Online DrugStore

System Design Document

1.0

08.12.2016

Emre Yılmaz

Çağla Çınar

Mehmet Sefa Korkmaz

Ertuğrul Koç

Prepared for

SE301 Software Engineering



Table of Contents

[1. Introduction 1](#_Toc500528764)

[1.1. Purpose of the System 1](#_Toc500528765)

[1.2. Design Goals 1](#_Toc500528766)

[1.3. Definitions, Acronyms, and Abbreviations 2](#_Toc500528767)

[1.4. References 2](#_Toc500528768)

[2. Current Software Architecture 2](#_Toc500528769)

[3. Proposed Software Architecture 2](#_Toc500528770)

[3.1. Overview 3](#_Toc500528771)

[3.2. System Decomposition 3](#_Toc500528772)

[3.3. Hardware Software Mapping 5](#_Toc500528773)

[3.4. Persistent Data Management 6](#_Toc500528774)

[3.5. Access Control and Security 6](#_Toc500528775)

[3.6. Global Software Control 9](#_Toc500528776)

[3.7. Boundary Conditions 9](#_Toc500528777)

[4. Subsystem Services 10](#_Toc500528778)

[5. References 11](#_Toc500528779)

SYSTEM DESIGN DOCUMENT

# Introduction

Design is the abstraction of an answer; it's the common description of the answer to an issue without a details. Design is view sample seen within the evaluation part to be a sample in a design part. After design part we are able to reduce the time required the implementation.

## Purpose of the System

As we mentioned in Requirement Analysis Document, the aim of online Drug Store system provides simple interface for user users to buy drugs that they need. The system is allows users to buy drugs without going out to go to pharmacy. The system’s point, that makes difference between others, is allowing the users to buy drugs which are prescriptive.

## Design Goals

The design objectives signify the specified qualities of on-line examination system and supply a constant set of standards that should be thought of when making design choices. Based on non-functionality requirements the next design targets must achieved as a way to qualify the system as profitable:

* **Security**

The system security is one of the most important non-functional requirements.

* **Reliability**

The system has to carry out the Drugstore system operations with no errors. The web site developed needs to be extremely dependable and safe in order that details about any questions and many others shouldn’t be leaked earlier than the precise examination is held.

* **End User**

Users can buy drugs if they are logged in, however can open website to look at the drugs before buying or just checking some data. System should be able to deal with a number of users. This system should run on a number of operating systems and support windows operating system.

* **Performance**

The system has to be sturdy enough to manage any valid input from the users.

Moreover, the other goals of our design Drugstore system should accept upgrades and should be implemented on NetBeans platform, In addition, “JSP” should be used as the programming language. For Database the system should use “MySQL”. User Interface should be web-based.

DS System’s architectural style is Modal-View-Controller

## Definitions, Acronyms, and Abbreviations

**User**: The system user who wants to buy some drugs.

**Admin**: The system administrator who will manage all data system data and user controls.

**Doctor**: The user who will prepare and submit the prescriptions for their users into the system and can manage the prescriptions.

**RAD:** Requirement Analysis Design

**SDD:** System Design Document

**MySQL**: My Structured Query Language

**JSP:** Java Script Page

## References

**[1]-**https://theonlinedrugstore.com/

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

**[3]-**<https://en.wikipedia.org/wiki/Software_design_description>

# Current Software Architecture

There is an existing system for selling drugs [1]. The site could show all drugs and their users can choose and look at the all drugs and their information. Moreover, they have carts to buy drugs like all online shopping sites. They could also delete from cart some drugs. Then if users want to buy it they could buy drugs with considering the drugs are prescriptive or non-prescriptive. The fact that if the drug is prescriptive they want users to enter their prescription information. At the end, they sell the drugs with getting user’s credit card and location information.

The problem that we consider is easiness with interaction between doctor and prescriptions for users. In our system, doctor could register and login the system and has a profile. The point is doctor could manage the prescriptions currently.

