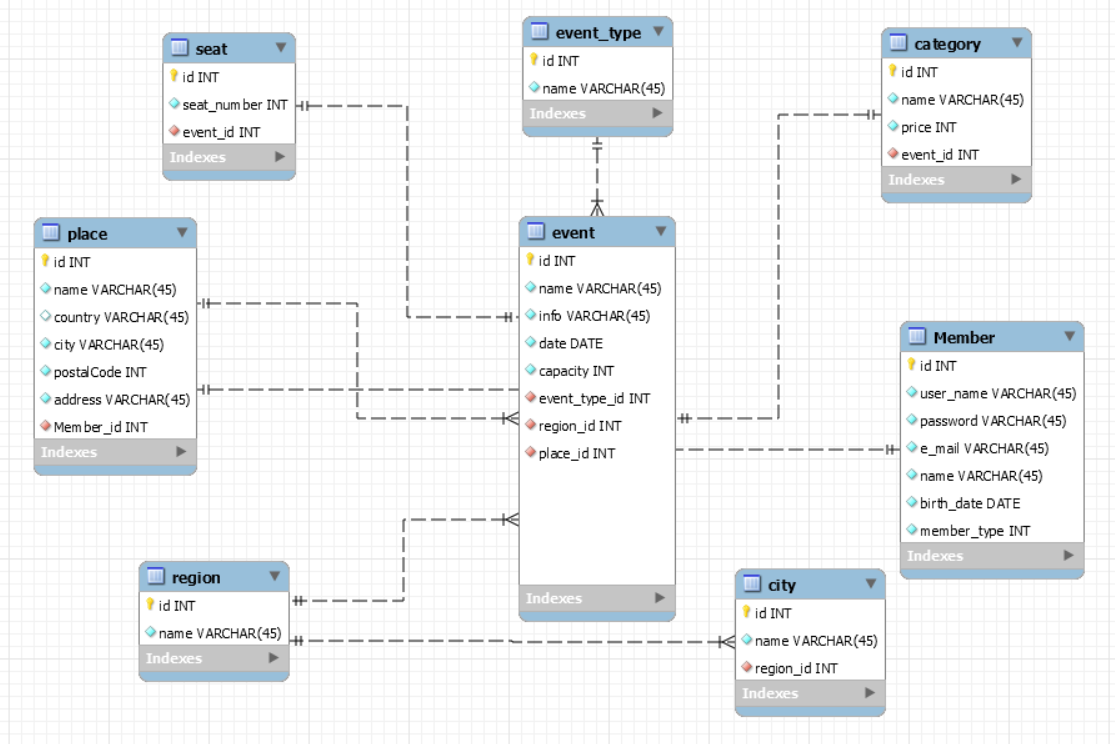
Describe how subsystems are assigned to hardware and off-the-shelf components. It also lists the issues introduced by multiple nodes and software reuse.

## Persistent Data Management

In Online Ticket system, we decide to store persistent data in a database. The persistent data is; Users of the system (User, Client, Admin), Region data that displays regions’ names. Place data that displays place’s features like city, post code, address. There is relation between place and client. We store client in member data. For each client there are some places. Event Type data shows event type name like as music, sport, and so on. Event table shows event type, date, capacity, event name, region\_id, place\_id and information of event. There is relation between region and event. For each region there are some events. There is relation between event type and event. For each event type there are some events. There is relation between place and event. For each place there are some events. Category data displays some categories like as X categories, Y categories. There is relation between category and event. For each event there are some categories. Seat data displays seat number. There is relation between seat and event. For each event there are some seats.

In Online Ticket despite its advantages for the Online Ticket system, we decided to use **relational database** for the data management infrastructure. Some advantages: Security, flexibility, easy use and so on.

Here Online Ticket’s ER diagram:



## Access Control and Security

Describe the user model of the system in terms of an access matrix. This section also describes security issues, such as the selection of an authentication mechanism, the use of encryption, and the management of keys.

## Global Software Control

Describe how the global software control is implemented. In particular, this section should describe how requests are initiated and how subsystems synchronize. This section should list and address synchronization and concurrency issues.

## 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

In our design, the system has a lot of subsystems. These are: User Interface, Region Interface, Category Interface, Seat Interface, EventType Interface, Event Interface, Place Interface, Login Interface,

Decoupling our application from the datasource means that the components used to implement the business logic and flow of control are not tied to the specific datasource. By utilising a pattern that adheres to this paradigm, we can easily switch out the datasource for a different kind of database or a different kind of storage medium without requiring code changes throughout the rest of our application.

Typically we achieve this by adding layers of abstraction to the data retrieval and storage process. In a web application that follows an MVC-based architecture, all of the code responsible for managing domain entities will be stored in Model classes.

Also we used the PDO(PHP Data Objects) in our project. PDO\_MYSQL is a driver that implements the PHP Data Objects (PDO) interface to enable access from PHP to MySQL 3.x, 4.x and 5.x databases.

**User Interface:** provides services for users (user, admin and client) for common interfaces, It has Login Form, Password Change Form, and etc.

**Region Interface**: provides services for users to display all regions and to select this region from box. And also provides services for admin to add new region into system.

**Category Interface:** provides services for users to display all categories and to select this category. And also provides services for client to add new category into any event.

**Seat Interface:** provides services for users to display all seats and to select this seat. And also provides services for client to add new seat into any category.

**EventType Interface:** provides services for users to display all types and to select this type(music, sport). And also provides services for admin to add new type to the system. And it provides services for client to add event type into any event.

**Event Interface:** provides services for users to select event whatever they want. And also it provides services for client to add new event into the system.

**Place Interface:**  provides services for admin to add new place into the system. And also provides services for admin to add place into the any event.

**Login Subsystem:** provides services for users (user, client, and admin) to login.

**Profile Info Subsystem:** provides services for user, client, and admin to update their personal information. In addition, the subsystem also provides the service of changing password for user, client and admin.

**Data Access Subsystem:** contains all our persistent objects, this part could be called Model of MVC.

.

# 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.