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

|  |
| --- |
|  |

# Copyrights and Trademarks Notices

Copyright © by Florida International University – Senior Project – DDD Team

All rights reserved. No part of the Designer Drug Database System Project or its documentation may be reproduced or transmitted in any form or by any means without prior written consent of the DDD Team.

# Executive Summary

For the fall 2015 semester I undertook the project of creating a web app for the Forensic Research Institute at FIU. The International Forensic Institute needed a tool to:

* Share information with the world of Academia and the Professionals Community.
* Manage shared information.
* Control the access to the management of this information

All the required features were implemented in this version. In addition to the features I implemented, this document will also look at our project plan which assisted us in keeping the project on track throughout the semester. Requirements for the new features are also described in addition to a detailed description of the system design. Information such as the use cases that were implemented and system diagrams will also be provided to assist the understanding of how the system was developed.

The appendix contains miscellaneous information such as use cases, diagrams, and models for the system. In addition a diary has been included .

# Table of Contents

[Copyrights and Trademarks Notices 1](#_Toc418135498)

[Executive Summary 2](#_Toc418135499)

[Table of Contents 3](#_Toc418135500)

[1. Introduction 5](#_Toc418135501)

[1.1 Problem definition. 5](#_Toc418135502)

[1.2 Scope of system. 5](#_Toc418135503)

[1.3 Definitions, acronyms, and abbreviations (at most one page). 5](#_Toc418135504)

[1.3.1 Definitions 5](#_Toc418135505)

[1.3.2 Acronyms and abbreviations 6](#_Toc418135506)

[1.4 Overview of document 6](#_Toc418135507)

[2. Feasibility Study 7](#_Toc418135508)

[2.1 Description of current system. Identify limitations and constraints 7](#_Toc418135509)

[2.2 Description of alternative solutions considered. 7](#_Toc418135510)

[2.4.1 Description of Alternatives 7](#_Toc418135511)

[2.3 Recommendation with explanation of why the solution was selected. 8](#_Toc418135512)

[3. Project Plan 9](#_Toc418135513)

[3.1 Project Organization 9](#_Toc418135514)

[3.1.1 Project Personnel Organization 9](#_Toc418135515)

[3.1.2 Hardware and Software Resources 9](#_Toc418135516)

[3.2 Identification of Tasks, Milestones and Deliverables (work breakdown) 10](#_Toc418135517)

[4. System Requirements 12](#_Toc418135518)

[4.1 Functional Requirements – describes high-level functionality 12](#_Toc418135519)

[4.2 Analysis of System Requirements 16](#_Toc418135520)

[4.2.1 Scenarios 16](#_Toc418135521)

[4.2.2 Use case model 16](#_Toc418135522)

[4.2.3 Static model e.g., object diagrams, class diagram 16](#_Toc418135523)

[4.2.4 Dynamic model e.g., sequence diagrams or state machines 17](#_Toc418135524)

[5. System Design (i.e., overall system design) 18](#_Toc418135525)

[2.1 Overview 18](#_Toc418135526)

[2.2 Subsystem Decomposition 18](#_Toc418135527)

[2.3 Hardware and Software Mapping. 19](#_Toc418135528)

[2.4 Persistent Data Management. 19](#_Toc418135529)

[2.5 Security/Privacy 20](#_Toc418135530)

[6. Detailed Design 21](#_Toc418135531)

[6.1 Overview 21](#_Toc418135532)

[6.2 Static model 21](#_Toc418135533)

[6.3 Dynamic model 22](#_Toc418135534)

[6.4 Code Specification 22](#_Toc418135535)

[7. System Validation 24](#_Toc418135536)

[7.1 Subsystem Tests 24](#_Toc418135537)

[7.2 System Tests 24](#_Toc418135538)

[7.3 Evaluation of Tests 25](#_Toc418135539)

[8. Glossary - define terms used in document, especially domain specific terms. 26](#_Toc418135540)

[8.3.1 Definitions 26](#_Toc418135541)

[8.3.2 Acronyms and abbreviations 26](#_Toc418135542)

[9. Appendix 27](#_Toc418135543)

[Appendix A - Project schedule (Gantt chart or PERT chart). 27](#_Toc418135544)

[Appendix B – All use cases with nonfunctional requirements. 28](#_Toc418135545)

[Appendix C – User Interface designs. 37](#_Toc418135546)

[Appendix D – Analysis models (static and dynamic) 45](#_Toc418135547)

[Appendix E – Design models 47](#_Toc418135548)

[Appendix F – Documented Class interfaces. 67](#_Toc418135549)

[Appendix G – Documented code for test drivers and stubs. 70](#_Toc418135550)

[Appendix H – Diary of meeting and tasks for the entire semester. 71](#_Toc418135551)

[Sprint 1 71](#_Toc418135552)

[Sprint 2 72](#_Toc418135553)

[Sprint 3 72](#_Toc418135554)

[Sprint 4 73](#_Toc418135555)

[Sprint 5 73](#_Toc418135556)

[10. References 74](#_Toc418135557)

# Introduction

The Designer Drug Database System is a system designed to easily add new compounds and manage the information through a centralized user friendly web application. This application was done with a responsive web design to be friendly in phones, tablets and any other device that you access the web application from.

This section will be divided in the following sections:

* 1. Problem Definition.
  2. Scope of system.
  3. Terminology.
  4. Overview of document.

## Problem definition.

1. Dr. Arroyo and the team at the Forensic Research Institute produce several reports and useful information from research they perform at the Institute. One of the challenges they have face is sharing the results that yields from their research. The purpose of this project will be to provide the Institute with a software solution that allows them to share the research result with the world of Academia and the Professionals Community

## 1.2 Scope of system.

The Forensic Research institute at FIU is a prestigious research facility that among all the things they research and do, they are developing a drug database from where any compound listed in it can be easily identified. They have wanted to have a medium to share this information with the rest of Academia and professional community.

## 1.3 Definitions, acronyms, and abbreviations (at most one page).

Following we will have the definitions. Acronyms and abbreviations that are used in this document.

### 1.3.1 Definitions

**Single-page application (SPA):** is a web application or web site that fits on a single web page with the goal of providing a more fluid user experience akin to a desktop application.

**Platform independent framework:** is a framework that allows programmers to create one application that can be seen over different platforms.

**Native system:** is a system that can only be seen in its own target devices, such as apple devices or android devices.

**Administrator:** Registered user with credentials. User has elevated privileges and can give access to other users.

**User:** Any general person who uses the system

### 1.3.2 Acronyms and abbreviations

**Admin:** Administrator

**CSS :** Cascading CIS Style Sheets

**DB:** Database

**FIU:** Florida International University

**GUI:** Graphical User Interface

**HTML:** Hypertext Markup Language

**JS:** JavaScript

**MVC:** Model View Controller

**PHP:** PHP: Hypertext Preprocessor

## 1.4 Overview of document

Throughout this document information about the System in general. The following chapters, from two to eleven are a compilation of all other documents .

# Feasibility Study

This section will go in depth on the feasibility study that is to be done to create the new software solution to the introduced problem. In this case, because the system starts from scratch I have no current system to describe, thus will only be taking about the future system that will be implemented.

This section divides as follows:

* 1. Description of Current System (Limitations and Constraints)
  2. Alternative Solutions
  3. Recommendations

## 2.1 Description of current system. Identify limitations and constraints

In this case there is no current system; the system is to be created from the beginning.

## 2.2 Description of alternative solutions considered.

The current options are platform independent framework, hybrid apps crators, and responsive web design application.

### 2.4.1 Description of Alternatives

For platform independent framework there are several different frameworks that can be used. Some of these are:

**Sencha Touch 2**

**Knowledge required:** HTML, CSS, JavaScript, general web development

**Platform support:** iOS, Android, Blackberry

**Cost:** Free [under commercial and open source licenses](http://www.sencha.com/products/touch/license/) (paid OEM license available)

**Documentation:** [Examples, screencasts, and tutorials](http://docs.sencha.com/touch/2-0/#!/guide/getting_started)

**jQuery Mobile Summary**

**Knowledge required:** HTML, CSS, jQuery

**Platform support:** Most mobile browsers

**Cost:** Free (license: [MIT and GPL](http://jquery.org/license/))

**Documentation:** There are [many demos on the site](http://jquerymobile.com/demos/1.1.0/) and [books on jQuery Mobile](http://www.amazon.com/s/ref=nb_sb_noss?url=search-alias%3Daps&field-keywords=jquery+mobile)

The other option is hybrid app creators (HTML/JAVASCRIPT AND HTML/RUBY BASED). The best choices based on support for hybrids are:

[**Appcelerator**](http://www.appcelerator.com/):  This is a solution that allows you to develop native apps with HTML/Javascript (run through a UIWebView on iPhone) . (Free)

[**Phonegap**](http://phonegap.com/): Similar to Appcelerator, I mentioned these two as they seem to have the most vibrant communities, and most extensive support. (Free)

[**Rhomobile**](http://rhomobile.com/) : This is a solution that uses Ruby, especially loved by Ruby on Rails developers. (Free only for noncommercial applications, prices vary)

The last option to create an app like this is to use a responsive web design. Some responsive web design frameworks are:

**Bootstrap:** Sleek, intuitive, and powerful front-end framework for faster and easier web development.   
**Webpage**: getboostrap.com

**Foundation 3:** An advanced responsive front-end framework. Foundation 3 is built with Sass, a powerful CSS preprocessor, which allows us to much more quickly develop Foundation itself — and gives you new tools to quickly customize and build on top of Foundation.  
**WebPage**: foundation.zurb.com

## 2.3 Recommendation with explanation of why the solution was selected.

1. Given that it have been established that a responsive web design is the best choice, now we are going to analyze the different choices. It is important to say that if we compare bootstrap with Foundation 3 we find that bootstrap has a lot more documentation and example on the web. Moreover, there is support because it is in large use nowadays. Looking deeper, If we look at the applications and framework that integrate with the listed responsive web design frameworks, it is important to say that bootstrap integrates very well with another framework call angularjs, which is developed and maintained by google. Thus it is my recommendation to implement the system using the bootstrap responsive web design framework.

# 3. Project Plan

In order to ensure a successful project I have created this initial plan with deadlines which we intend to stick by throughout the semester. This will allow us to see whether or not we are on track with our requirements.

This section divides as follows:

* 1. Project Organization
     1. Project Personnel Organization
     2. Hardware and Software Resources
  2. Identification of Tasks, Milestones and Deliverables (work breakdown)

## 3.1 Project Organization

### 3.1.1 Project Personnel Organization

In this project there is only one team member, me (**Carlos Dominguez)**, and I am the Developer, Tester, System Designer, GUI Designer, Database Manager. On the other hand, the mentor is the product owner that is contacted when question about the product itself are needed.

Moreover, every week on Friday at 1:30pm there is a meeting in person, for the developer to report what was done over the course of that week.

Lastly, every weekday the developer will upload and update the project and its status.

### 3.1.2 Hardware and Software Resources

The resources for this project are:

* GitHub repository, for code development.
* Virtual machine, to host the final product
* My computer, to develop test and upload changes
* PHP, mysql, javascript, angular, bootstrap, and html

## 3.2 Identification of Tasks, Milestones and Deliverables (work breakdown)

The following table easily shoes the work breakdown and each other dependencies.

|  |  |  |
| --- | --- | --- |
|  | Tasks | Task Dependencies |
| **1** | Problem Definition |  |
| **2** | Obtain High Level User Requirements | 1 |
| **3** | Identify Alternatives Solutions | 2 |
| **4** | Determine Solutions | 3 |
| **5** | Requirement Analysis and Elicitation | 1 |
|  | **Milestone: System Analysis and Implementation** |  |
| **6** | Implement Database Structural Changes | 5 |
| **7** | Populate Database with Data | 5, 6 |
| **8** | Modify System Architecture | 5 |
| **9** | Begin UI Design Modification for Forms | 7, 8 |
| **10** | Begin UI Design for Flowchart | 7, 8 |
| **11** | Implement Functions | 7, 9, 10 |
| **12** | Initial Testing | 7, 11 |
|  | **Milestone: Implemented System Functionality** |  |
| **13** | Finalize System Implementation | 12 |
| **14** | Functional Testing | 12 |
| **15** | Evaluate Test Results | 13, 14 |
| **16** | Complete Final Documentation and Presentation | 15 |
|  | **Milestone: System Complete** | 16 |

# System Requirements

This chapter defines the functional requirement to be implemented in version 1 of the Designer Drug Database project. These requirements are described in terms of functional requirements and their respective nonfunctional requirements.

The section will be divided as follows:

* 1. Functional Requirements – describes high-level functionality
  2. Analysis of System Requirements
  3. Scenarios
  4. Static model e.g., object diagrams, class diagram
  5. 4.5 Dynamic model e.g., sequence diagrams or state machines

## 4.1 Functional Requirements – describes high-level functionality

Below are the functional requirements for Designer Drug Database System.

The system shall allow…

**Admins and lab operator must be able to login with their credentials**

* **Usability**: The form to be filled must be easy to follow when filling up.
* **Reliability**: System must work flawlessly 99% of the time.
* **Performance**: Data must be saved with the time according to the size of file being imported.
* **Supportability**: This functionality must work for any browser.

**Admins and lab Operator must be able to add new compounds**

* **Usability**: The form to be filled must be easy to follow when filling up.
* **Reliability**: System must work flawlessly 99% of the time.
* **Performance**: Data must be save within 3s seconds.
* **Supportability**: This functionality must work for any browser.

**Admins and lab Operator must be able to edit compounds**

* **Usability**: The form to be modified must be easy to follow when changing the data.
* **Reliability**: System must work flawlessly 99% of the time.
* **Performance**: Data must be save within 3s seconds.
* **Supportability**: This functionality must work for any browser.

**Admins and lab Operator must be able to delete compounds**

* **Usability**: The compound needs to be deleted from all places
* **Reliability**: System must work flawlessly 99% of the time.
* **Performance**: Data must be save within 3s seconds.
* **Supportability**: This functionality must work for any browser.

**Admins must be able to import compounds**

* **Usability**: The compound needs to be imported from a csv file and jpg pictures.
* **Reliability**: System must work flawlessly 99% of the time.
* **Performance**: Data must be saved with the time according to the size of file being imported.
* **Supportability**: This functionality must work for any browser.

**Admins and Lab Operators must be able to modify their name in the system**

* **Usability**: The form to be filled must be easy to follow when filling up.
* **Reliability**: System must work flawlessly 99% of the time.
* **Performance**: Data must be save within 3s seconds.
* **Supportability**: This functionality must work for any browser.

**Admins and Lab Operators must be able to modify their password in the system**

* **Usability**: The form to be filled must be easy to follow when filling up.
* **Reliability**: System must work flawlessly 99% of the time.
* **Performance**: Data must be save within 3s seconds.
* **Supportability**: This functionality must work for any browser.

**Admins must be able to create new users in the system**

* **Usability**: The form to be filled must be easy to follow when filling up.
* **Reliability**: System must work flawlessly 99% of the time.
* **Performance**: Data must be save within 3s seconds.
* **Supportability**: This functionality must work for any browser.

**Admins must be able to delete users of the system**

* **Usability**: The form to be filled must be easy to follow when filling up.
* **Reliability**: System must work flawlessly 99% of the time.
* **Performance**: Data must be save within 3s seconds.
* **Supportability**: This functionality must work for any browser.

**Admins must be able to modify user roles of the system**

* **Usability**: The form to be filled must be easy to follow when filling up.
* **Reliability**: System must work flawlessly 99% of the time.
* **Performance**: Data must be save within 3s seconds.
* **Supportability**: This functionality must work for any browser.

**Admins and Lab Operators must be able to see summary of the system in a dashboard.**

* **Usability**: The information needs to be clear
* **Reliability**: System must work flawlessly 99% of the time.
* **Performance**: Data must be save within 3s seconds.
* **Supportability**: This functionality must work for any browser.

**Admins and Lab Operators must be able to create new compound classes in the system**

* **Usability**: The form to be filled must be easy to follow when filling up.
* **Reliability**: System must work flawlessly 99% of the time.
* **Performance**: Data must be save within 3s seconds.
* **Supportability**: This functionality must work for any browser.

**Admins and Lab Operators must be able to modify compound classes in the system**

* **Usability**: The form to be filled must be easy to follow when filling up.
* **Reliability**: System must work flawlessly 99% of the time.
* **Performance**: Data must be save within 3s seconds.
* **Supportability**: This functionality must work for any browser.

**Admins and Lab Operators must be able to delete empty compound classes (no compounds inside).**

* **Usability**: The steps must be easy to follow.
* **Reliability**: System must work flawlessly 99% of the time.
* **Performance**: Data must be save within 3s seconds.
* **Supportability**: This functionality must work for any browser.

**Admins and Lab Operators must be able to read contact emails sent to system**

* **Usability**: The form to be filled must be easy to follow when filling up.
* **Reliability**: System must work flawlessly 99% of the time.
* **Performance**: Data must be save within 3s seconds.
* **Supportability**: This functionality must work for any browser.

**Admins and Lab Operators must be able to delete contact emails sent to system**

* **Usability**: The form to be filled must be easy to follow when filling up.
* **Reliability**: System must work flawlessly 99% of the time.
* **Performance**: Data must be save within 3s seconds.
* **Supportability**: This functionality must work for any browser.

**Admins and Lab Operators must be able to reply contact emails sent to system**

* **Usability**: The form to be filled must be easy to follow when filling up.
* **Reliability**: System must work flawlessly 99% of the time.
* **Performance**: Data must be save within 3s seconds.
* **Supportability**: This functionality must work for any browser.

**Any user of the system must be able to download the information of an entire compound class.**

* **Usability**: The steps must be easy.
* **Reliability**: System must work flawlessly 99% of the time.
* **Performance**: Data must be save within 3s seconds.
* **Supportability**: This functionality must work for any browser.