# Proposed Software Architecture

Drugstore system is a web-based online application. This project would be very useful for users and doctors. Our system has a difference of user-friendly interface, easy to use and reliable, than the others.

Briefly, in our system (DS), the system allows users to search and look at the drugs, then if users want to buy drugs, they could add the drugs to the cart and they also could delete drugs from cart. Later on, they could buy the drugs with considering that drugs are prescriptive or non-prescriptive with entering the prescription information then entering the credit card and location information. Furthermore, the system also allows doctor to do that listing the prescriptions, adding prescriptions, deleting prescriptions, managing prescriptions currently.

## Overview

In our design, the system has a lot of subsystems. These are: User Interface, DrugType Interface, Drug Interface, Prescription Interface, Login Interface,Update Profile Information are subsystems for the Presentation Layer. We use “MySQL”. for data access layer.

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

**Prescription Interface:** provides services for users to buy drug whatever they want. And also it provides services for doctor to add new prescription into the system.

**DrugType Interface:** provides services for users to display all types and to select this type (normal, prescription drug).

**Drug Interface:** provides services for users to select drug whatever they want. And also it provides services for admin to add new drug into the system.

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

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

## System Decomposition

The decomposition shows the existence of the following subsystems:

* Admin management subsystem
* User management subsystem
* Doctor management subsystem
* Database subsystem

**Admin management subsystem**

This subsystem managing admin accounts. It offers perform for creating an account, updating an, approve and close an account for admin side. Admin is the only actor who has permission to access close and approves functions. Create and update functions are accessed by doctor and user. This subsystem uses login services of the user management and doctor management subsystems

**The operations provided by this subsystem are:-**

* Login ()
* Add Doctor ().
* Delete Doctor ()
* Approve Request of Doctor ()
* List Of Doctor (), List of Drugs ()
* Add Drug ()
* Delete Drug ()
* Logout ()
* Update Drug ()

**User management subsystem**

This subsystem is managing user actor’s function, offers user side to its functions. Managing users access to list of drugs. After the registering and logging step, adding to cart and deleting from cart and buying the drugs.

**The operations provided by this subsystem are:-**

* Login ()
* Register ()
* Searching Drugs ()
* Adding to cart ()
* Canceling drugs from cart ()
* Buying the drug ()
* Logout ()

**Doctor management subsystem**

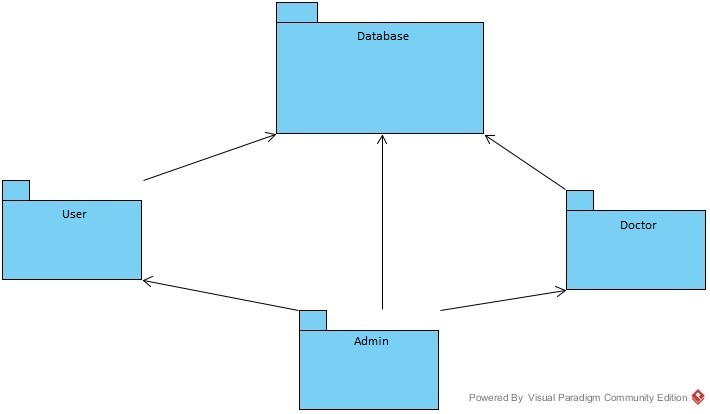
This subsystem is managing doctor actor’s function, offers doctor side to its functions after authenticate. The subsystem managing access of doctor to creating prescription, managing prescriptions of the users which is connected with database subsystem thus the user which are added prescriptions are available to buy prescriptive drugs with considering which drugs are added into the prescription of the user.

