|  |
| --- |
| Designer drug database |
| Design Document |
| Senior Project, CIS 4911- U01 |
| **Professor: Seyedmasoud Sadjadi Mentor: Dr. Luis Arroyo** |
| **Team Member: Carlos Dominguez** |
| **2/2/2015** |

|  |
| --- |
|  |

Copyrights and Trademarks Notices

Nothing yet

Abstract

One or two paragraph

Table of Contents

Contents

[1. Introduction 5](#_Toc410657622)

[1.1 Problem definition (very similar to RD). 5](#_Toc410657623)

[1.2 Design methodology used e.g., identify software process model, ease of creating a design from the systems requirements, types of models (UML models) used to represent the design. 5](#_Toc410657624)

[1.3 Definitions, acronyms, and abbreviations. 5](#_Toc410657625)

[1.4 Overview of document 5](#_Toc410657626)

[2. System Design (i.e., overall system design) 6](#_Toc410657627)

[2.1 Overview – high-level description of the system design (architecture) e.g., provides a package diagram showing the major subsystems and briefly describes each subsystem. Relate the system decomposition to the requirements of the system. Use at least two (2) architectural patterns. 6](#_Toc410657628)

[2.2 Subsystem Decomposition – provide a detailed description for each of the major subsystems. Identify the requirements associated with each subsystem. 6](#_Toc410657629)

[2.3 Hardware and Software Mapping – map subsystems to h/w and s/w. The h/w and s/w are for the systems to be implemented. May include a deployment diagram showing the associations between the subsystems and hardware. 7](#_Toc410657630)

[2.4 Persistent Data Management – identify data that needs to be stored and the structure of the data. Use a data dictionary to represent the initial data extracted from the use cases. 7](#_Toc410657631)

[2.5 Security/Privacy – describe user authentication processes, encryption of data, and use of firewalls or security servers. 8](#_Toc410657632)

[3. Detailed Design 9](#_Toc410657633)

[3.1 Overview – briefly describe the behavior and structure of each subsystem. 9](#_Toc410657634)

[3.2 Static model – detailed description of the structure for each subsystem. May include detailed class diagrams. Place diagrams (e.g., minimal class diagram, detailed class diagram per subsystem) inline. Use at least four (4) design patterns. 9](#_Toc410657635)

[3.3 Dynamic model – state machine diagram for the main control object in each subsystem. Include the design of the *main algorithms* used in the problem solution. Refinement of the sequence diagram from the analysis model. Place diagrams inline. 9](#_Toc410657636)

[3.4 Code Specification - describe the class interfaces (attributes and method signatures) and constraint (invariants, pre-condition and post-conditions) for the main control object in each system. Code should be in Appendix C. 9](#_Toc410657637)

[4. Glossary - define terms used in document, especially domain specific terms. 10](#_Toc410657638)

[5. Appendix 11](#_Toc410657639)

[5.1 Appendix A - Use case diagram for use cases being implemented. 11](#_Toc410657640)

[5.2 Appendix B - Use cases being implemented (from the RD). 11](#_Toc410657641)

[5.3 Appendix C – Documented class interfaces (code) for the subsystem(s) you will implement and the constraints. 11](#_Toc410657642)

[5.4 Appendix D - Diary of meeting and tasks. 11](#_Toc410657643)

[6. References 12](#_Toc410657644)

# Introduction

This document will introduce in detail the design of the system and how each part works.

## 1.1 Problem definition (very similar to RD).

A design needs to be created for a better implementation. I will describe in detail how the system is going to be design for future maintenance and better implementation of the system.

## 1.2 Design methodology used e.g., identify software process model, ease of creating a design from the systems requirements, types of models (UML models) used to represent the design.

The design will be represented mainly by a design diagram.

## 1.3 Definitions, acronyms, and abbreviations.

**HTML:** Hypertext Markup Language, a standardized system for tagging text files to achieve font, color, graphic, and hyperlink effects on World Wide Web pages.

**AJAX:** Ajax (also AJAX; /ˈeɪdʒæks/; short for asynchronous JavaScript and XML) is a group of interrelated Web development techniques used on the client-side to create asynchronous Web applications.

**Javascript:** an object-oriented computer programming language commonly used to create interactive effects within web browsers.