**Any user of the system must be able to search and visualize the information of a compound.**

* **Usability**: The steps must be easy.
* **Reliability**: System must work flawlessly 99% of the time.
* **Performance**: Data must be save within 3s seconds.
* **Supportability**: This functionality must work for any browser.

## 4.2 Analysis of System Requirements

This section contains scenarios for some system requirements as well as the different models related to the requirements such as use case model, static model, and dynamic model.

### 4.2.1 Scenarios

This section contains the scenarios to be developed for this first version of this solution.

#### Managing Compounds

Any administrator or lab operator after having signed in in the system can go to a compound management area where he/she can add, modify or delete existing compounds of the system, as well as any information regarding compounds.

#### Managing Users

Admins after they have logged in they have to be able to create new users, delete users and edit user type, except for their own.

#### Change User Information

Admins and Lab Operators have to be able to change their account information such as their name and password.

#### Manage Contact messages.

Admins and lab operators have to be able to see message of people contacting the application, as well as reply to them and delete them

#### Show the accepted information

Any user of the system, including guest, have to be able to search for a compound and see the information about that compound.

### 4.2.2 Use case model

Appendix B contains the Use Case diagrams with all use cases pertaining to the Designer Drug Database System. This diagram clearly shows the different actors that will be interacting with the system. These are admin, lab Operator, and the guest of the system. Each of these actors is linked with his/her respective capabilities within the system.

### 4.2.3 Static model e.g., object diagrams, class diagram

Appendix C shows some of the different class diagrams for the Designer Drug Database System. These diagrams show the relationship between classes. The class diagrams provided in Appendix C are minimal due to space. On top of each class diagram, the name of diagram can be seen.

### 4.2.4 Dynamic model e.g., sequence diagrams or state machines

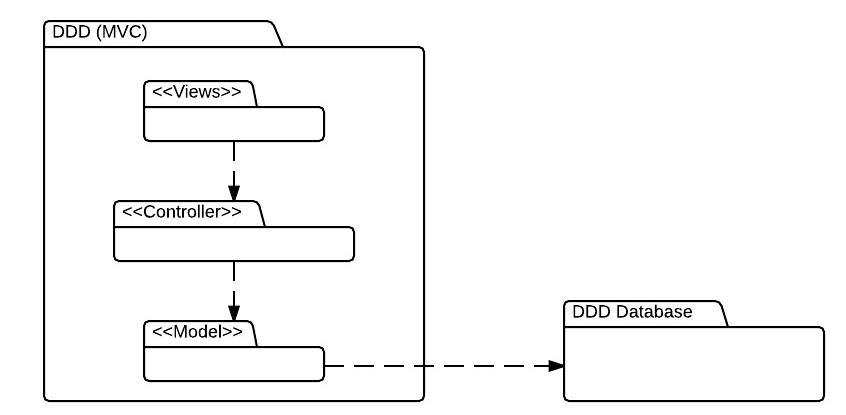
Appendix D shows the different sequence diagrams for the Designer Drug Database System. These diagrams show the interaction among the objects as they are planned to occur when the system is in execution. On top of each sequence diagrams, its use case id is displayed as well as its name.

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

This chapter gives a high level description of the system design for the Designer Drug Database System. It starts by giving and overview of the system by describing the architectures used for this project. Then it explains how the system is decomposed in the different subsystems. Furthermore, it describes how hardware and software are mapped. It also addresses how data is management occurs. Finally, it explains how security and privacy is implemented.

## 2.1 Overview

The Designer Drug Database has been developed based Model-View-Controller (MVC). This architecture separates the representation of data (view), the logic of the system (controller), and the data itself (model).



## 2.2 Subsystem Decomposition

1. **DDD View**

This subsystem has all forms that are presented to the user. Thus the user can only interact directly with this subsystem. This subsystem varies its presentation according to who is interacting with it, none of the three users of the system have the same options. Once the user make a request through this subsystem then the request is passed to the Controller subsystem.

1. **DDD Controller**

This subsystem has the logic of the system. Most of the data that is needed to be either displayed or stored goes through here. When a request is received from the view module, it is analyzed and depending on the request it goes or not to the model.

1. **DDD Model**

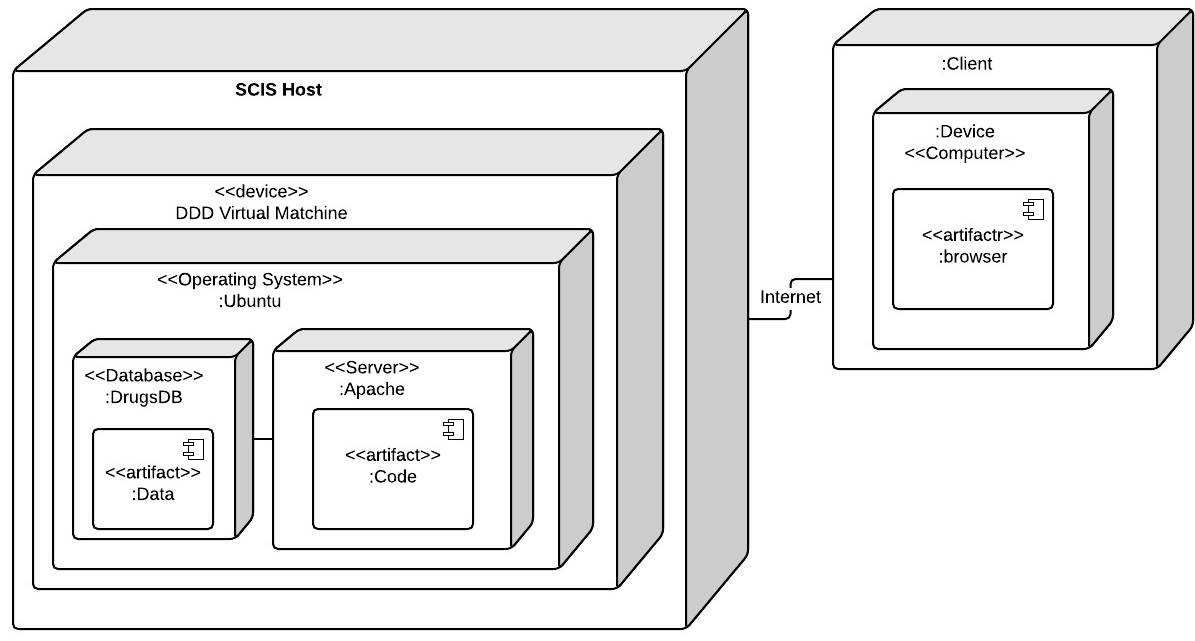
This subsystem provides a way to interact with the data so that it can be stored or retrieved from the repository. It receives request from Controller and performs it by requesting to the database.

1. **DDD Database**

This is the module where all the Information is store

## 2.3 Hardware and Software Mapping.

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.

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

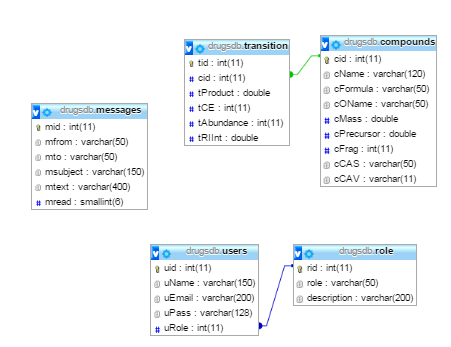
1. **User Information:**

The system needs to save the user information of those who have the right to modify different modules of the system. In this case the system will save the name, email and password of admins and lab operators.

1. **Compound Information**

Compound information is also saved in the database. In this case although the data being stored has changed from sprint to sprint, the current information being stored is Compound name, 1. Other Names, Formula, Mass , Precursor, Product, No. of Transitions, Frag, CE, Abundance, Relative Ion Intensity , CAS, Cayman #

In general the database look like the following



## 2.5 Security/Privacy

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.

Moreover, PDO with prepare statements have been use to prevent SQL Injections.

# 6. Detailed Design

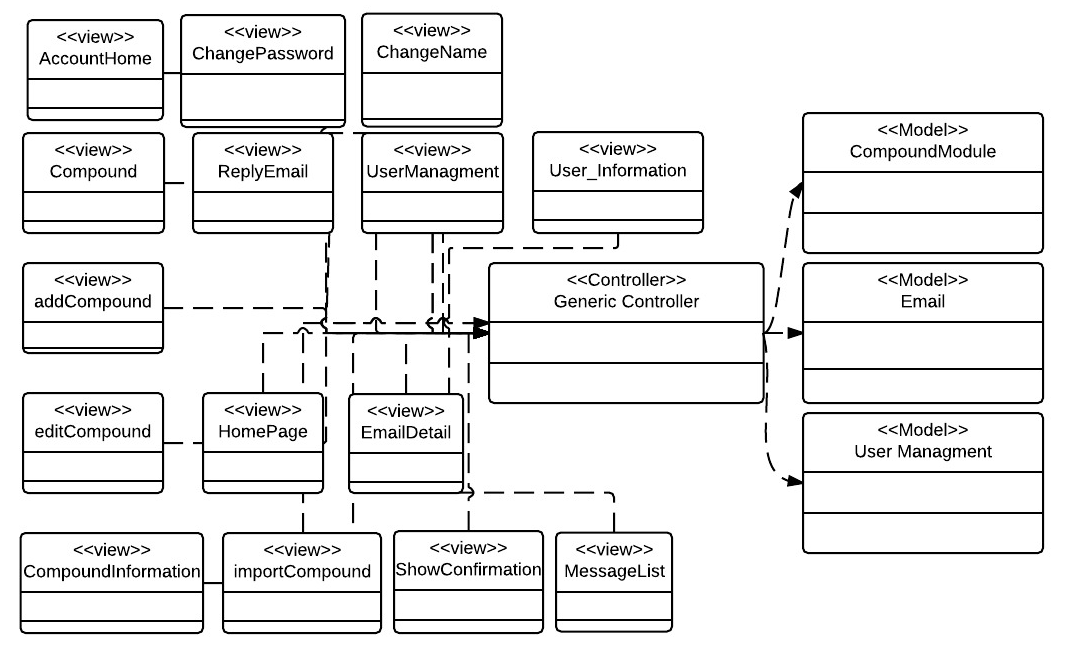
The Detail Design chapter gives a better understanding of the static and dynamic models for the Designer Drug Database System. The document describes the behavior and structure of each subsystem, the different classes along with their methods, and attributes that are part of the system. In addition a minimal class diagrams is provided to help understanding the relationship between classes, and the dynamic model of the system. Here, we can find the sequence diagrams for the use cases that are being implemented for the current version of the project. Main algorithms that have been used are included as well in this section. Finally, the document describes the class interfaces as well as the constraints for the main control object in each system.

## Overview

The Designer drugs database is basically composed of four subsystems, three of them fall inside the DDD MVC, and the other is the repository. This first subsystem is the controller which is in charge of controlling and processing all command within the system. Without this subsystem, the DDD would be just a bunch of forms and death data. The second subsystem is the view, which allows the user to interact with system. This subsystem makes possible that user can see and enter data to the repository. The third subsystem is the model, which allows the system to interact with the data stored in the database. This means to retrieve or send data to be stored. Finally, the last subsystem is the repository, which has the data for the system.

## 6.2 Static model

Here are provided minimal class diagrams. Detailed class diagrams have not been placed here. However, detail class diagrams can be found in the Appendix.



## 6.3 Dynamic model

The Sequence diagram although not included here, can be found in appendix.

## 6.4 Code Specification

There are several things that need to be specified.

1. Client Code and how was it organized
2. Server Code and how is it organized

Client Code:

For the client Code there are several pages that need to be visited:

**/index.php**: contains the main page with the view that is filled by every page

**/functions/function.js**: contains the generic controller that serves to a controller to every view and model. It also contains the route that ANGULARJS uses to dynamic reroute different pages. Also it contains the Controller used by the navigation bar and lastly it contains some global javascript variables

**/pages/….html**: This folder pages contains all the pages or views of the system. The pages that contains a JavaScript function called cntrl which are the portion that belongs to the generic controller for each specific view or page.

Server Code:

For server code you have to look in the folder/request/. This folder contains all php server processing pages (Model). In this case I have:

CompoundModuleClass.php and CompoundModuleCall.php: where all compound information is processed at the model level

UserModuleClass.php, UserModuleCall.php: where you get all user information at the model level

AuthModuleClass.php and AuthModuleCall.php: where all the authentication is done.

MsgModuleClass.php and MsgModuleCall.php: where all messages are processed at model level.

# System Validation

This chapter will talk about the different testing done on the web application.

## Subsystem Tests

Due to lack of time, subsystem testing has not been done.

## System Tests

|  |  |  |
| --- | --- | --- |
| Test Case ID | Purpose | Result |
| Test\_DDD – 0001 | Admins and lab operator must be able to login with their credentials | PASSED |
| Test\_DDD- 0002 | Admins and lab Operator must be able to add new compounds | PASSED |
| Test\_DDD – 0003 | Admins and lab Operator must be able to edit compounds | PASSED |
| Test\_DDD- 0004 | Admins and lab Operator must be able to delete compounds | PASSED |
| Test\_DDD – 0005 | Admins must be able to import compounds | PASSED |
| Test\_DDD – 0006 | Admins and Lab Operators must be able to modify their name. | PASSED |
| Test\_DDD – 0007 | Admins and Lab Operators must be able to modify their password. | PASSED |
| Test\_DDD – 0008 | Admins must be able to create new users in the system | PASSED |
| Test\_DDD – 0009 | Admins must be able to delete users of the system | PASSED |
| Test\_DDD – 00010 | Admins must be able to modify user roles of the system | PASSED |
| Test\_DDD – 00011 | Admins and Lab Operators must be able to see summary of the system in a dashboard. | PASSED |
| Test\_DDD- 00012 | Admins and Lab Operators must be able to read contact emails. | PASSED |
| Test\_DDD – 00013 | Admins and Lab Operators must be able to delete contact emails. | PASSED |
| Test\_DDD – 00014 | Admins and Lab Operators must be able to reply contact emails. | PASSED |
| Test\_DDD – 00015 | Any user of the system must be able to search and visualize the information of a compound. | PASSED |
| Test\_DDD – 00016 | Admins and lab operator must be able to login with their credentials | PASSED |
| Test\_DDD – 00017 | Admins and lab Operator must be able to add new compounds | PASSED |
| Test\_DDD – 00018 | Admins and lab Operator must be able to edit compounds | PASSED |
| Test\_DDD – 00019 | Admins and lab Operator must be able to delete compounds | PASSED |

## 7.3 Evaluation of Tests

The system test done have all passed successfully.

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

### 8.3.1 Definitions

**Single-page application (SPA):** is a web application or web site that fits on a single web page with the goal of providing a more fluid user experience akin to a desktop application.

**Platform independent framework:** is a framework that allows programmers to create one application that can be seen over different platforms.

**Native system:** is a system that can only be seen in its own target devices, such as apple devices or android devices.

**Administrator:** Registered user with credentials. User has elevated privileges and can give access to other users.

**User:** Any general person who uses the system

### 8.3.2 Acronyms and abbreviations

**Admin:** Administrator

**CSS :** Cascading CIS Style Sheets

**DB:** Database

**FIU:** Florida International University

**GUI:** Graphical User Interface

**HTML:** Hypertext Markup Language

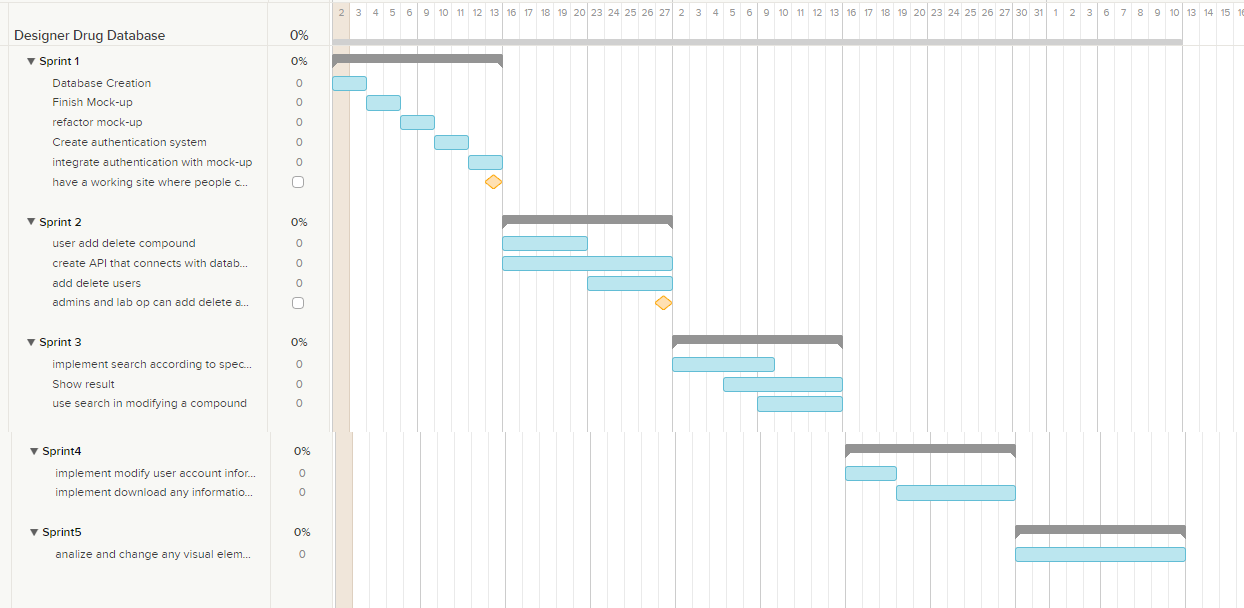
**JS:** JavaScript

**MVC:** Model View Controller

**PHP:** PHP: Hypertext Preprocessor

# Appendix

## Appendix A - Project schedule (Gantt chart or PERT chart).



## Appendix B – All use cases with nonfunctional requirements.

|  |
| --- |
| Name: User Login |
| Participating actor: a non-logged in user |
| Entry condition:   * User is in login page. * User is not logged in. |
| Exit condition:   * User is on the account main page |
| Event flow:   1. User clicks the Login button 2. System loads login page 3. User enters email and password on their respective boxes and click submit button 4. System loads the account main page |
| Alternative Exit condition: System says that login and password are incorrect. |

