

System Design Document

MyStudyGroupPlanner

Version 1.2

Client

Katie Hirsch

Team 2

Aparna V. Kaliappan

Ying Zhang

Siqi Lin

Sean Murren

Tyler Campbell

4/28/2016

MyStudyGroupPlanner System Design Document

Table of Contents

	<u>Page</u>
1. Changes in this Document	
2. Introduction	
2.1 Purpose of This Document	
2.2 References	
3. System Architecture	
3.1 Architectural Design	
3.2 Decomposition Description	
4. Persistent Data Design	
4.1 Database Descriptions	
5. Requirements Matrix	
Appendix A – Agreement Between Customer and Contractor	
Appendix B – Team Review Sign-off	
Appendix C – Document Contributions	

1. Changes in this Document

The Persistent Data Design has been modified with additional tables.

2. Introduction

2.1 Purpose of this document

The purpose of this document is to describe the design of the MyStudyGroupPlanner application. Key topics covered in this document include the high level system architecture, low level class design, and the persistent data design of MyStudyGroupPlanner.

2.2 References

Throughout this document, references will be made to:

1. MyStudyGroupPlanner System Requirements Specification Document
2. <https://docs.angularjs.org/guide/introduction>
3. <http://getbootstrap.com/>
4. http://www.w3schools.com/aspnet/mvc_intro.asp
5. <https://www.lucidchart.com>
6. <https://www.djangoproject.com/>

3. System Architecture

3.1 Architectural Design

The MyStudyGroupPlanner application's structural framework will be created using AngularJS, which uses a Model-View-Controller (MVC) architectural model. The application will be implemented using Django, which provides a high-level web framework that uses the Python language. We will also use Bootstrap for well-formatted features such as buttons. The database used in this application will be managed using MariaDB.

Figure 1 shows that AngularJS's MVC architectural model provides a simple interconnection between the web server, users, and the database, through the model, view, and controller. The controller will interpret input/data provided by the user and will store information in the web server, and this will be done using Django. The view will manage the display of info to the users, and this will be done using HTML, CSS and Bootstrap. The model will manage the behavior and data of the whole application, and will interact with the database using MariaDB.