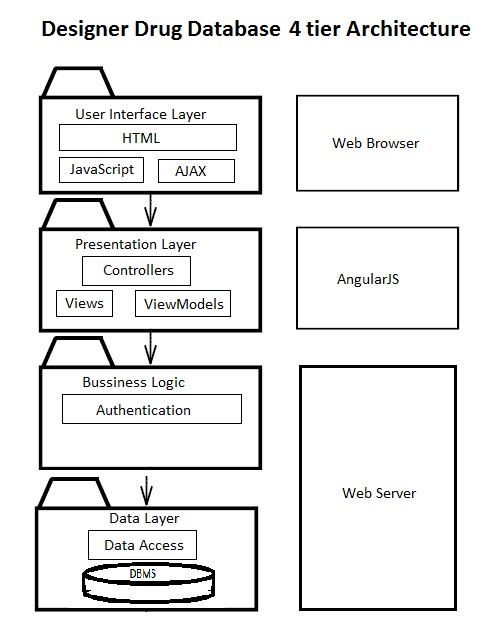
## 1.4 Overview of document

The following document is organized in system design and detail design. In system design, or section 2, we will talk about…., while in detail design we will talk about …

# System Design (i.e., overall system design)

The system decomposition decomposes into two subsystems, user components and compound components. Both of these two components will access a storage component that in term accesses the database.

## 2.1 Overview – high-level description of the system design (architecture) e.g., provides a package diagram showing the major subsystems and briefly describes each subsystem. Relate the system decomposition to the requirements of the system. Use at least two (2) architectural patterns.



The high level description of system design consists of 4 layers. The 4 layers are:

- User Interface Layer

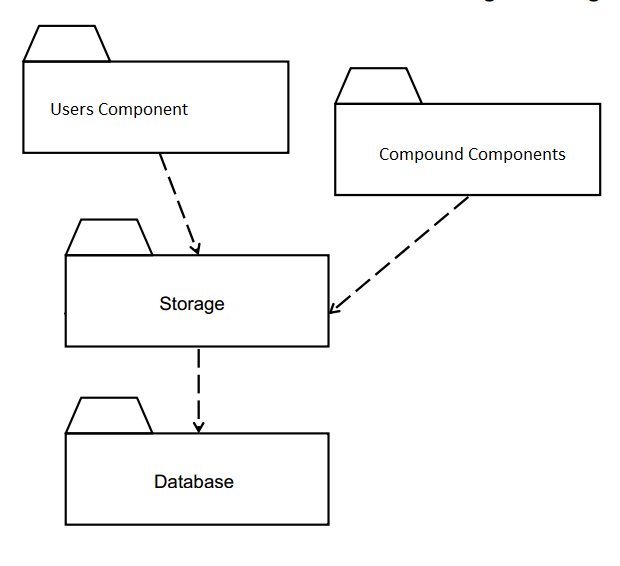
- Presentation Layer

- Business Logic

- Data Layer. On the

The user interface layer is basically the template, made with HTML. This template can be modified using JavaScript, css and other technologies. The Presentation layer consist of the controllers, view and view-models, which represents the business logic on the client. The business logic on the web server is in charge of the authentication and the data layer is in charge of the data.

## 2.2 Subsystem Decomposition – provide a detailed description for each of the major subsystems. Identify the requirements associated with each subsystem.

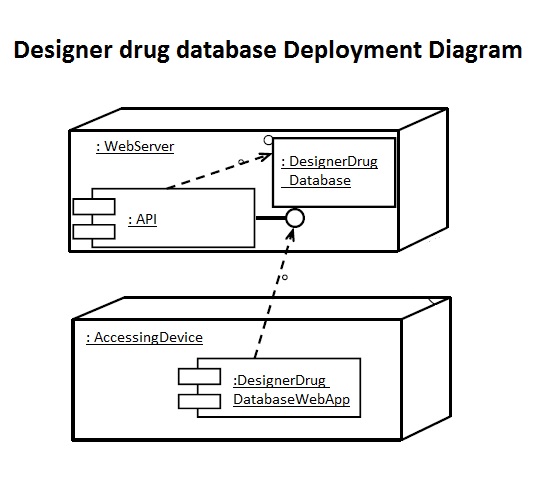


The user component will do all user operations. A user can be added, deleted, and modified. All this is possible given that the user has the appropriate permission. Moreover, the user component must handle request of name, email, and any other data storage in the user table and related tables.