|  |
| --- |
| Name: Modify Name of User |
| Participating actor: a logged in user. |
| Entry condition:   * User is logged in. * User is in Account Page |
| Exit condition:   * Confirmation name of user is changed is shown. |
| Event flow:   1. User clicks on its name 2. System shows a view to change the name of the user 3. User replaces old name with new name and clicks submit 4. System changes the name of the user and shows a confirmation. |

|  |
| --- |
| Name: Modify user password |
| Participating actor: a logged in user. |
| Entry condition:   * User is logged in. * User is in Account Page |
| Exit condition:   * Confirmation that password has been changed |
| Event flow:   1. User clicks on the password link 2. System shows a view to change the password 3. User enters old, new and confirmation of new password. 4. System changes the password of the user and shows a confirmation. |

|  |
| --- |
| Name: Search For Compound |
| Participating actor: any user of 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 enters the email or formula of compound in the search criteria box and click the search button. 2. Systems show the information of the compound |
| Alternative Exit condition: System does not find a compound and show a message saying the search was unsuccessful. |

|  |
| --- |
| Name: Download Compound Class |
| Participating actor: any user of 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 and clicks on it 2. A window is pop up to ask user where he wants to save the information 3. User pick location where information will be saved and click save 4. System downloads the file |

|  |
| --- |
| Name: add new Compound |
| Participating actor: either an admin or a labOP |
| Entry condition:   * User is logged in the system * User is in compound page. |
| Exit condition: Compound data is added in the system |
| Event flow:   1. User click on the add compound button 2. System shows add compound form page 3. User selects the picture of the compound, fill the fields: name, formula, other name, mass, fragments, retention time, precursor, CAS. User add the corresponding transitions and click submit all button. 4. System confirm that compound has been saved |

|  |
| --- |
| Name: delete Compound |
| Participating actor: either an admin or a labOP |
| Entry condition:   * User is logged in the system * User is in the compound page. |
| Exit condition: Compound data is deleted from the system |
| Event flow:   1. User enters the name or formula in the search box and clicks the search button. 2. System searches for compound and goes to modify compound view 3. User clicks delete compound button. 4. System deletes the compound and shows a confirmation |

|  |
| --- |
| Name: edit Compound |
| 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 |
| Event flow:   1. User enters the name or formula in the search box and clicks the search button. 2. System searches for compound and goes to modify compound view. 3. User edit any of the information on the modify compound view and click the update button. 4. System updates the compound and shows a confirmation |

|  |
| --- |
| Name: import Compound |
| Participating actor: either an admin or a labOP |
| Entry condition:   * User is logged in the system * User is in the compound page. * File to be imported has a csv extension and has 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 and pictures has been added to the system |
| Event flow:   1. User click in the import compound button 2. System shows the import page 3. User clicks the choose file to choose location of csv file from import csv panel 4. A pop up menu will show for user to choose the file he wishes to import. 5. User selects the csv file and clicks submit button. 6. System saves the compounds information and shows a confirmation. 7. User click the choose file to pick the location of the folder where the compound pictures are at, from import picture panel. 8. A pop up menu will show for user to choose the folder where pictures of compounds are 9. User selects all the pictures to be upload and clicks the submit button 10. System upload all the pictures associated with the compounds uploaded previously. |

|  |
| --- |
| Name: Add New User |
| Participating actor: admin user |
| Entry condition:   * User is logged in the system * User is in the user management page. |
| Exit condition: User invite is sent and confirmation is shown |
| Event flow:   1. User enters the email of the user to be invited and clicks the invite button. 2. System sends an email to the user and shows a confirmation. |

|  |
| --- |
| Name: edit User Rights |
| Participating actor: an admin |
| Entry condition:   * User is logged in the system. * User is in the user management page. * User is not trying to changes rights to itself. |
| Exit condition: User access right is updated |
| Event flow:   1. User enters the email of the user to be searched and clicks the search button. 2. System shows a view with the searched user information. 3. User picks the downgrade or upgrade option 4. System updates user rights and show confirmation. |

|  |
| --- |
| Name: delete User |
| Participating actor: an admin |
| Entry condition:   * User is logged in the system * User is in the user Management page. * user is not trying to delete itself |
| Exit condition: User is deleted |
| Event flow:   1. User enters the email of the user to be searched and clicks the search button. 2. System shows a view with the user information. 3. User clicks the delete button 4. System deletes the user and shows confirmation. |

|  |
| --- |
| Name: Add New Compound Class |
| Participating actor: admin or LabOP user |
| Entry condition:   * User is logged in the system * User is in the compound class page. |
| Exit condition: Compound Class is added and confirmation is shown |
| Event flow:   1. User Clicks compound classes button. 2. System shows the Compound Classes view 3. User clicks button add class 4. System shows view to add new class 5. User enters the class name and clicks submit. 6. System saves the new class and shows confirmation. |

|  |
| --- |
| Name: Edit Compound Class Name |
| Participating actor: admin or LabOP user |
| Entry condition:   * User is logged in the system * User is in the compound class page. |
| Exit condition: Compound Class name is updated and confirmation is shown |
| Event flow:   1. User Clicks compound classes button. 2. System shows the Compound Classes view 3. User clicks on the class that is going to be updated 4. System shows view where the class name can be updated 5. User updates the class name and clicks submit. 6. System updates the class name and shows confirmation. |

|  |
| --- |
| Name: delete Compound Class |
| Participating actor: admin or LabOP user |
| Entry condition:   * User is logged in the system * User is in the compound class page. * Compound class to be deleted has no compounds associated with it |
| Exit condition: Compound Class is deleted and confirmation is shown |
| Event flow:   1. User Clicks compound classes button. 2. System shows the Compound Classes view 3. User clicks on the class that is going to be deleted. 4. System shows view with the class name to be deleted 5. User clicks the delete button. 6. System deletes the class and shows confirmation. |

|  |
| --- |
| Name: Add Read contact email |
| Participating actor: admin or LabOP user |
| Entry condition:   * User is logged in the system * User is in the Messages page. |
| Exit condition: Message content is shown to the user |
| Event flow:   1. User Clicks on a message to be read from list of messages 2. System shows the message content. |

|  |
| --- |
| Name: reply contact message |
| Participating actor: admin or LabOP user |
| Entry condition:   * User is logged in the system * User is in the Messages page. |
| Exit condition: Message reply is sent and confirmation is shown. |
| Event flow:   1. User Clicks on a message to be replied from list of messages 2. System shows the message content. 3. User clicks on the reply button 4. System shows a view to reply the message 5. User enters the text to be replied and click reply button 6. System replies the message and show a confirmation |

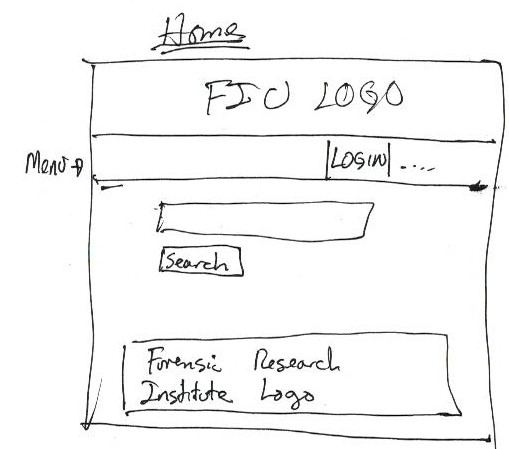
|  |
| --- |
| Name: delete message |
| Participating actor: admin or LabOP user |
| Entry condition:   * User is logged in the system * User is in the Messages page. |
| Exit condition: Message content is shown to the user |
| Event flow:   1. User Clicks on a message to be deleted from list of messages 2. System shows the message content. 3. User clicks the delete button 4. System deletes the message and shows confirmation message. |

## Appendix C – User Interface designs.

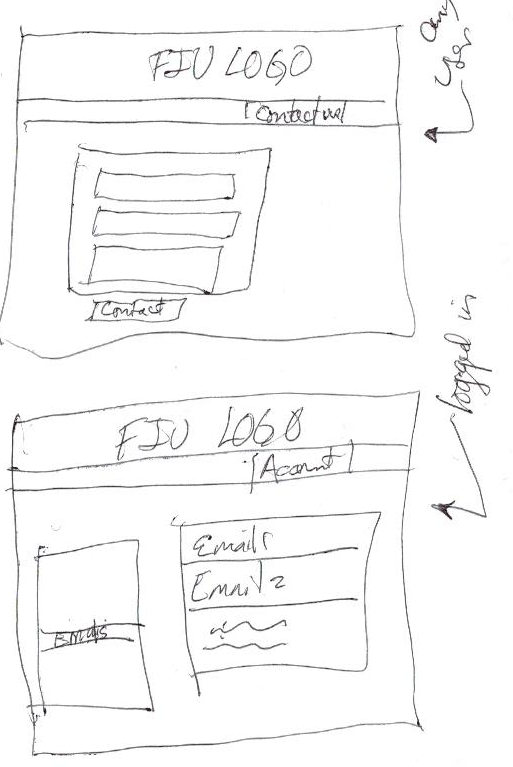
Account Menu



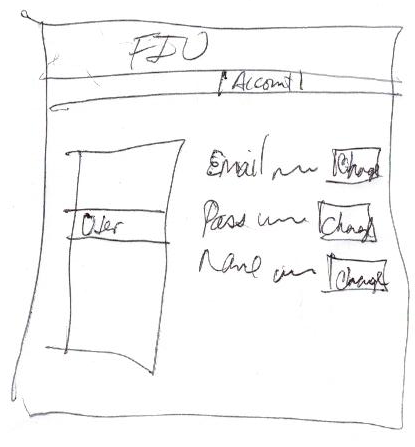
Account Home



Account Contact Us and Read Email



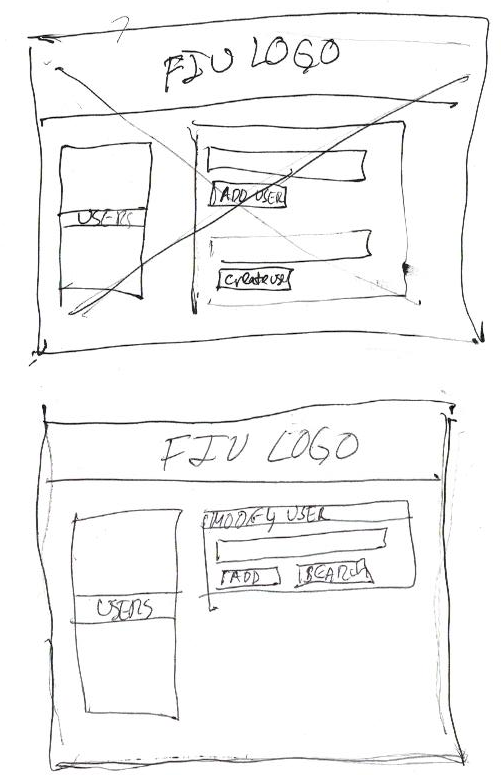
Account User Information Management



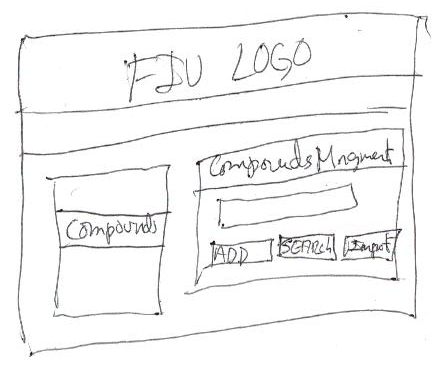
Compound Information



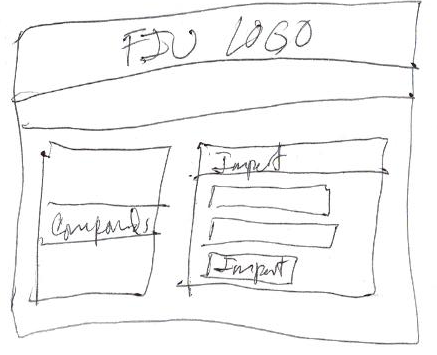
User Managment



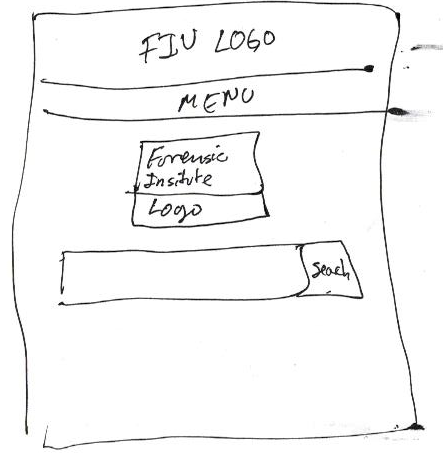
Compound Management

****

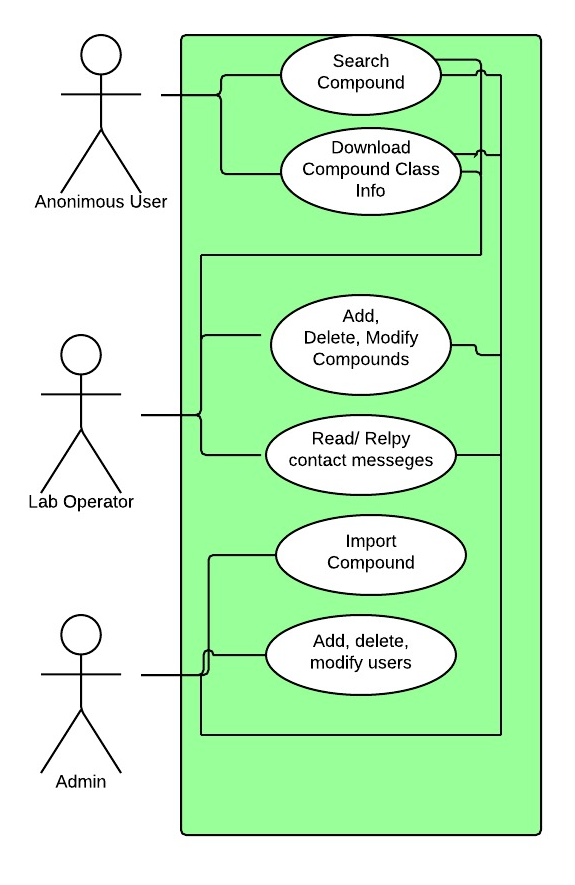
Compound Import Management

****

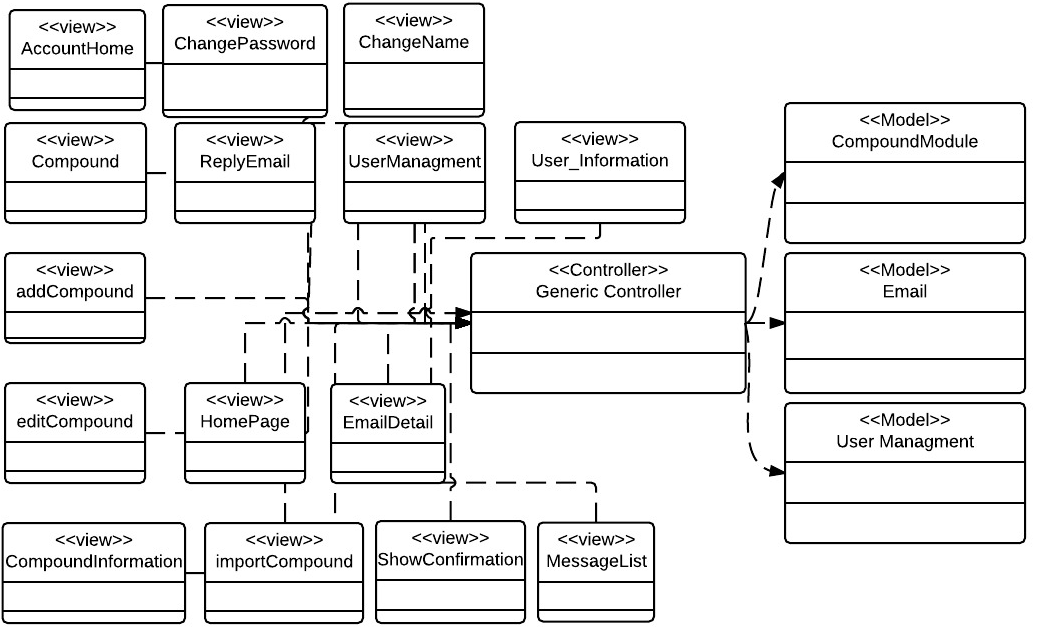
App Home

****

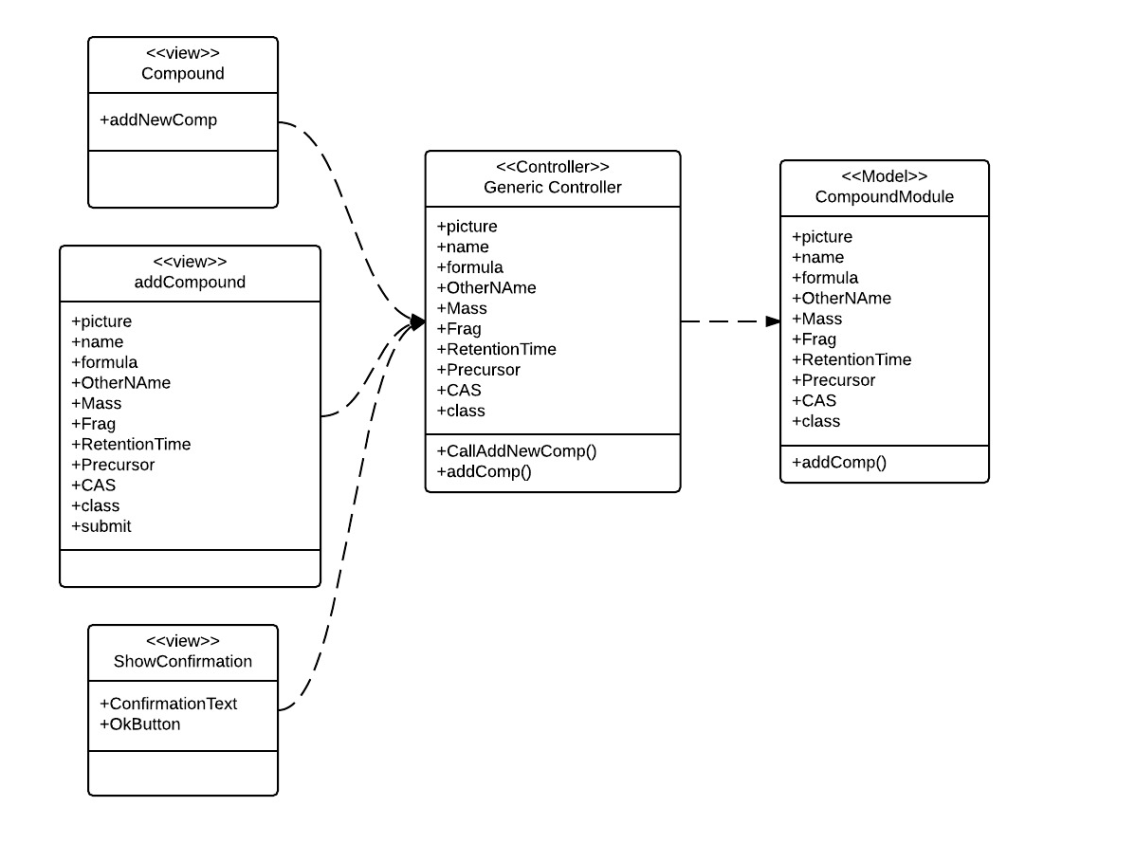
## Appendix D – Analysis models (static and dynamic)