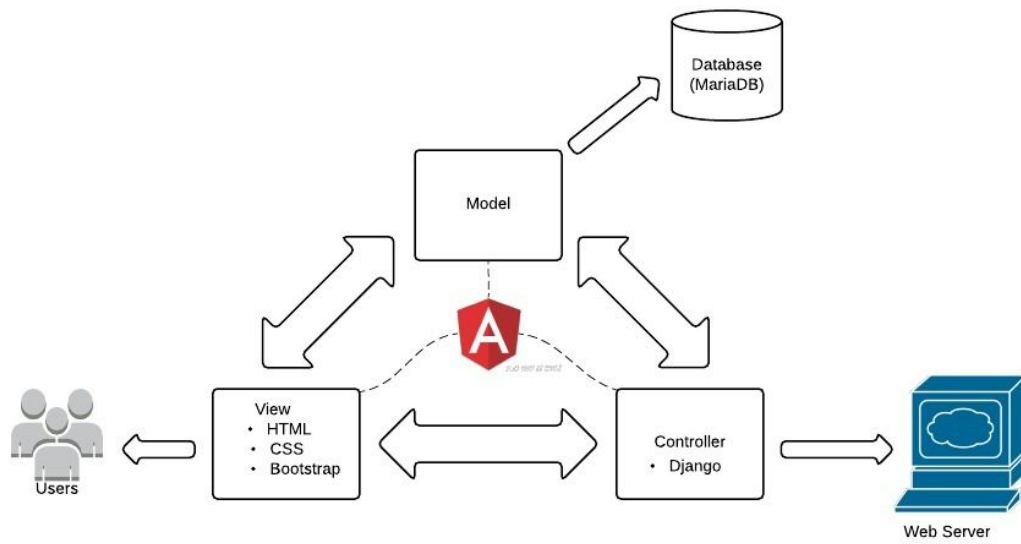


Figure 1. Architectural Design

3.2 Decomposition Description

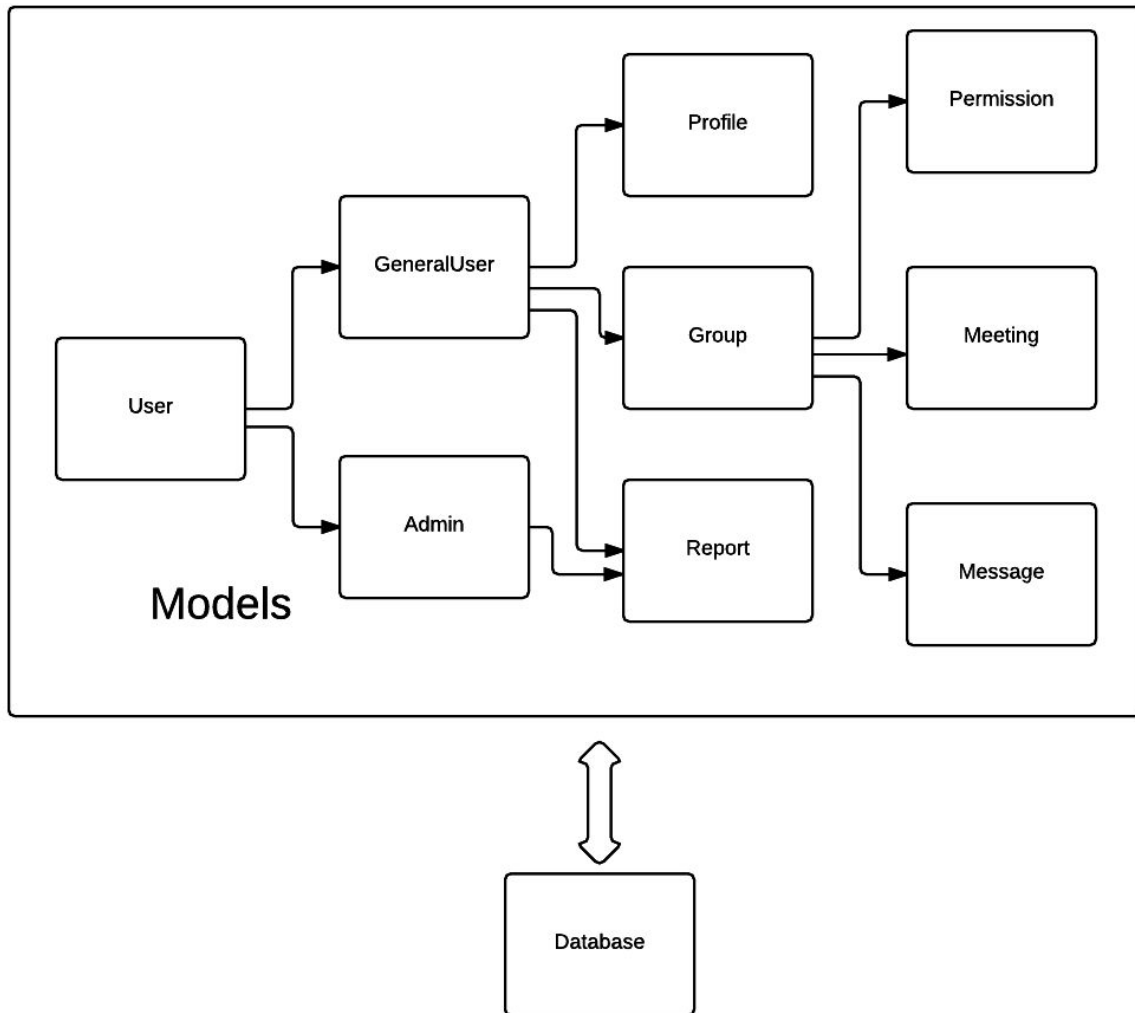


Figure 2. Models

Figure 2 illustrates the relationships between the classes we will be using when coding the application. Since we will have two types of users, GeneralUser and Admin, we will have a User class that encompasses the attributes common to both classes. GeneralUser and Admin will contain attributes unique to their own classes. A GeneralUser has a profile page, so it will have a Profile class. A GeneralUser can also be part of a group, so there is a Group class. Members of a group can send messages, attend meetings, and have permission levels. So, there are three classes that extend from Group: Permission, Meeting, and Message. A general user can send reports, while an admin can view reports. So, there is a Report class that can be accessed by both the GeneralUser class and the Admin class. The database will constantly be communicating with these models, in order to send stored data, and receive new data.