The compound component is in charge of all compound information. It should be capable of adding a compound, modifying it and deleting. The compound component should also be capable of importing data and export data in different formats.

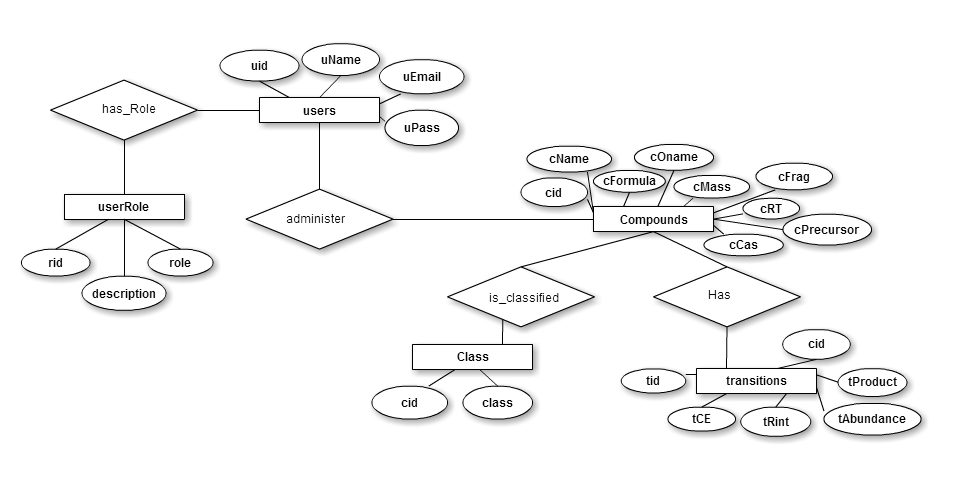
## 2.3 Hardware and Software Mapping – map subsystems to h/w and s/w. The h/w and s/w are for the systems to be implemented. May include a deployment diagram showing the associations between the subsystems and hardware.

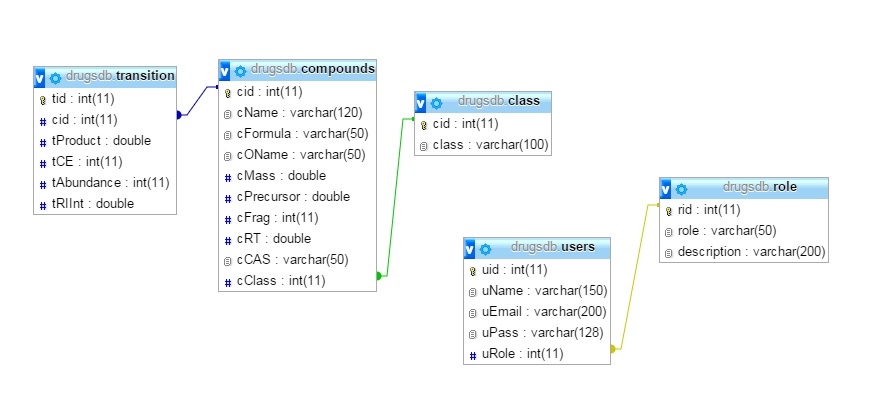
The hardware and software mapping as the following figure shows, consist of two hardware devices and all the software running on them. On the device that is accessing the single page application the app is fun and any data request is sent to the webserver, who responds with data, then the requested device process the information and display it.



## 2.4 Persistent Data Management – identify data that needs to be stored and the structure of the data. Use a data dictionary to represent the initial data extracted from the use cases.

The persistent data to be stored will be stored in databases. The following relational tables is the way in which the data is to be stored.





## 2.5 Security/Privacy – describe user authentication processes, encryption of data, and use of firewalls or security servers.

User authentication will use the sessions as a way to maintain persistence between all the pages and as a way to authenticate that a user has the right to access to certain information.

Encryption of data is only used for password and the type of encryption in use will be MD5.

# Detailed Design

The detail design

## 3.1 Overview – briefly describe the behavior and structure of each subsystem.

The user component will do all user operations. A user can be added, deleted, and modified. All this is possible given that the user has the appropriate permission. Moreover, the user component must handle request of name, email, and any other data storage in the user table and related tables.