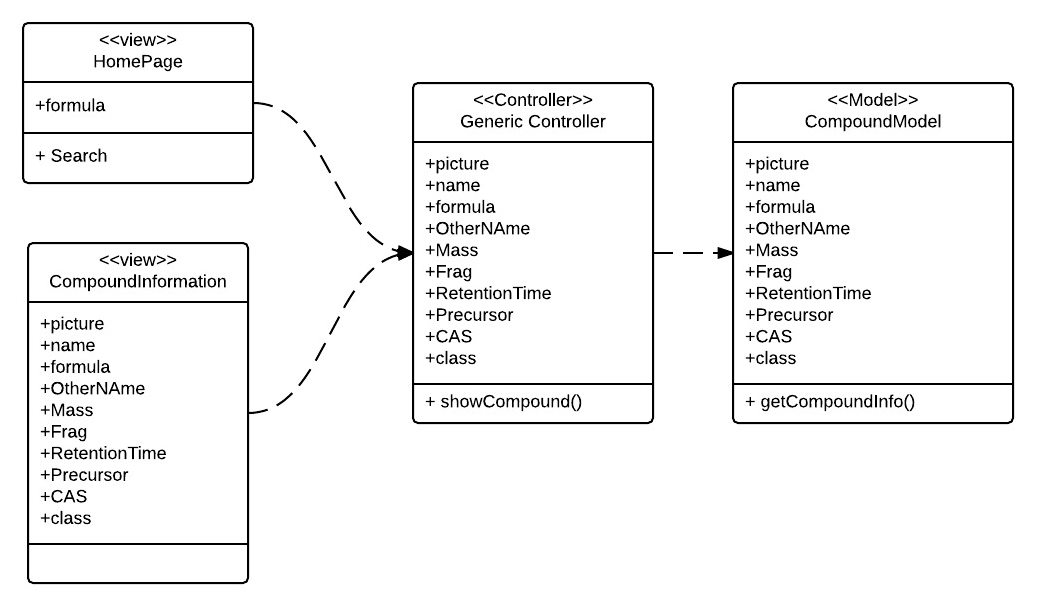
## Appendix E – Design models



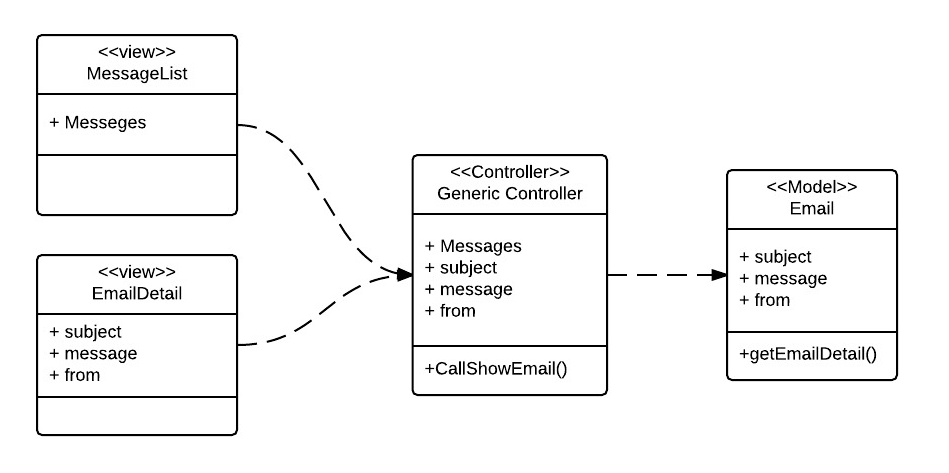
Add Compound Class Diagram



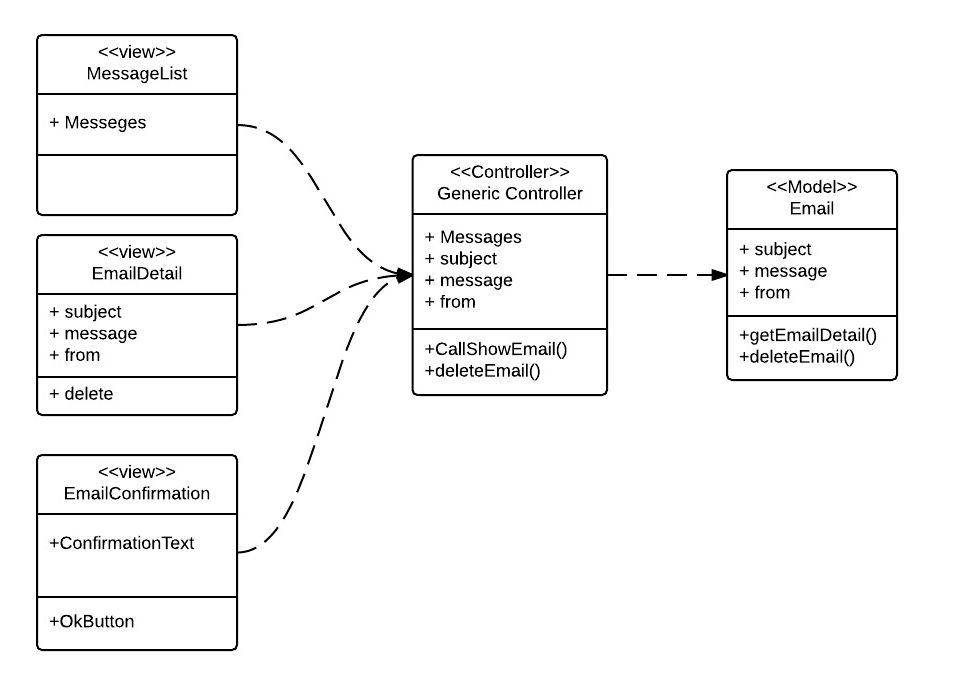
Show Compound Class Diagram



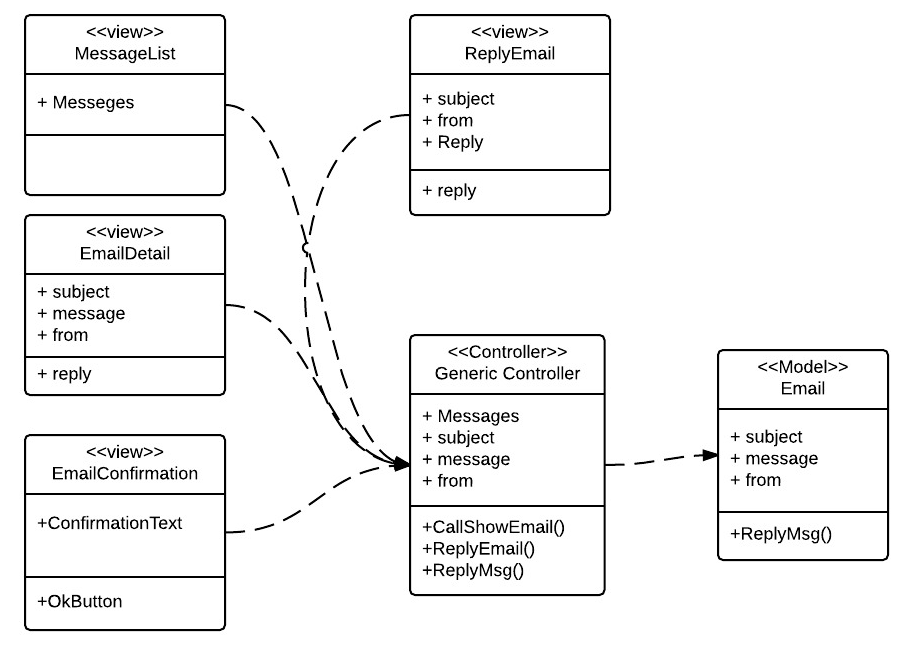
Read Email Class Diagram



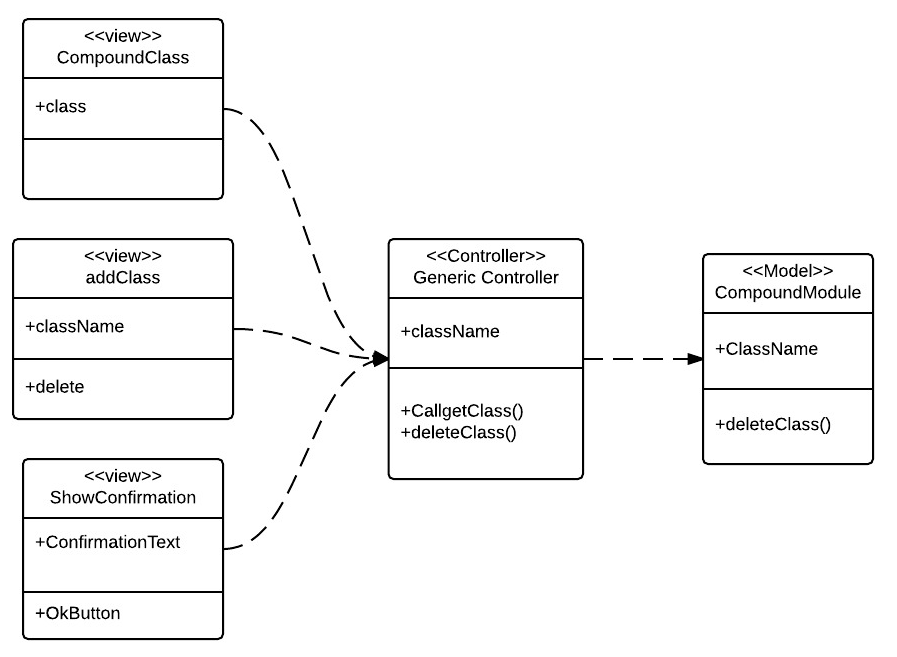
Delete Email Class Diagram



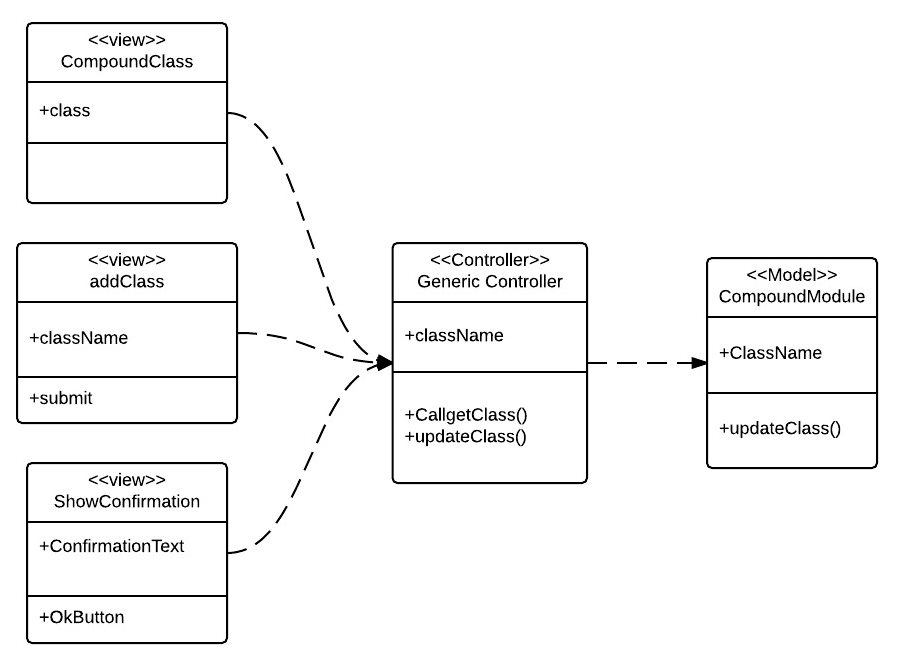
Reply Email Class Diagram



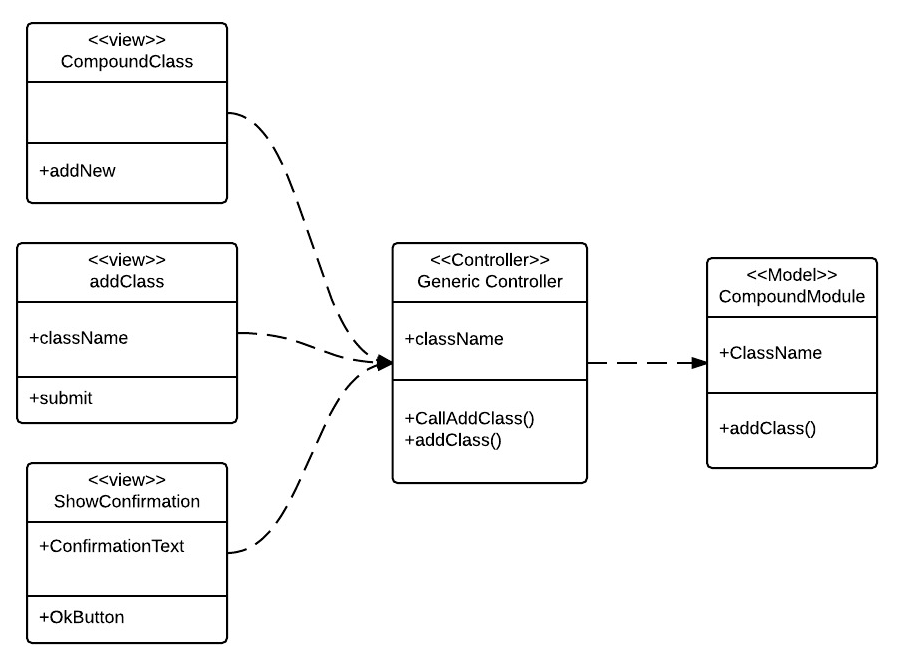
Delete Compound Classes Class Diagram



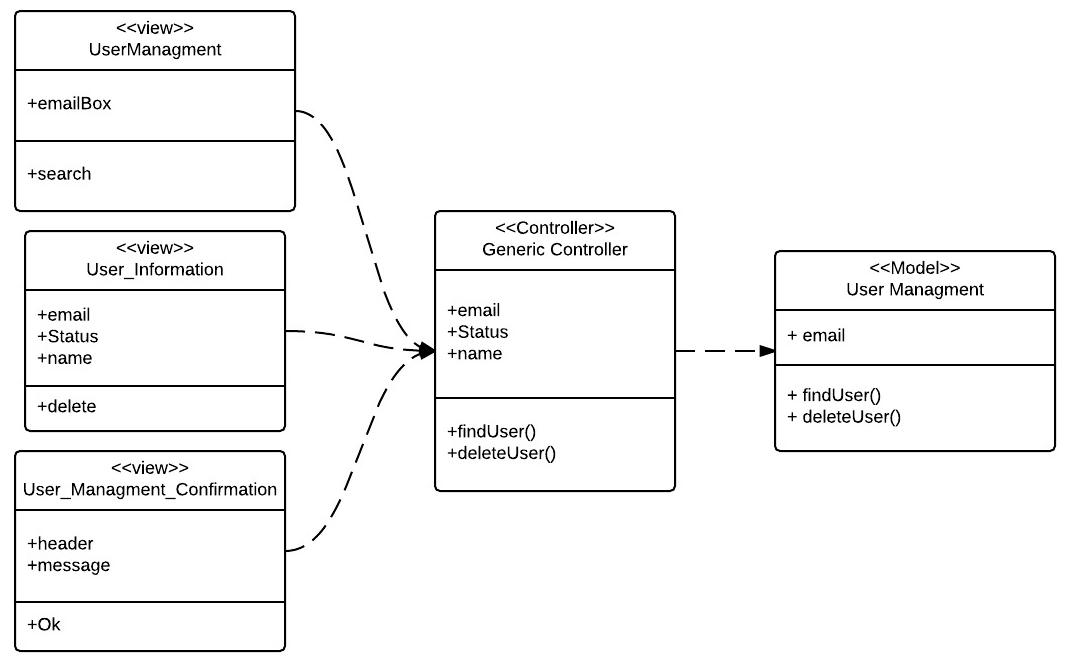
Edit Compound Classes Class Diagram



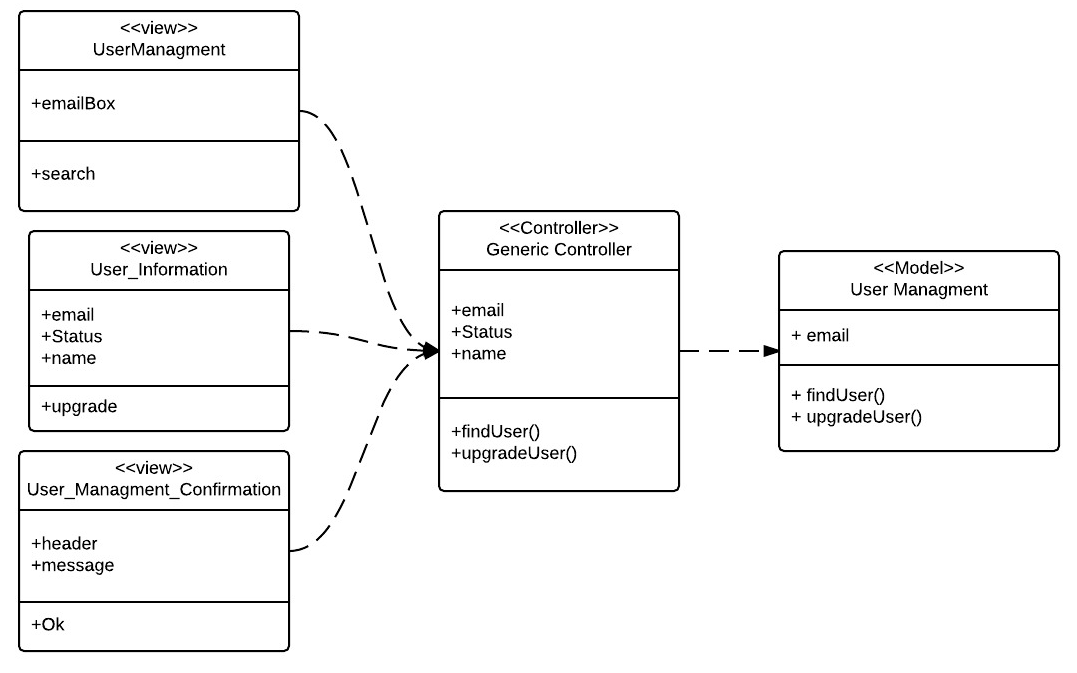
Add Compound Classes Class Diagram



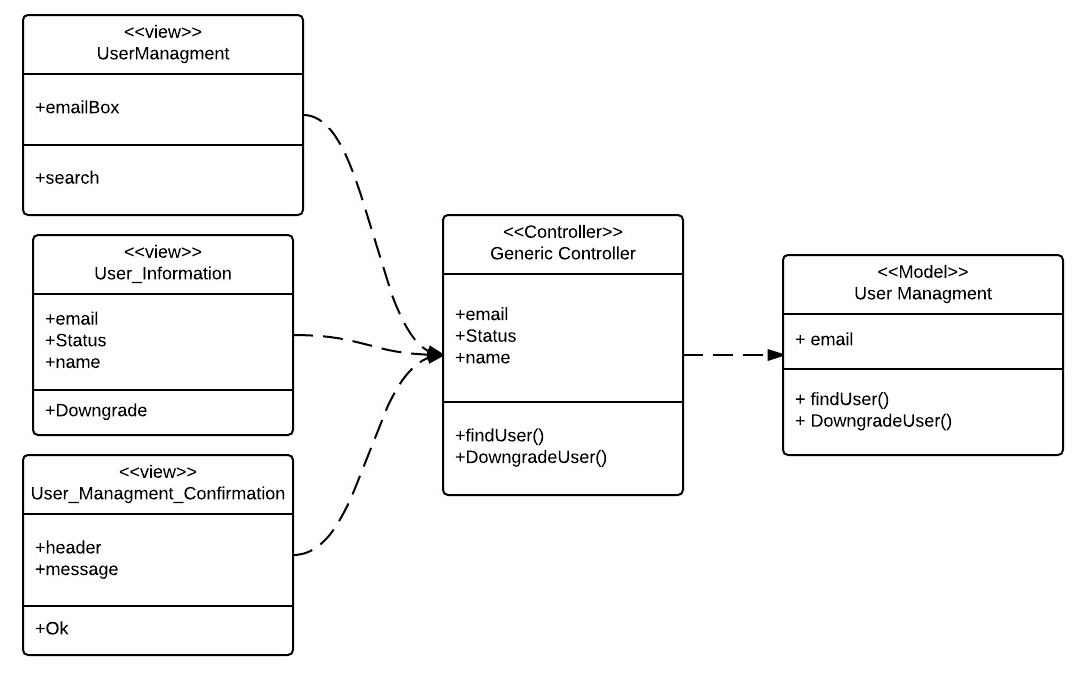
Delete User Class Diagram



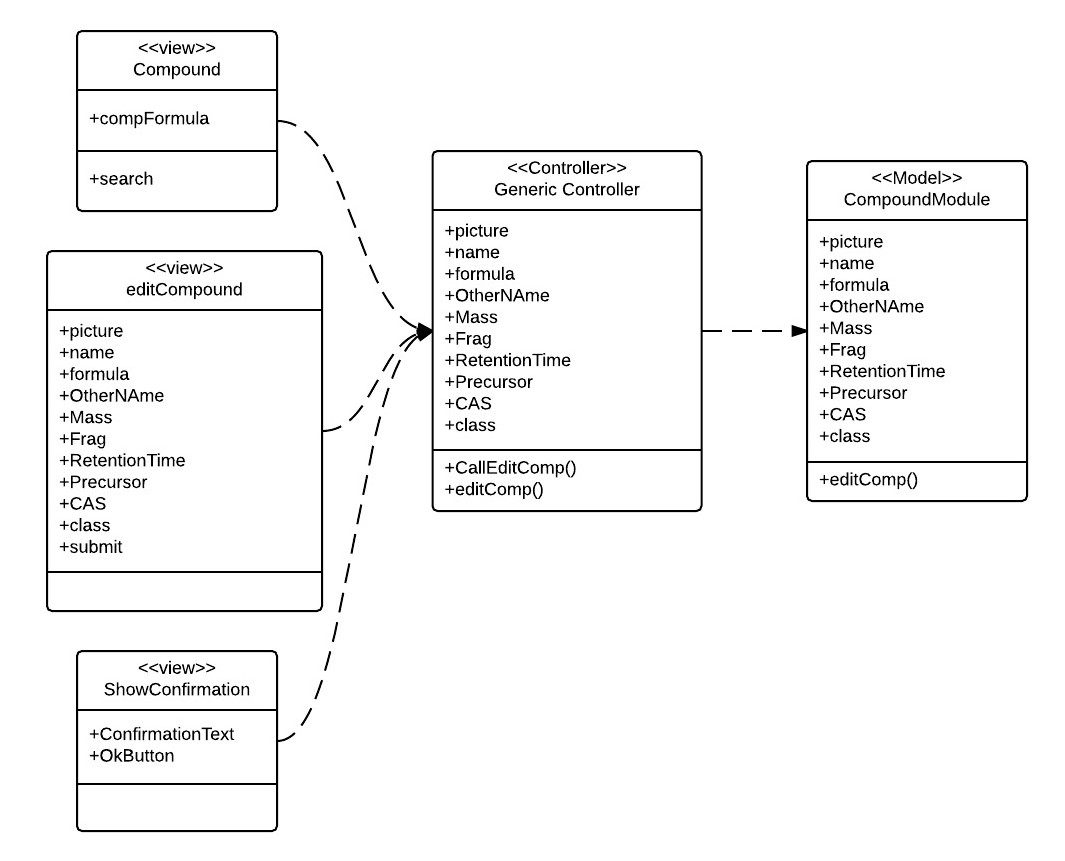
Upgrade User Role Class Diagram



Downgrade User Role Class Diagram



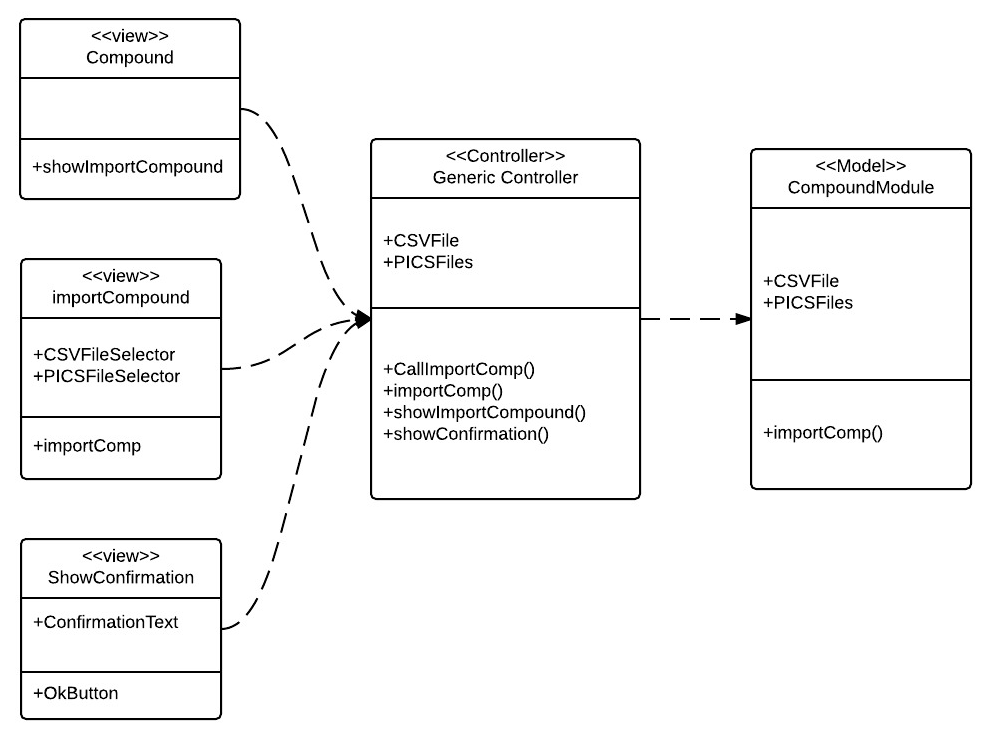
Edit Compound Class Diagram



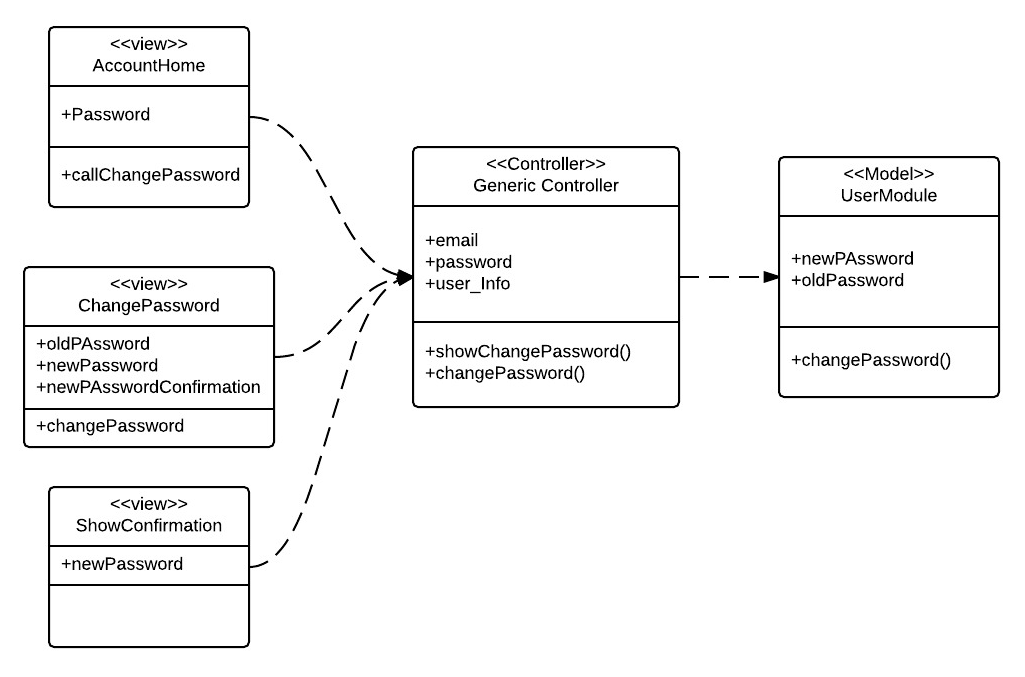
Delete Compound Class Diagram



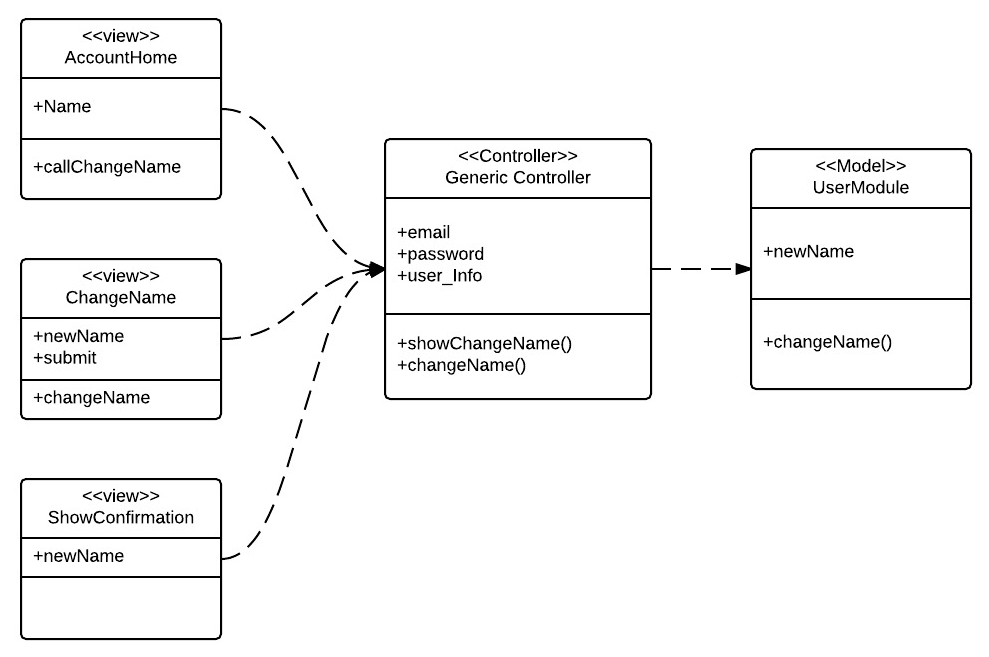
Import Compound Class Diagram



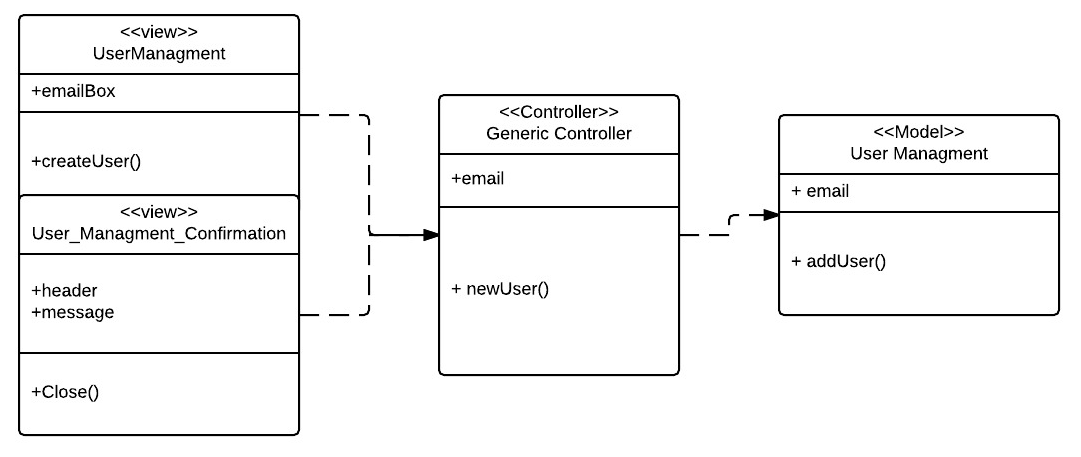
Change User Password Class Diagram



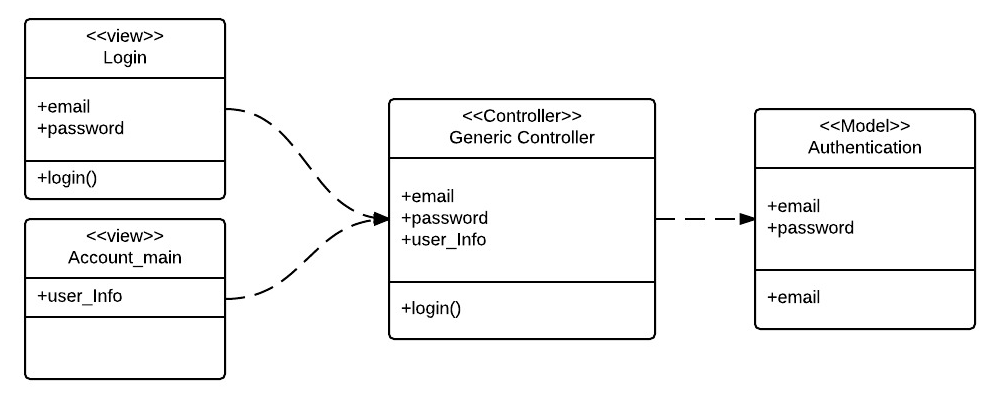
Change User Name Class Diagram



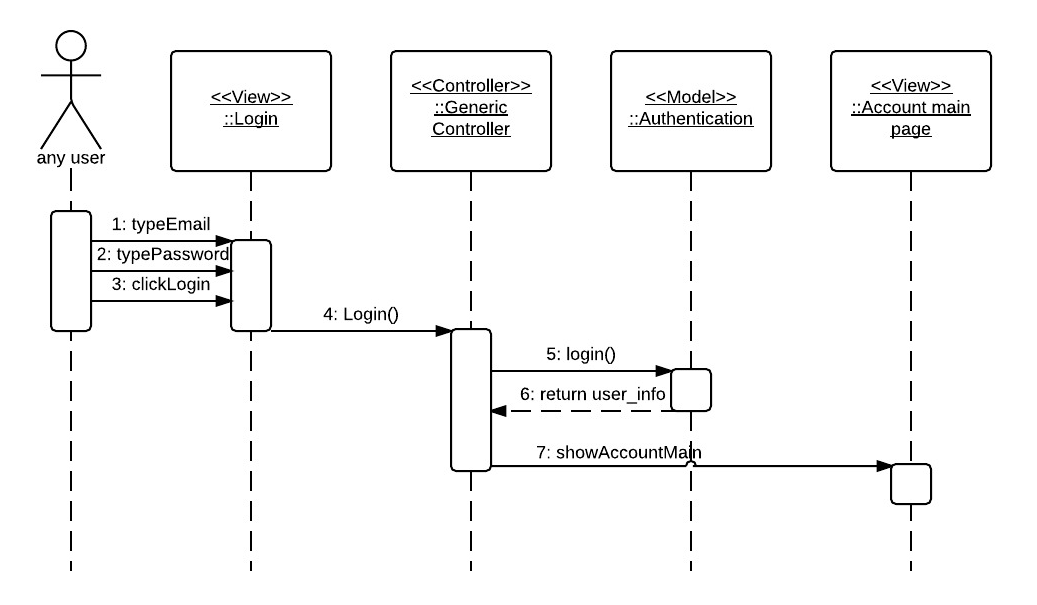
Invite User Class Diagram



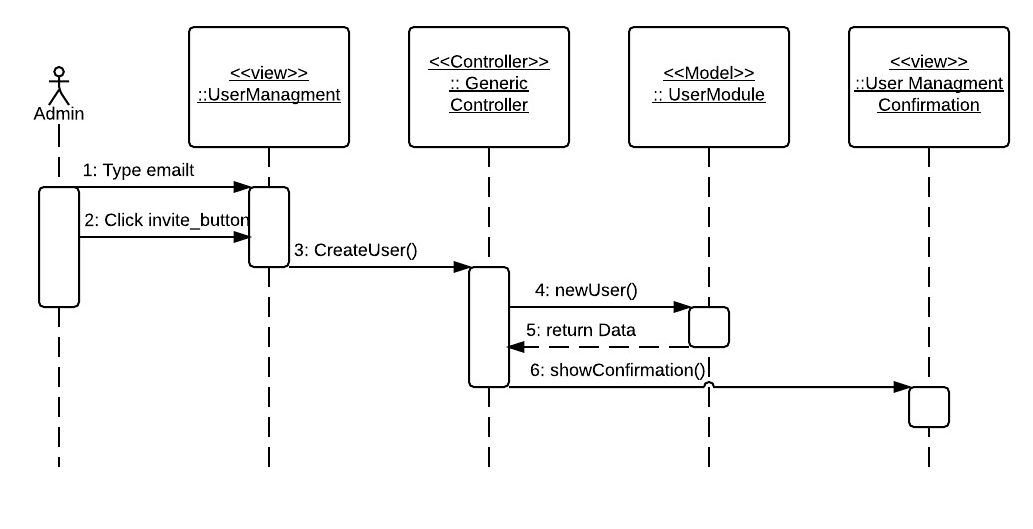
User Login Class Diagram



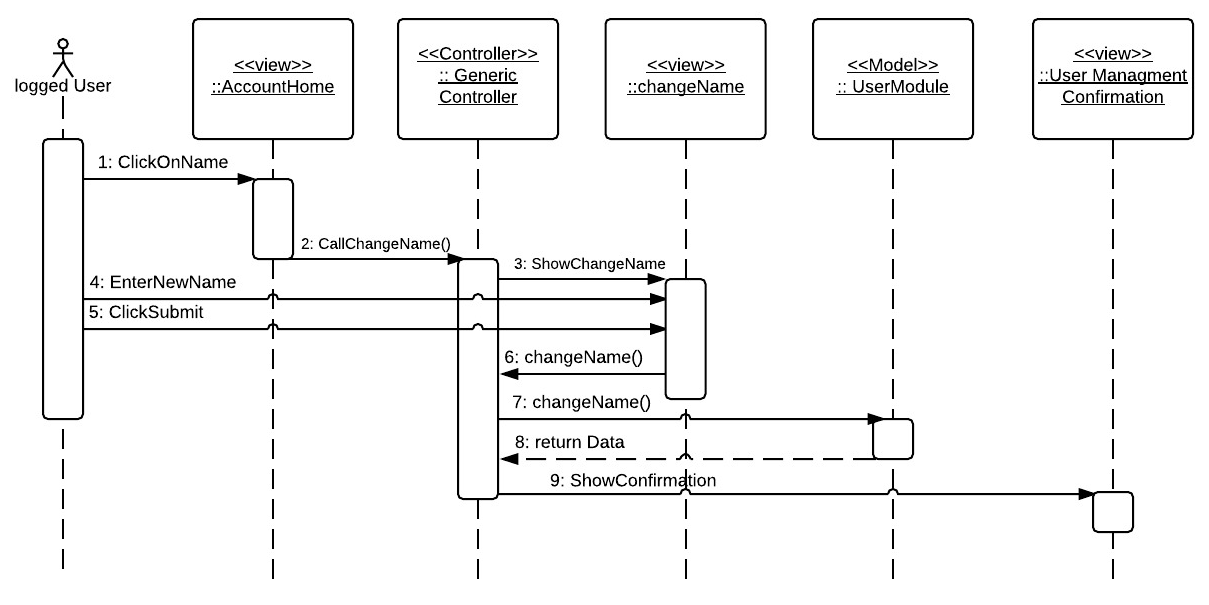
User Login Sequence Diagram



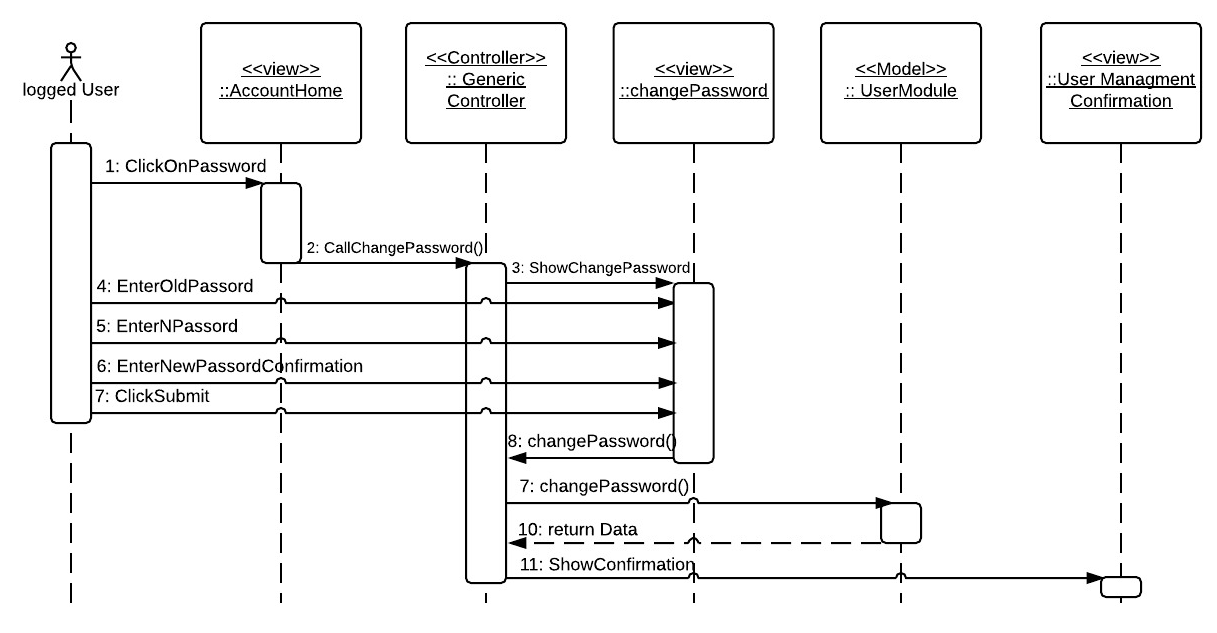
Invite User Sequence Diagram



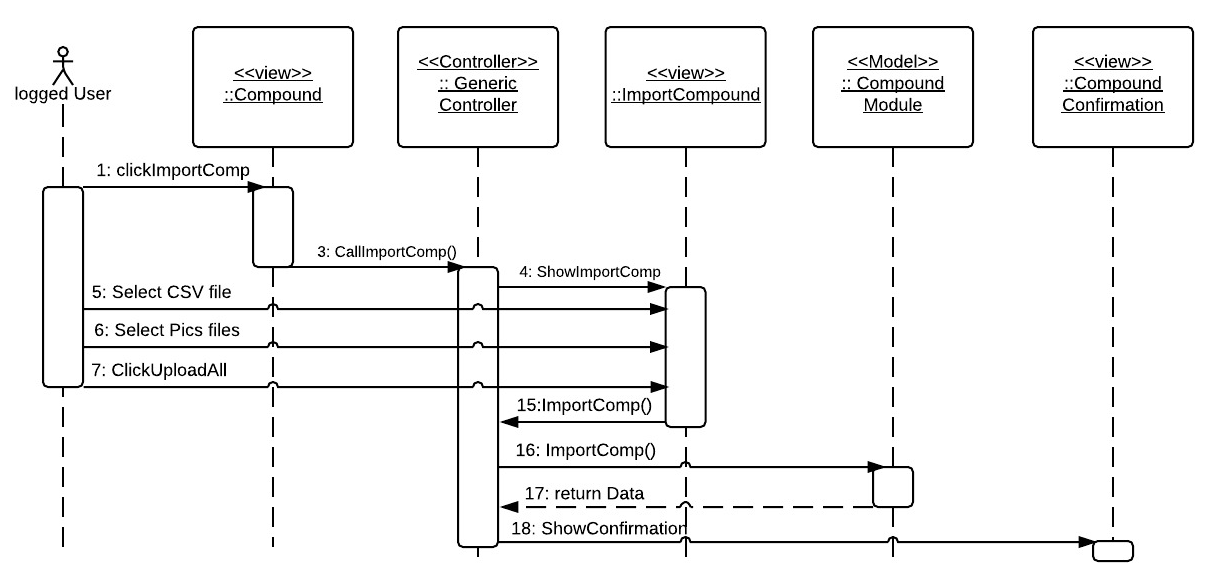
Change User Name Sequence Diagram



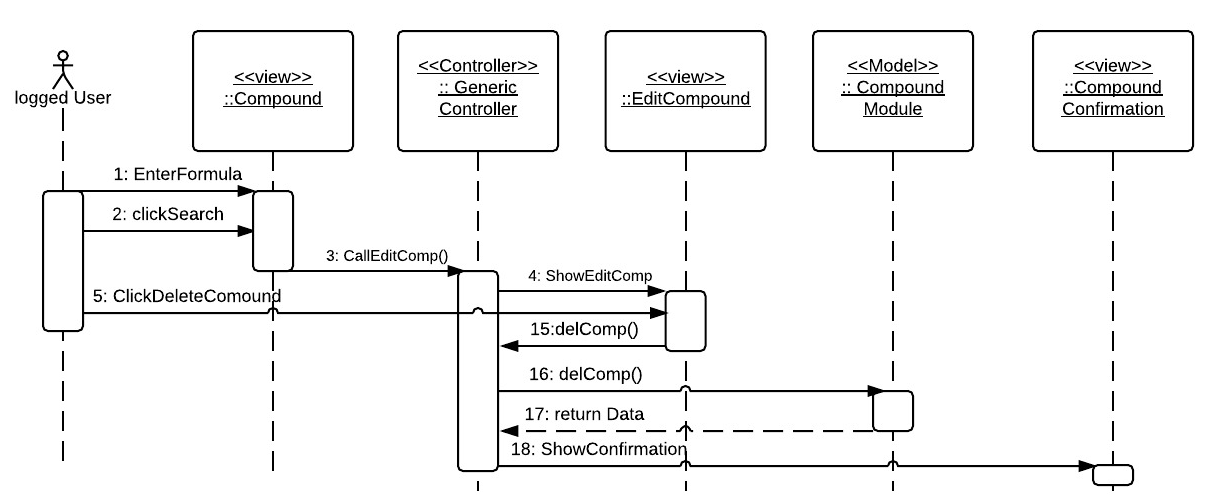
Change User Password Sequence Diagram



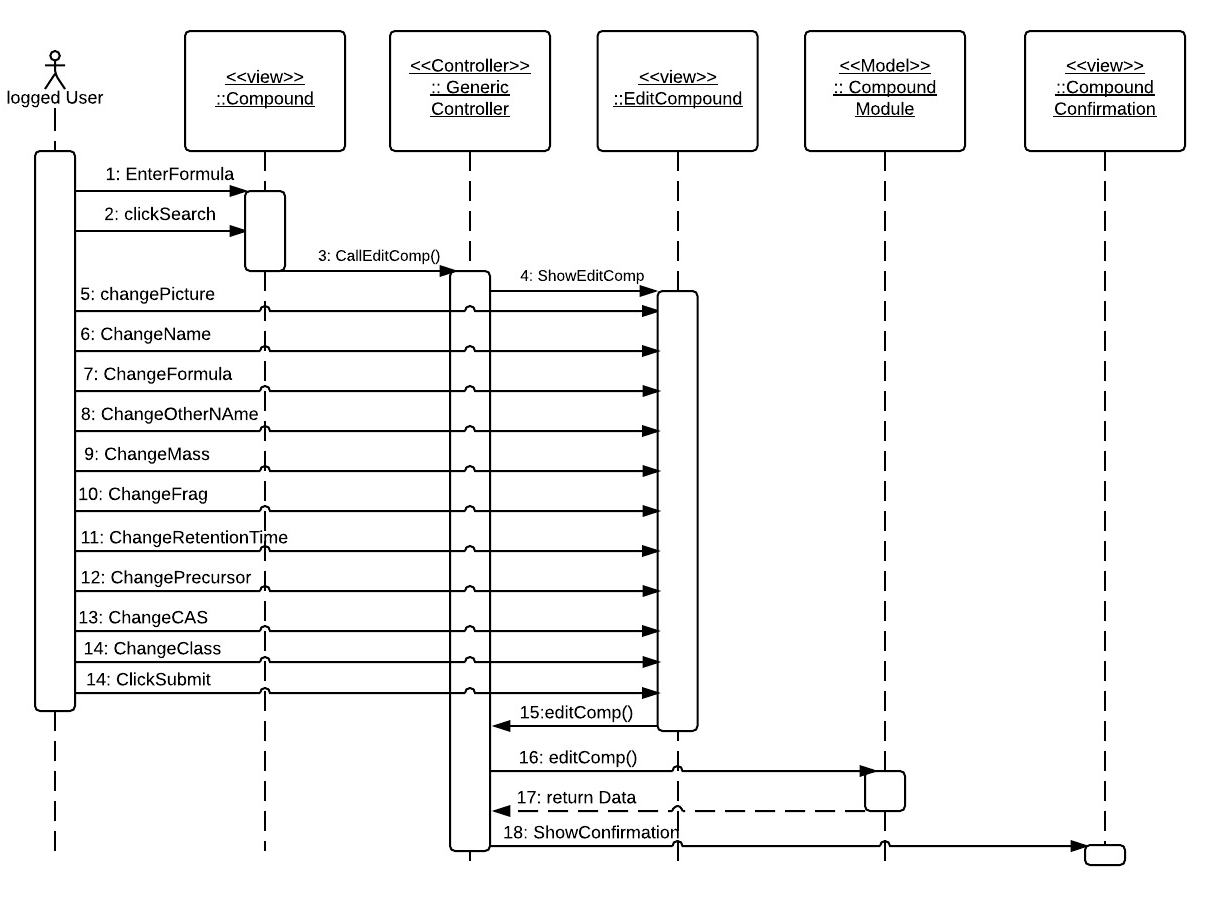
Import Compound Sequence Diagram



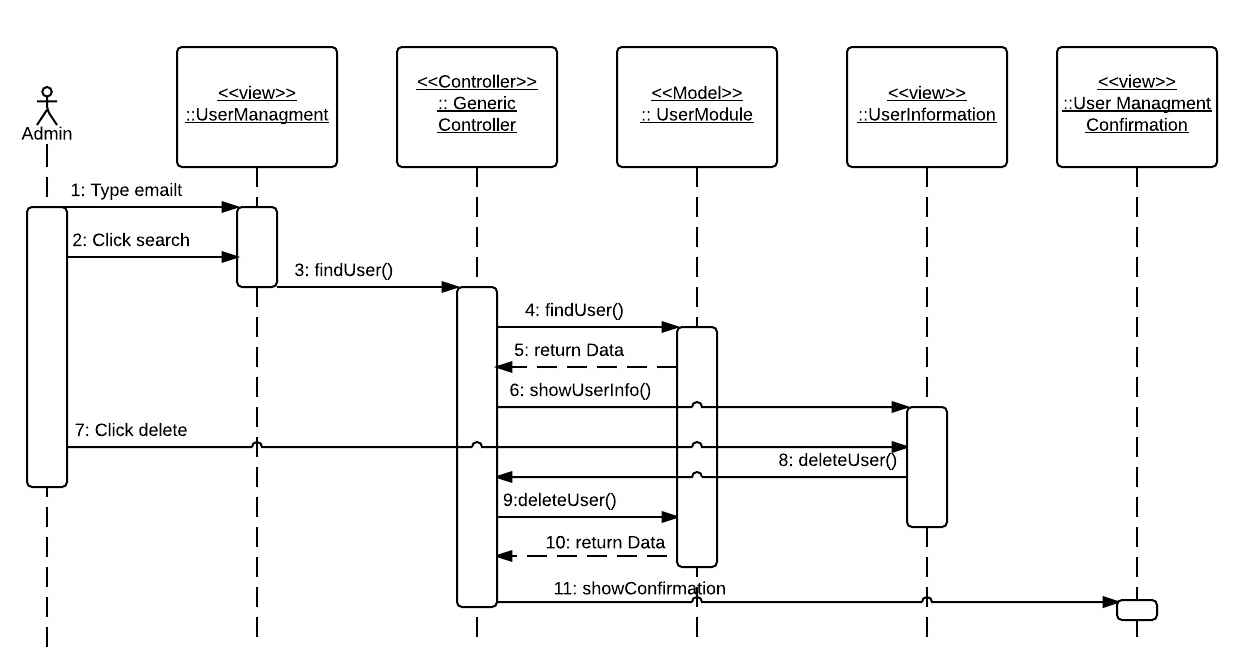
Delete Compound Sequence Diagram



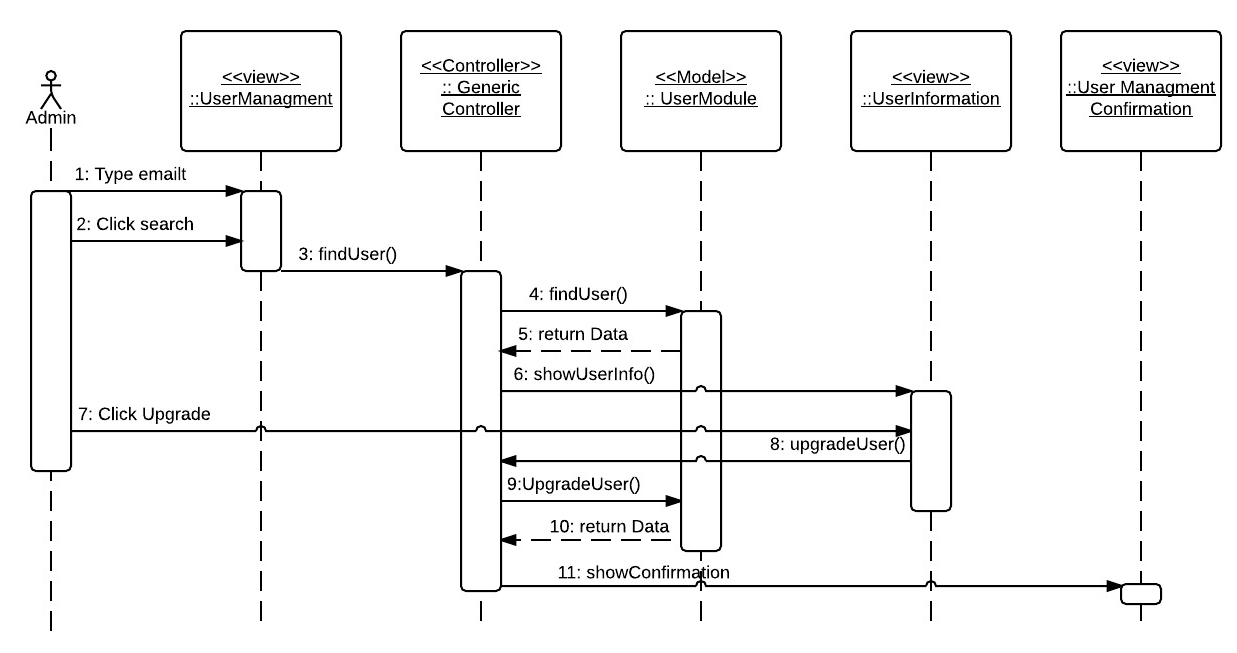
Edit Compound Sequence Diagram



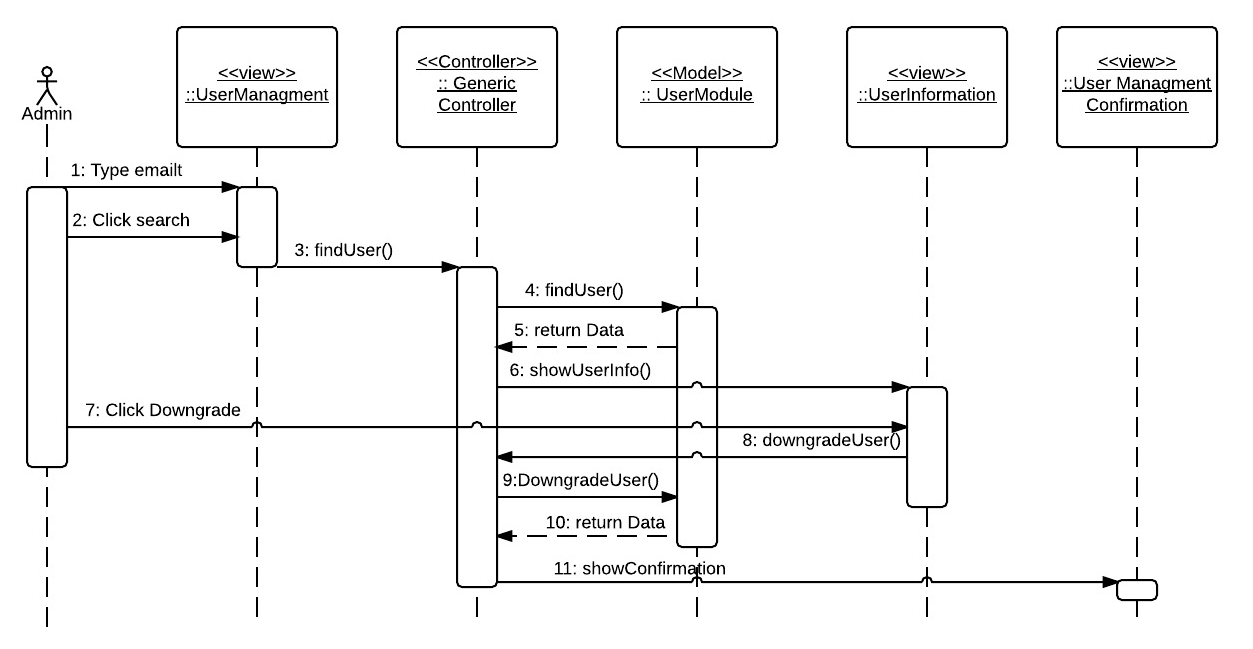