**The operations provided by this subsystem are:-**

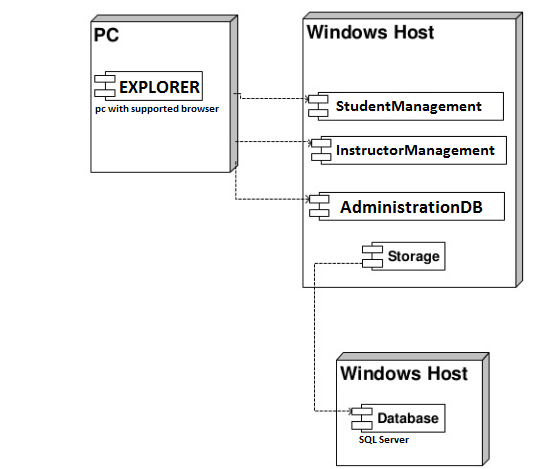
* Register ()
* Login ()
* Create New Prescription ().
* List Prescription ()
* Delete Prescription ()
* Update Doctor Information ()

**Database subsystem**

This subsystem will be implemented by relational database management system used to store the persistent data. All subsystems are related and having service with this subsystem

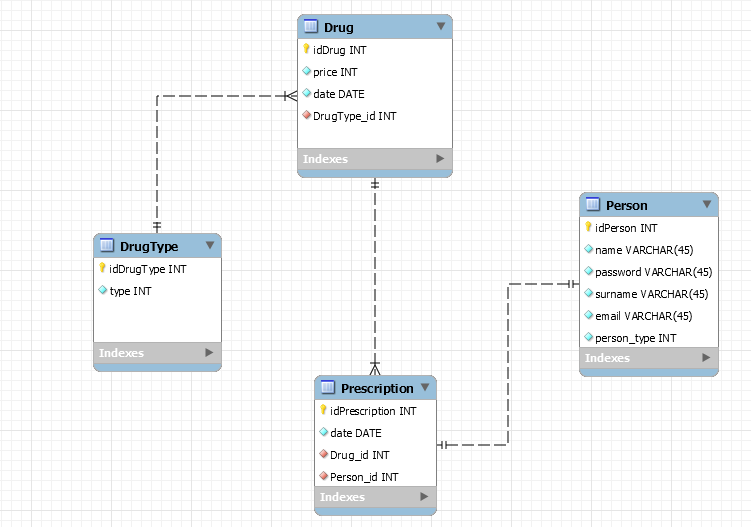


## Hardware Software Mapping



## Persistent Data Management

Our system will use the SQL Server database engine to store data. This will allow the database to be easily integrated with and accessed by the rest of the system. The database will retain user information for functions such as login for user, show prescriptions that are created by doctors for users to see etc. Our database structure is seen below with entity field’s relations etc.



## Access Control and Security

The application is a multiuser application so it consists of 3 types of users which are doctor, user and administrator. Because of this, the application will provide different interfaces for each user type.

First, the administrator will connect to the system with the membership interface, and will do the administrator's duties on the dating system, such as adding, updating, editing, deleting etc. By the way, registration is not necessary for the administrator through the system website, information will be entered manually into the database at the beginning of the system and the administrator will be the authority that will access the database directly. As a summary, the administrator does not have to register because it is initially registered in the database and the system. The administrator approved the registering of the doctors. Each registration process included the doctor registration made by the doctors themselves, will be done with the user interfaces of the system. The system will store all the information in the database and in the login processes again the system will use them by collecting data from the database. The information in the database will use both the confirmation and the use of the system for users. All types of users (doctors and patients) must log in to the system with their username and password.

After these steps, the system will be ready for the creating the prescription of the doctor. The doctors will register in the application, the system will send the request to admin and admin will approve the request of the doctor, the system will add a new doctor to the database. The doctor will then connect to the system with your information such as username and password. During the login procedure, the data in the user database table will be obtained and compared to the data entered by the doctor. Since this operation requires read-only access to the database, it can be performed from different access points simultaneously.

During registration, field filling does not require access to the database, while completion of the process requires the data to be written to the database, which requires read and write access to the database. In that case, the required database fields will be blocked and simultaneous access of multiple usernames will be denied.

For some situation like updating or deleting information it is necessary to update one of the tables in the database in its phase of completion and therefore must be handled with more care since several users(doctors and patients) can be the cause of updating the table at the same time. This will also be avoided by blocking.

Finally, viewing the information or lists again requires read-only access to the database. Therefore, multi-user access does not impose problems and new restrictions.