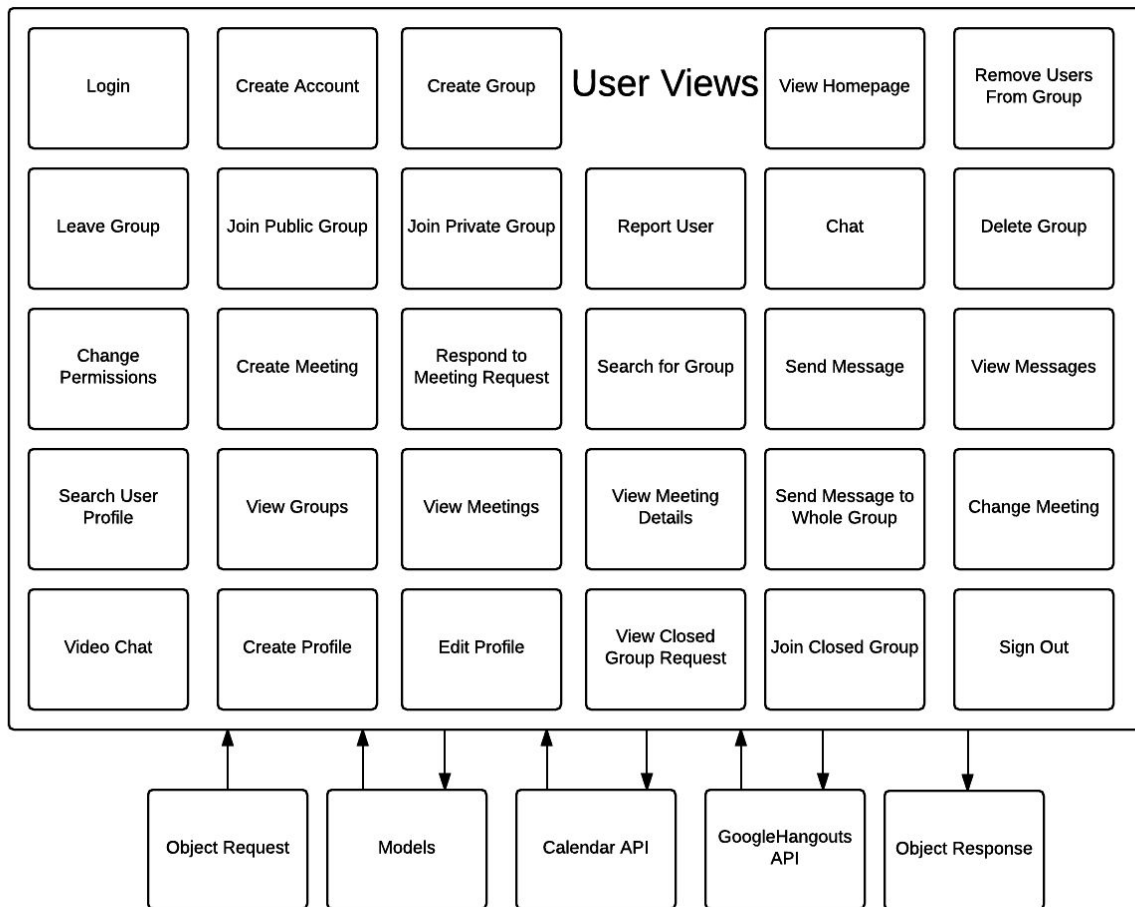


Figure 3. User Views

Figure 3 shows all of the different views that a user may see while using the application. For instance they will see a page to login to the application, a page with chatting facilities, a page to view messages, a page to search for a group, etc. The models will continually interact with the user views. The controller object will continually request user input through the views and will receive responses as well. A calendar API will be interacting with the views in order to update newly created meeting dates to the calendar. The GoogleHangouts API will be interacting to provide the chat and video chat services.