The compound component is in charge of all compound information. It should be capable of adding a compound, modifying it and deleting. The compound component should also be capable of importing data and export data in different formats.

## 3.2 Static model – detailed description of the structure for each subsystem. May include detailed class diagrams. Place diagrams (e.g., minimal class diagram, detailed class diagram per subsystem) inline. Use at least four (4) design patterns.

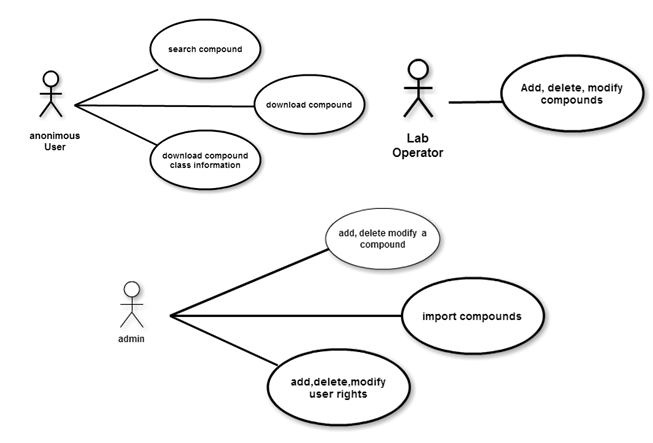
## 3.3 Dynamic model – state machine diagram for the main control object in each subsystem. Include the design of the *main algorithms* used in the problem solution. Refinement of the sequence diagram from the analysis model. Place diagrams inline.

## 3.4 Code Specification - describe the class interfaces (attributes and method signatures) and constraint (invariants, pre-condition and post-conditions) for the main control object in each system. Code should be in Appendix C.

# Glossary - define terms used in document, especially domain specific terms.

Appendix

## 5.1 Appendix A - Use case diagram for use cases being implemented.



|  |
| --- |
| Name: SearchForCompound |
| Participating actor: any of the three actors in the system |
| Entry condition:   * User is in main page or home page |
| Exit condition:   * User found the data of the search compound |
| Event flow:   1. User click on the box where the search criteria is supposed to be entered 2. User enters criteria to search a compound, which could be the name of the formula of the compound. 3. User clicks the search button. 4. If search produced more than one result. - User will get a window to pick the right compound, once it picks the right compound, it will go to the page with the data of the compound - Otherwise it will go to the page with the information of the compound. |
| Alternative Exit condition: Compound is not found and user is prompted |

|  |
| --- |
| Name: DownloadCompoundClass |
| Participating actor: any of the three actors in the system |
| Entry condition:   * User is in main page or home page |
| Exit condition:   * User saved to his computer the desired data |
| Event flow:   1. User searches for the class of the compound he desires to download in the class download list 2. User click download button 3. A window is pop up to ask user where he wants to save the information 4. User pick location where information will be saved 5. User click save |

|  |
| --- |
| Name: addCompound |
| Participating actor: either an admin or a labOP |
| Entry condition:   * User is logged in the system * User is in their account home page. |
| Exit condition: Compound data is added in the system |
| Event flow:   1. User click on the compound menu option 2. User click on the add compound button 3. User selects the picture of the compound 4. User add the following data of the compound in their respective fields: name, formula, other name, mass, fragments, retention time, precursor, CAS 5. User selects the class this compound is from the select option. 6. User adds as many transitions as the compound has by clicking in a Add transition button, entering the information and click on add button. 7. User click submit all button. |