Delete User Sequence Diagram



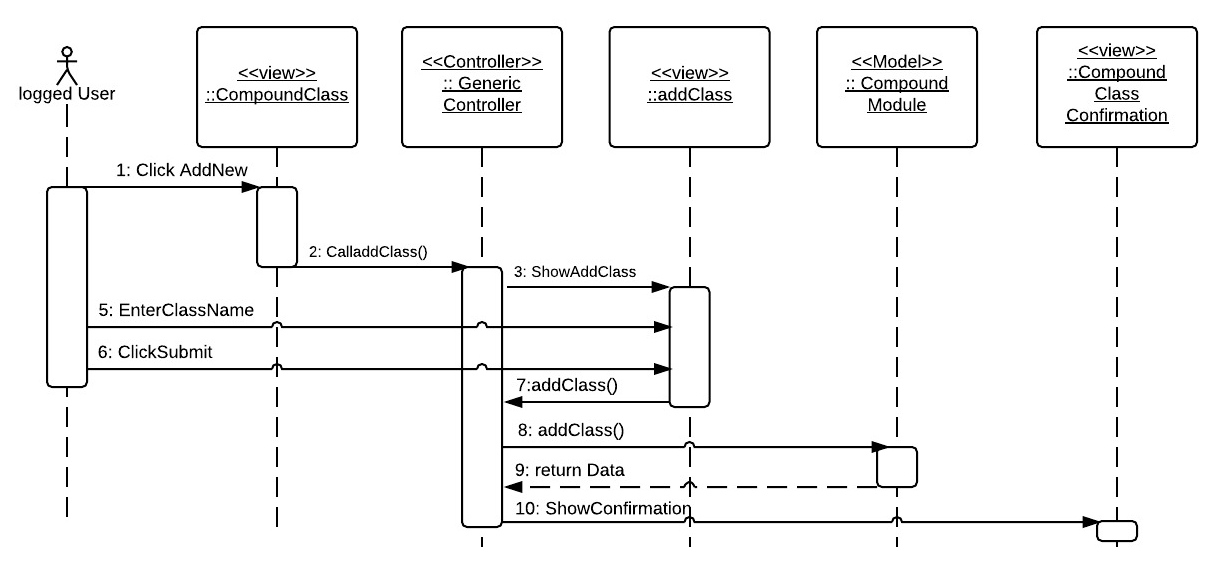
Upgrade User Role Sequence Diagram



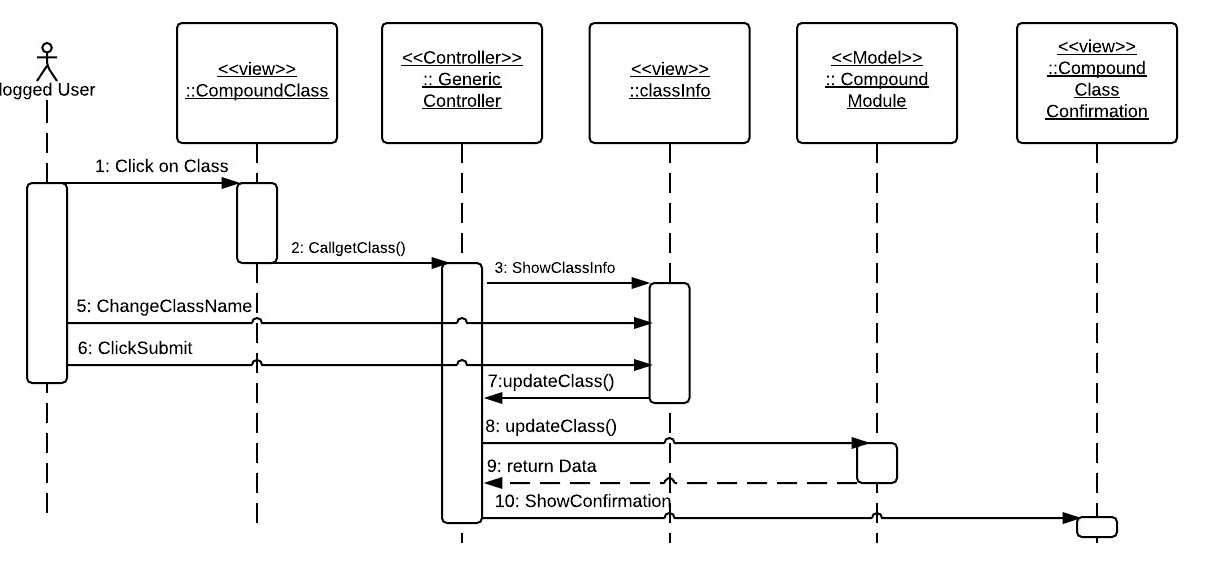
Downgrade User Role Sequence Diagram



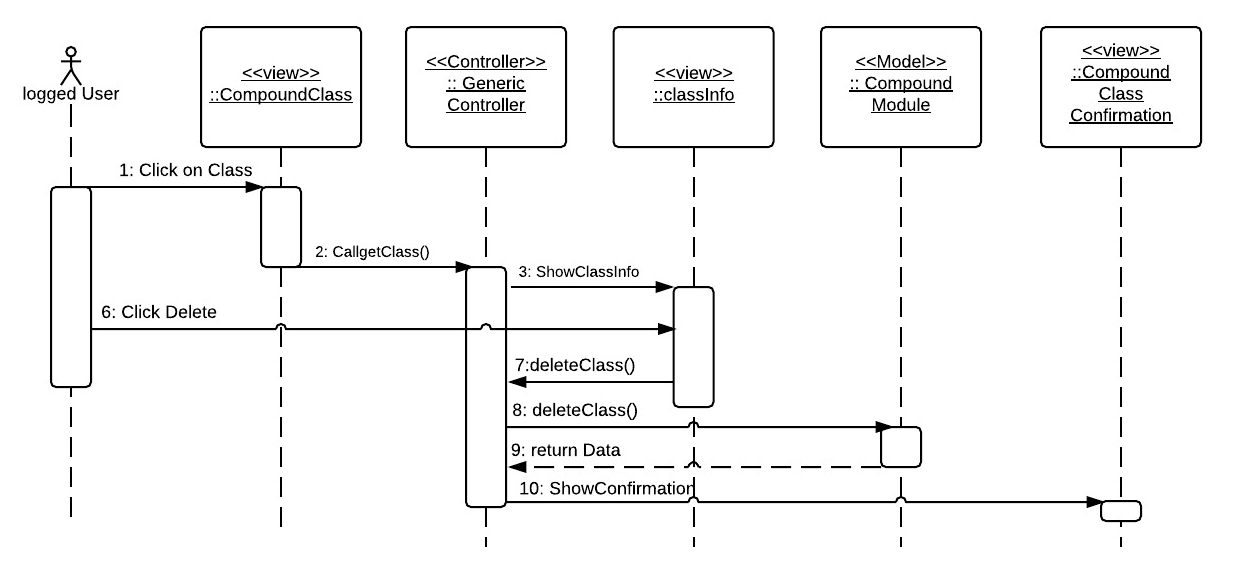
Add Compound Class Sequence Diagram



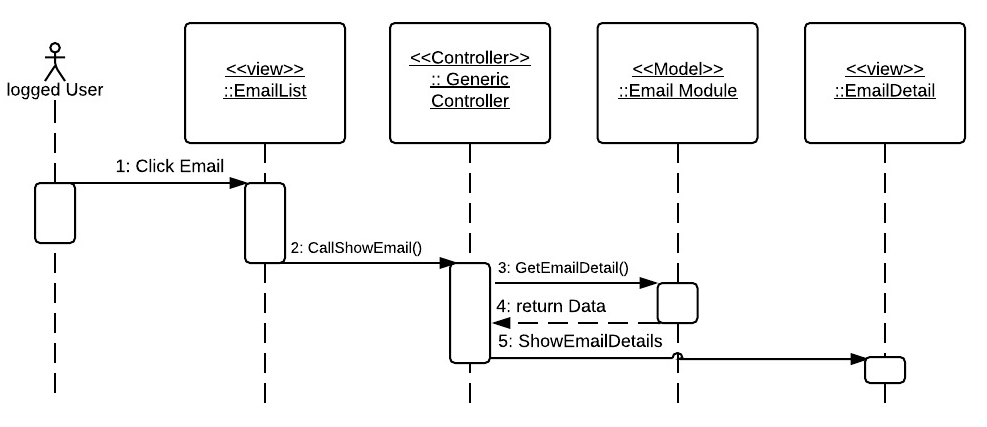
Edit Compound Class Sequence Diagram



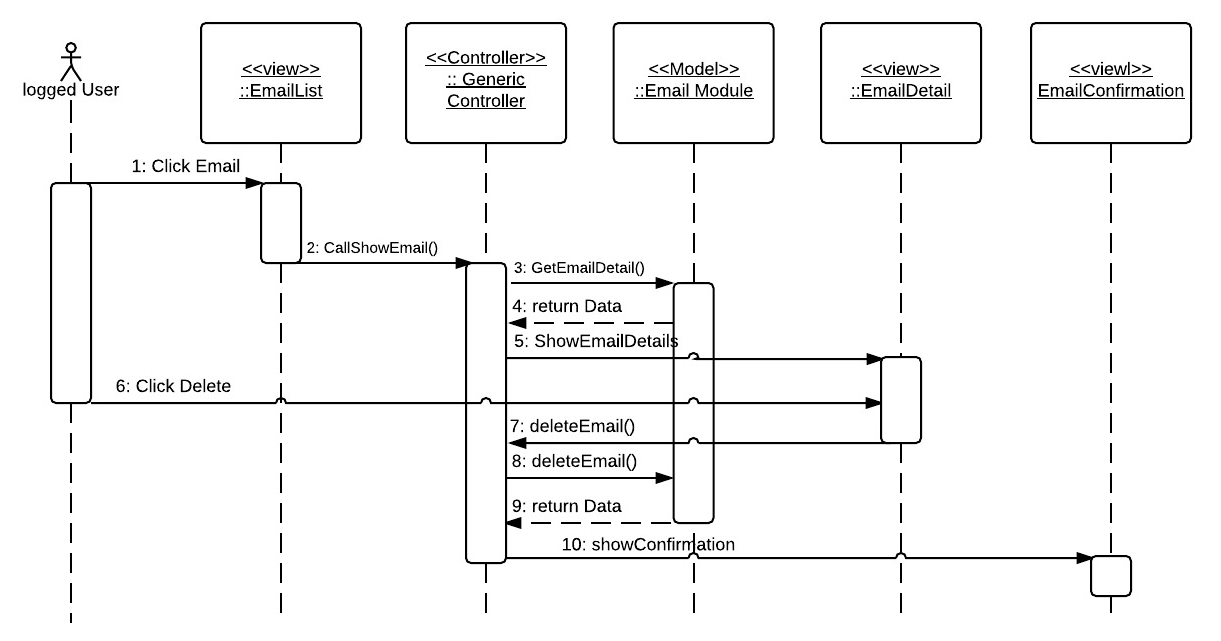
Delete Compound Class Sequence Diagram



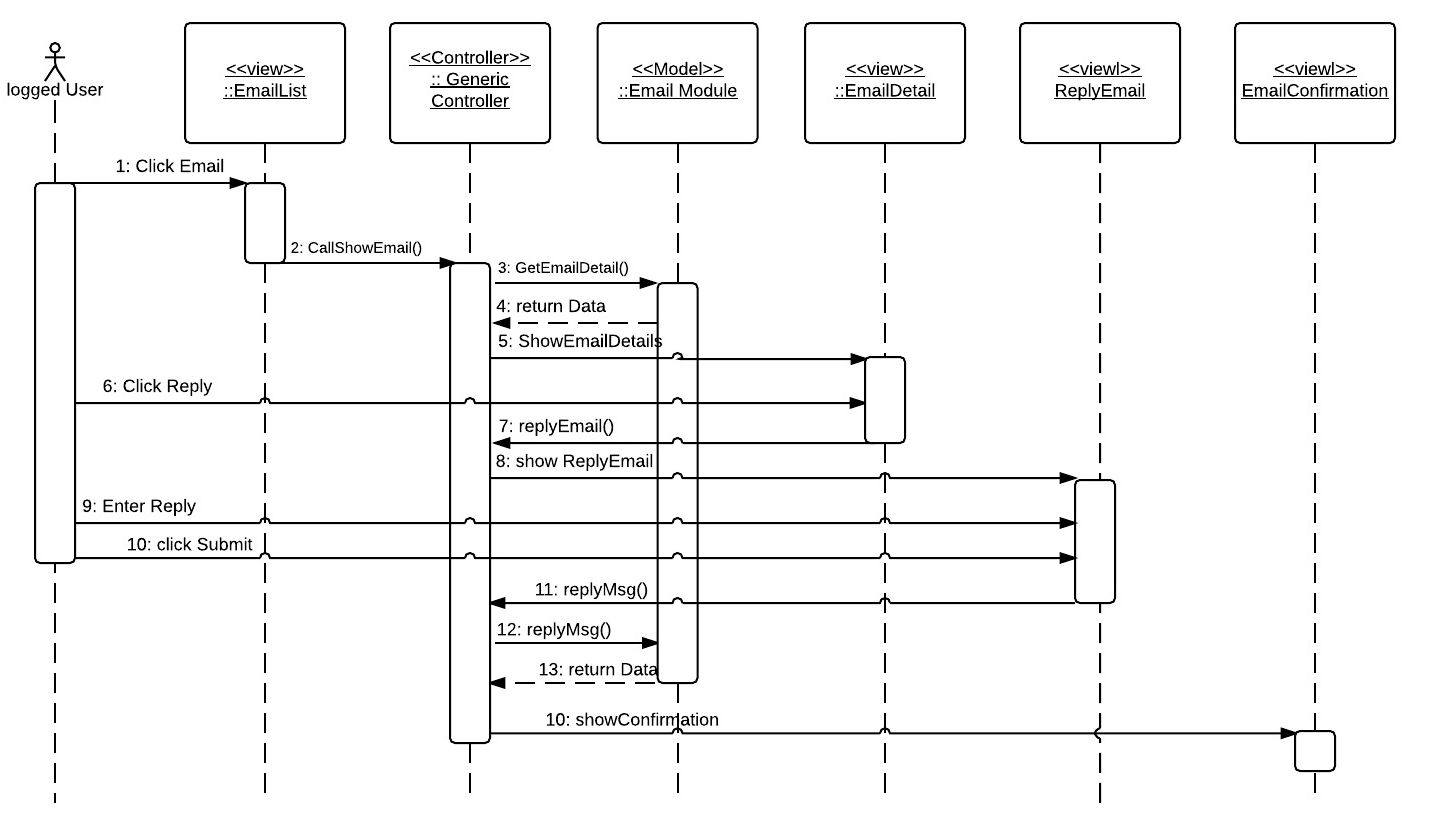
Read Email Sequence Diagram



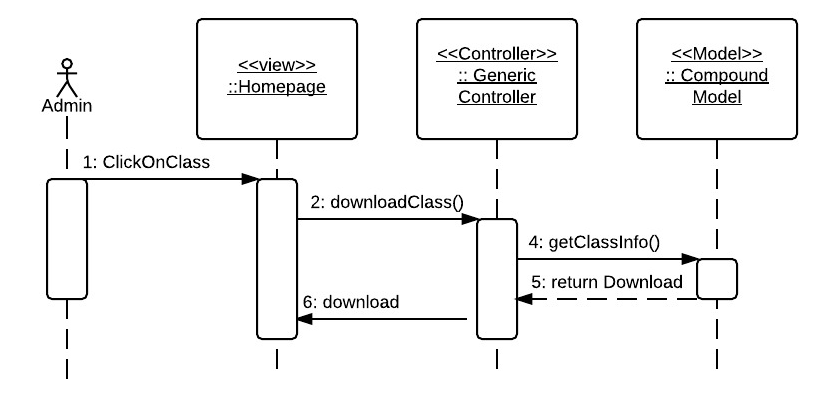
Delete Email Sequence Diagram



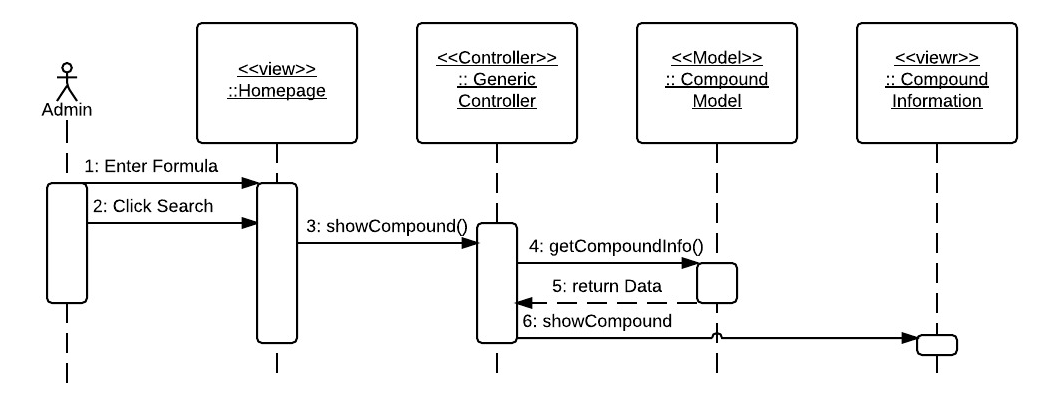
Reply email Sequence Diagram



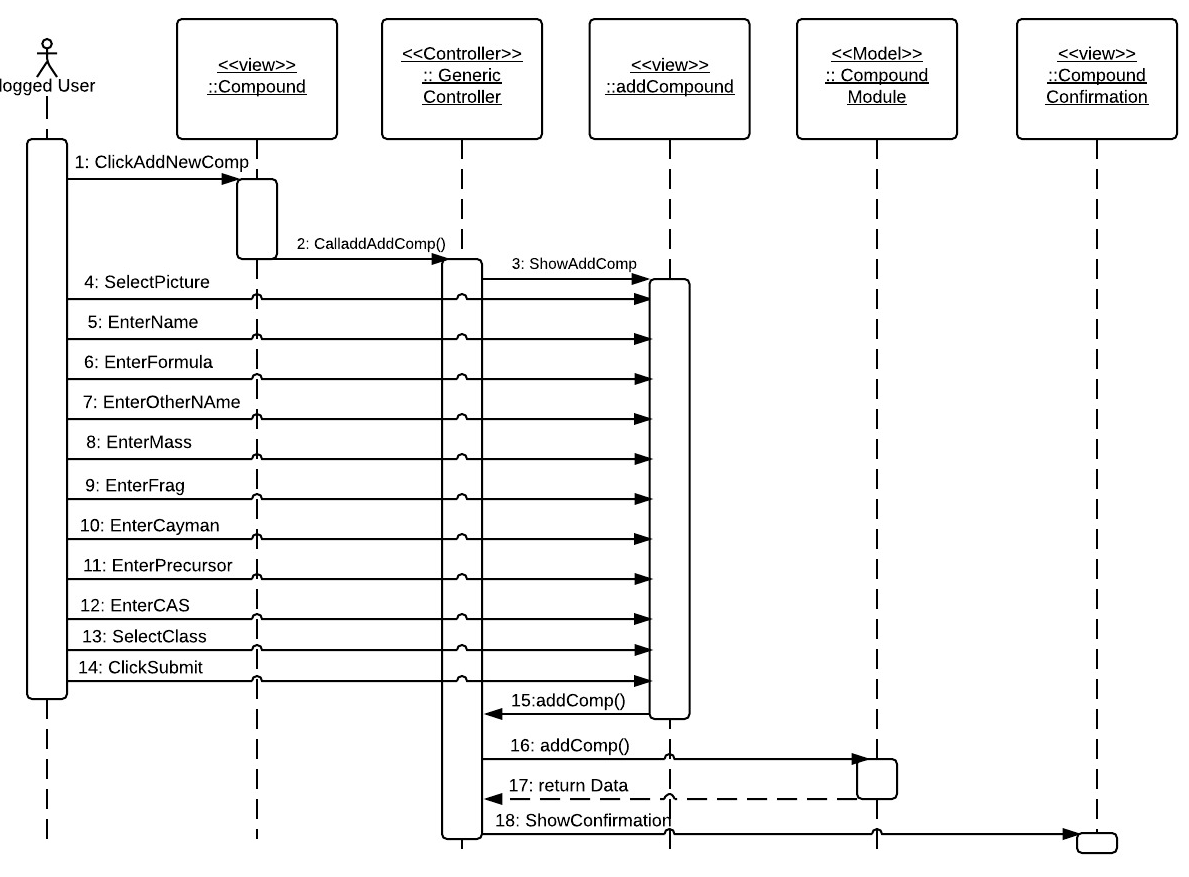
Download Class Sequence Diagram



Show Compound Sequence Diagram



Add Compound Sequence Diagram



## Appendix F – Documented Class interfaces.

There are several things that need to be specified.

1. Client Code and how was it organized
2. Server Code and how is it organized

Client Code:

For the client Code there are several pages that need to be visited:

**/index.php**: contains the main page with the view that is filled by every page

**/functions/function.js**: contains the generic controller that serves to a controller to every view and model. It also contains the route that ANGULARJS uses to dynamic reroute different pages. Also it contains the Controller used by the navigation bar and lastly it contains some global javascript variables

**/pages/….html**: This folder pages contains all the pages or views of the system. The pages that contains a JavaScript function called cntrl which are the portion that belongs to the generic controller for each specific view or page.

Server Code:

For server code you have to look in the folder/request/. This folder contains all php server processing pages (Model). In this case I have:

CompoundModuleClass.php and CompoundModuleCall.php: where all compound information is processed at the model level