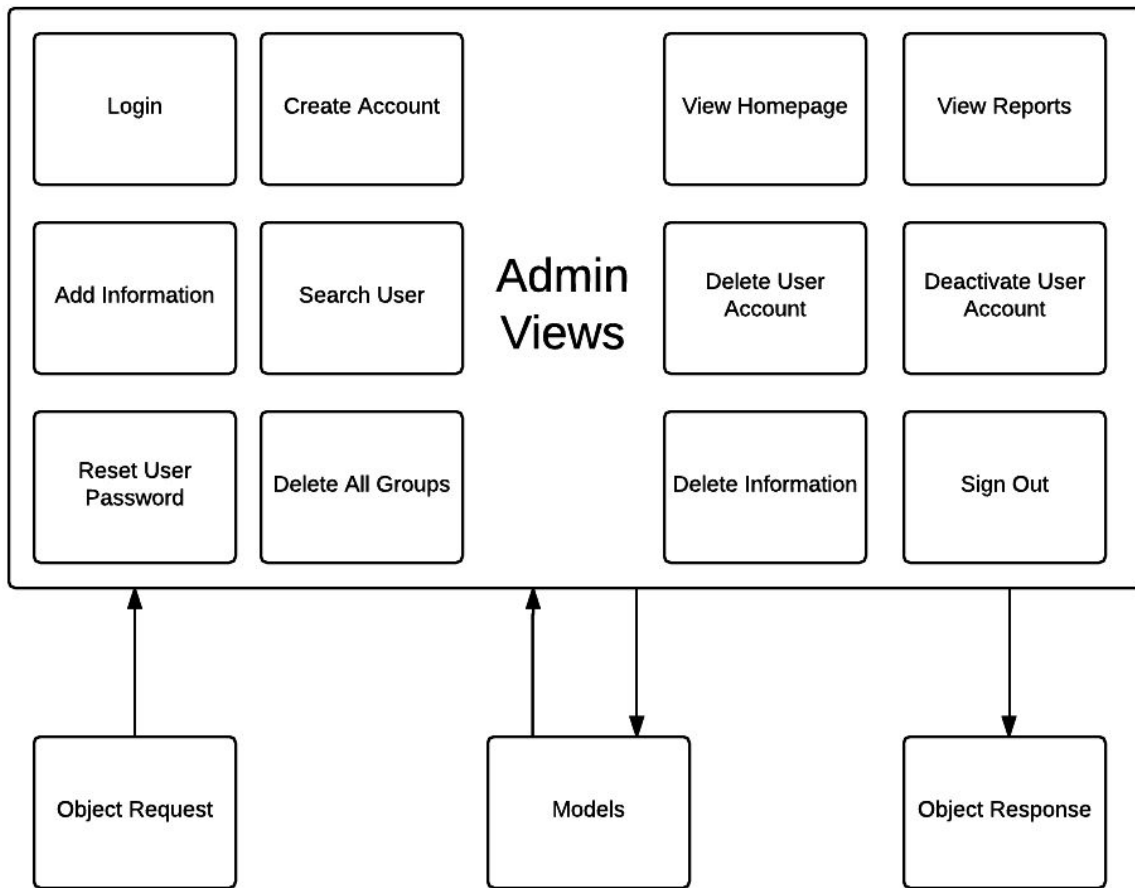


Figure 4. Admin Views

Figure 4 shows all of the different views that an admin may see while using the application. For instance they will see a page to login to the application, a page to view reports, a page to add information to the database, etc. The models will continually interact with the admin views. The controller object will continually request admin input through the views and will receive responses as well.

4. Persistent Data Design

All application data will be kept in a MariaDB database called MSGP. We will be using the CherryPy Python web framework to create the database. The database will store data about each user and admin, group information, permissions, and the data of each notification and report created.

User
ID
Email
Password
First_name
Last_name
Username
Is_admin
Created_at
Updated_at

Group
ID
groupName
Subject
ClassName
Section
GroupOwner
MemberCount
TotalMembersAllowed

Chat
ID
UserID
GroupID
Message
DateTime

Meeting
ID
Title
Building
Room_num
Start_time
End_time
Users_attending
GroupID
Comment