As last words, the usernames and passwords of users will be stored in the user table. No one else accepts the administrator can have access to this information. Authentication interfaces are different for each type of user and will be directed to their own main pages after the login process.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Actors/  Classes | User | Authentication | Prescriptions | Drugs |
| Admin | createDoctor()  deleteDoctor()  approverequestofDoctor()  listofDoctor()  approveRequest() | Login()  logout() |  | addDrug()  deleteDrug()  updateDrug()  listofDrugs() |
| Doctor | updateDoctorInformation() | Register()  Login()  logout() | addPrescription()  listofPrescription()  deletePrescription() |  |
| User |  | Register()  Login()  logout() |  | searchingDrugs()  buyingDrugs()  cancelDrugs()  addtoCart() |

Figure 3.1 Access Control Matrix for DS System

Admin, User, {createDoctor (), deleteDoctor (), approverequestofDoctor (), listofDoctor (),

approveRequest ()}

Doctor, User, {updateDoctorInformation ()}

Admin, Authentication, {login (), logout ()}

Doctor, Authentication, {login (), logout (), register ()}

User, Authentication, {login (), logout (), register ()}

Doctor, Prescriptions, {addPrescription (), listofPrescription (), deletePrescription ()}

Admin, Drugs, {addDrug (), deleteDrug (), updateDrug (), listofDrugs ()}

User, Drugs, {searchingDrugs (), buyingDrugs (), cancelDrugs (), addtoCart ()}

## Global Software Control

External Control Flow (Between Subsystems): DS system defined by the web application with a simple feature. Web server requests request submission of user data. Because the system is multi-user, simultaneous executions can occur. However, the control flow of a single user has a predefined form. After the login step, the system has a web page structure in the form of a tree formed by links or buttons.

Concurrent Control: Because the application is web-based, all subsystems and components can run simultaneously for different users (patients, doctors) in the application.

Internal Control (Within a Single Process): The process control is carried out by means of the designed forms on the web. The system is based on the page structure of the page request page. This makes the designed procedures simple and mostly linear. However, procedure calls can be made to other subsystems or to the current subsystem. Threads or multiple processes can be required for a process. The system uses a database so that the response time from the database should be minimized.

User Interface: The system user interface will be made through web pages. The control of the next step depends on the user. In addition to this, the flow is implemented within the web page. Most subsystems have a different web page. Due to the system event-driven design, subsystems cannot be considered to have their own event loop. However, events are controlled by web pages.

## Boundary Conditions

Startup: go to system URL and login

Shut Down: click log out and close browser

Error Conditions:

* Logging in:
  + Username or password field can’t be blank.
  + Username is not a 5 digit decimal number.
  + Password is not 8 characters long.
  + Password and username don’t match.
  + Username is wrong or does not exist.
  + The welcome screen does not appear after logging in.

* User settings
  + User is unable to change certain settings or changes don’t reflect.
  + Between the time of editing and updating, the system crashes.
* Data Entry
  + The system fails when the dispatcher is entering information.

* Doctor Entry
* Doctor information can’t be exist.
* Doctor information can’t be blank.
* Prescription entry
* Prescription id, date can’t be blank.
* Date can’t be a date before today.
* Logging out
  + Dispatcher unable to logout.

# Subsystem Services

In our design, the system has a lot of subsystems. These are: User Interface, DrugType Interface, Drug Interface, Prescription Interface, Login Interface, Update Profile Information,

Decoupling our application from the data source means that the components used to implement the business logic and flow of control are not tied to the specific data source. By utilizing a pattern that adheres to this paradigm, we can easily switch out the data source 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 MVC in our Project

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

**Prescription Interface:** provides services for users to buy drug whatever they want. And also it provides services for doctor to add new prescription into the system.

**DrugType Interface:** provides services for users to display all types and to select this type (normal, prescription drug).

**Drug Interface:** provides services for users to select drug whatever they want. And also it provides services for admin to add new drug into the system.

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

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

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

# References

**[1]** **https://theonlinedrugstore.com/**