UserModuleClass.php, UserModuleCall.php: where you get all user information at the model level

AuthModuleClass.php and AuthModuleCall.php: where all the authentication is done.

MsgModuleClass.php and MsgModuleCall.php: where all messages are processed at model level.

CompoundModuleClass.php

- compRequest()

- getComp()

- getCompbyId()

- addComp()

- editComp()

- deleteComp()

- impPics()

- impComp()

- getTrans()

- addTrans()

- editTrans()

- deleteTrans()

- countComp()

- checkVariableNotEmpty(valueOfVariable, nameOfVariable)

- checkEmailIsLoggedUser()

- checkisAdmin()

- checkisAdminOrLabOP()

- checkEmailIsNotLoggedUser()

- requestDatabase(returnValue)

- returnJson(result)

- returnCSV(result,title)

UserModuleClass.php

- userRequest()

- findUser()

- upgradeUser()

- downgradeUser()

- deleteUser()

- addUser()

- getMyInfo()

- setMyName()

- setMyPass()

- signUP()

- countUsers()

- checkVariableNotEmpty(valueOfVariable, nameOfVariable)

- checkEmailIsLoggedUser()

- checkisAdmin()

- checkisAdminOrLabOP()

- checkEmailIsNotLoggedUser()

- requestDatabase(returnValue)

- returnJson(result)

AuthModuleClass.php

- authRequest()

- testLoginInfo(email,password)

- testSesionInfo()

- checkUserLoggedIn()

- logOut()

- checkVariableNotEmpty(valueOfVariable, nameOfVariable)

- requestDatabase()

- returnJson(result)

MsgModuleClass.php

- userRequest()

- deleteMsg()

- newMsg()

- newReply()

- getMsg()

- modMsg()

- countMsg()

- checkVariableNotEmpty(valueOfVariable, nameOfVariable)

- requestDatabase()

- returnJson(result)

## Appendix G – Documented code for test drivers and stubs.

In this case because of lack of time Subsystem testing were not created.

## Appendix H – Diary of meeting and tasks for the entire semester.

### Sprint 1

|  |  |
| --- | --- |
| Diary Entry 1 | |
| Date | Monday, February 2rd, 2015 |
| Location | OE building |
| Start | 1:00 PM |