Building
ID
Name
Abrv

Room
ID
Building
Room_num

Day
ID
Room
Day

Hour
ID
Room
Start_hour
End_hour

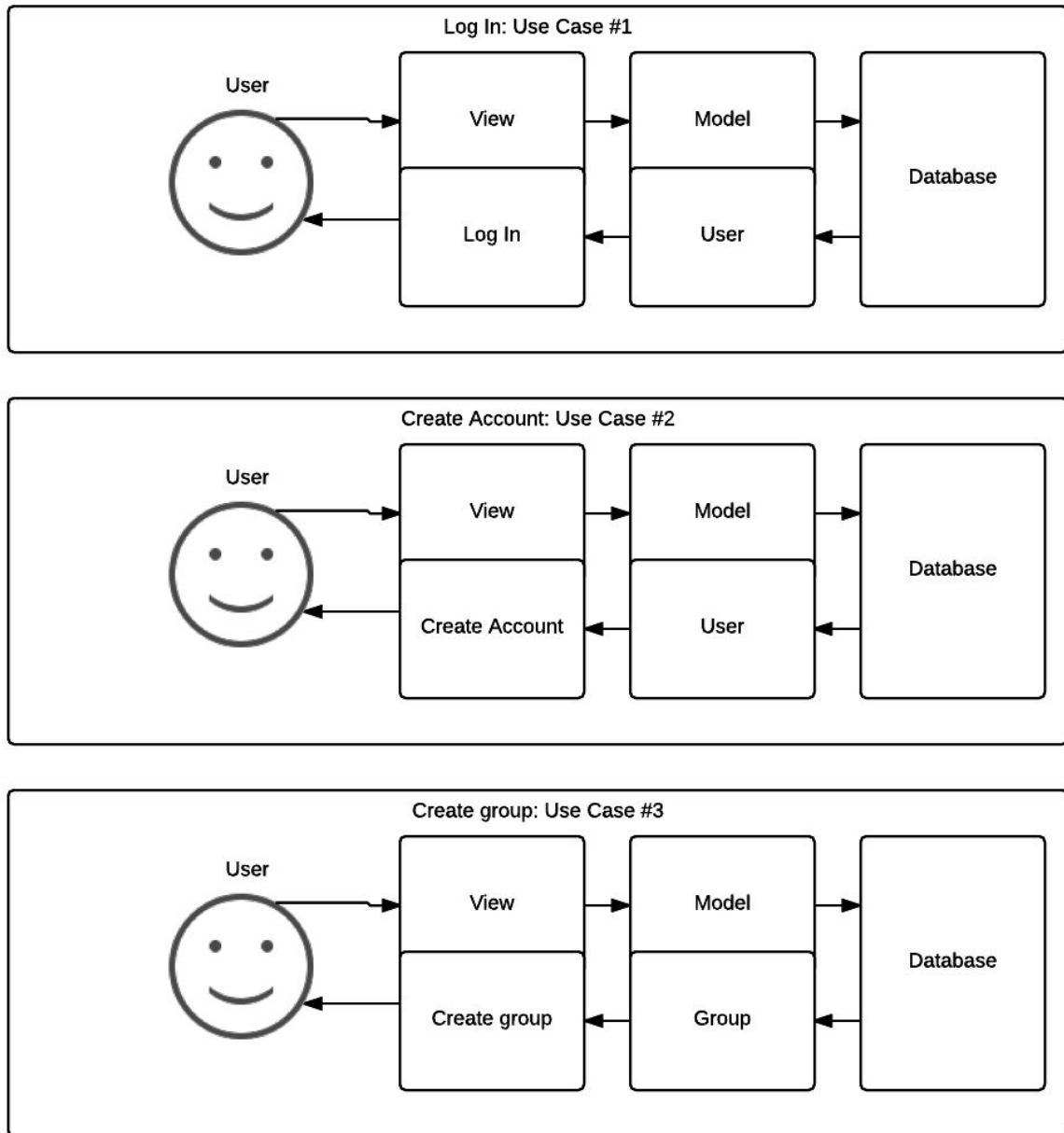
MsgpUser
ID
MsgpUserID
MsgpUsername
MsgpGroupID
MsgpGroupName
MsgpMeetingID