|  |
| --- |
| Name: deleteCompound |
| Participating actor: either an admin or a labOP |
| Entry condition:   * User is logged in the system * User is in their account home page. |
| Exit condition: Compound data is deleted from the system |
| 1. Event flow: 2. User click on the compound menu option 3. User click enter the name or formula in the search box. 4. User clicks the search button. 5. If search produced more than one result.  * User will get a window to pick the right compound, once it picks the right compound, it will go to the page with the data of the compound to be modified * Otherwise it will go to the page with the information of the compound to be modified.  1. User clicks delete compound. |
| Name: editCompound |
| Participating actor: either an admin or a labOP |
| Entry condition:   * User is logged in the system * User is in their account home page. |
| Exit condition: Compound data is updated in the system |
| 1. Event flow: 2. User click on the compound menu option 3. User click enter the name or formula in the search box. 4. User can modify any of the following data:  * name, formula, other name, mass, fragments, retention time, precursor, CAS. * Different class in the class select menu for this compound. * adds as many transitions as wanted by clicking in the Add transition button, entering the information and click on add button. * delete as many transitions as wanted by locating desire transition and clicking in the Delete transition button. * editing as many transitions as wanted by locating desire transition and clicking in the edit transition button, entering the information and click on update button.  1. User click submit all button. |

|  |
| --- |
| Name: importCompound |
| Participating actor: either an admin or a labOP |
| Entry condition:   * User is logged in the system * User is in their account home page. * User file to be imported meets the following criterias:   + Has a csv extension   + Keeps the following rows: Compound Name, Compound Class, Formula, Mass, Precursor, Product, No. of Transitions, Frag, CE, Abundance, Relative Ion Intensity, RT (Zorbax), CAS |
| Exit condition: Compounds data in csv file has been added to the system |
| Event flow:   1. User click on the compound menu option 2. User click in the import compound button 3. A pop up menu will show for user to choose the file he wishes to import. 4. User selects the csv file 5. User clicks submit button. |

|  |
| --- |
| Name: userModifyItsName |
| Participating actor: either an admin or a labOP |
| Entry condition:   * User is logged in the system * User is in their account home page. |
| Exit condition: User Information is updated |
| Event flow:   1. User click on the account details menu 2. User click where its name is displayed on the account details panel. 3. A pop up menu will show for user to modify its name. 4. User delete old name and enters the new name. 5. User clicks submit button. |

|  |
| --- |
| Name: userModifyItsPassword |
| Participating actor: either an admin or a labOP |
| Entry condition:   * User is logged in the system * User is in their account home page. |
| Exit condition: User Information is updated |
| Event flow:   1. User click on the account details menu 2. User click where password is displayed on the account details panel 3. A pop up menu will show for user to modify its password. 4. User enters the old password, new password and then again the new password to confirm a mistake was not made. 5. User clicks submit button. |

|  |
| --- |
| Name: AddNewUser |
| Participating actor: admin user |
| Entry condition:   * User is logged in the system * User is in their account home page. * User has admin rights |
| Exit condition: User Information is updated |
| Event flow:   1. User click on the account details menu 2. User click where password is displayed on the account details panel 3. A pop up menu will show for user to modify its password. 4. User enters the old password, new password and then again the new password to confirm a mistake was not made. 5. User clicks submit button. |

|  |
| --- |
| Name: editUserRights |
| Participating actor: either an admin or a labOP |
| Entry condition:   * User is logged in the system * User is in their account home page. |
| Exit condition: User Information is updated |
| Event flow:   1. User click on the account details menu 2. User click where password is displayed on the account details panel 3. A pop up menu will show for user to modify its password. 4. User enters the old password, new password and then again the new password to confirm a mistake was not made. 5. User clicks submit button. |

|  |
| --- |
| Name: deleteUser |
| Participating actor: either an admin or a labOP |
| Entry condition:   * User is logged in the system * User is in their account home page. |
| Exit condition: User Information is updated |
| Event flow:   1. User click on the account details menu 2. User click where password is displayed on the account details panel 3. A pop up menu will show for user to modify its password. 4. User enters the old password, new password and then again the new password to confirm a mistake was not made. 5. User clicks submit button. |

|  |
| --- |
| Name: acceptNewUserInvitation |
| Participating actor: either an admin or a labOP |
| Entry condition:   * User is logged in the system * User is in their account home page. |
| Exit condition: User Information is updated |
| Event flow:   1. User click on the account details menu 2. User click where password is displayed on the account details panel 3. A pop up menu will show for user to modify its password. 4. User enters the old password, new password and then again the new password to confirm a mistake was not made. 5. User clicks submit button. |

## 5.2 Appendix B - Use cases being implemented (from the RD).

## 5.3 Appendix C – Documented class interfaces (code) for the subsystem(s) you will implement and the constraints.

## 5.4 Appendix D - Diary of meeting and tasks.

# References