| End | 2:00 PM |
| Attendees | * Carlos Dominguez * Dr. Luis Arroyo |
| Agenda | * Talk about the system that needs to be done. * Create system user stories. |
| Summary | * Explanation of the current system. * Brief definition of the functionalities to be implemented. * Created some user stories based on the explanation |
| Assigned Tasks | * Get familiar with the tools to be used. * Create a mock up for acceptance. |

|  |  |
| --- | --- |
| Diary Entry 2 | |
| Date | Friday, February 6th, 2015 |
| Location | OE building |
| Start | 1:00 PM |
| End | 2:00 PM |
| Attendees | * Carlos Dominguez * Dr. Luis Arroyo |
| Agenda | * Show the mock up to where it was. * Show authentication system. |
| Summary | * Showed the mocked up to where it was done. * Talked about authentication system and showed what I had done. |
| Assigned Tasks | * Continue working on first Sprint |

|  |  |
| --- | --- |
| Diary Entry 3 | |
| Date | Friday, February 13th, 2015 |
| Location | OE building |
| Start | 1:00 PM |
| End | 2:00 PM |
| Attendees | * Carlos Dominguez * Dr. Luis Arroyo |
| Agenda | * Show what was done on Sprint 1 * Assigned for User Stories for Sprint 2 |
| Summary | * Showed the mocked up. * Show all that was done during Sprint 1. * New user story were assigned |
| Assigned Tasks | * Get ready to start spring 2 next Monday |

### Sprint 2

|  |  |
| --- | --- |
| Diary Entry 4 | |
| Date | Friday, February 27th, 2015 |
| Location | OE building |
| Start | 1:00 PM |
| End | 2:00 PM |
| Attendees | * Carlos Dominguez * Dr. Luis Arroyo |
| Agenda | * Show what was done in Sprint 2. * Assign User Stories to Sprint 3 |
| Summary | * Showed what was done in sprint 2. * Dr. Arroyo assigned user stories for sprint 3 |
| Assigned Tasks | * Get ready to start spring 3 next Monday |

### Sprint 3

|  |  |
| --- | --- |
| Diary Entry 5 | |
| Date | Friday, March 20th, 2015 |
| Location | OE building |
| Start | 1:00 PM |
| End | 2:00 PM |
| Attendees | * Carlos Dominguez * Dr. Luis Arroyo |
| Agenda | * Show what was done on Sprint 3 * Assigned for User Stories for Sprint 4 |
| Summary | * Show all that was done during Sprint 3. * New user story were assigned for Sprint 4 * Dr. Arroyo pointed out that different compounds could have the same formula, yet provide different information. |
| Assigned Tasks | * Get ready to start spring 4 next Monday * Fix what needed to be fixed on Sprint 4 regathing the new clarification on compounds formulas. |

### Sprint 4

|  |  |
| --- | --- |
| Diary Entry 5 | |
| Date | Friday, April 3rd, 2015 |
| Location | OE building |
| Start | 1:00 PM |
| End | 2:00 PM |
| Attendees | * Carlos Dominguez * Dr. Luis Arroyo |
| Agenda | * Show what was done on Sprint 4 * Assigned for User Stories for Sprint 5 |
| Summary | * Show all that was done during Sprint 4. * New user story were assigned for Sprint 5 * Dr. Arroyo said he wanted to change some of the information the institute is going to be sharing. |
| Assigned Tasks | * Get ready to start spring 5 next Monday * Fix the information that is to be shared on Sprint 5. |

### Sprint 5

|  |  |
| --- | --- |
| Diary Entry 5 | |
| Date | Friday, April 17th, 2015 |
| Location | OE building |
| Start | 1:00 PM |
| End | 2:00 PM |
| Attendees | * Carlos Dominguez * Dr. Luis Arroyo |
| Agenda | * Show what was done on Sprint 4 * Assigned for User Stories for Sprint 5 |
| Summary | * We discussed the agenda.. He was happy with the end result |
| Assigned Tasks | * No more tasks assigned. |

# References

awwwards-team. (2014, February 20). Retrieved February 02, 2015, from awwwards-team.com: http://www.awwwards.com/what-are-frameworks-22-best-responsive-css-frameworks-for-web-design.html

VizTeams. (2013, July 30). Retrieved February 02, 2015, from VizTeams.com: http://www.vizteams.com/blog/android-ios-top-solutions-for-creating-cross-platform-mobile-apps/#.VM7lbEfF-X0