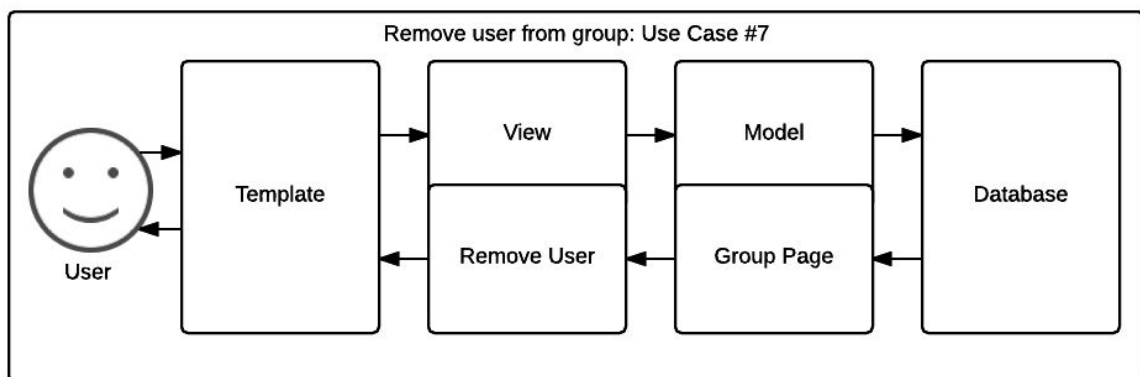
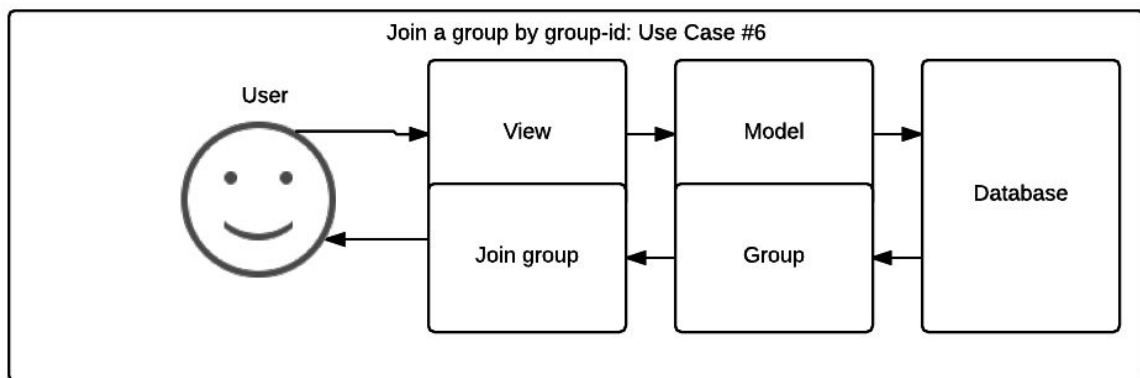
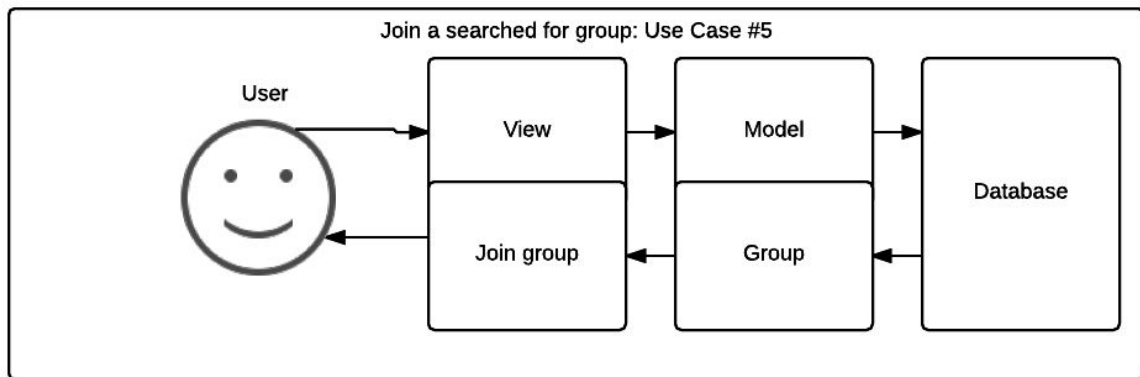
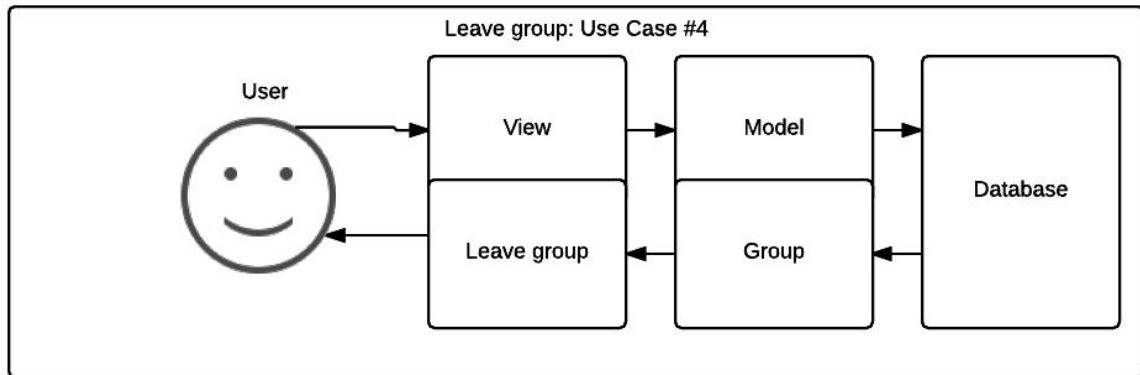
Report
ID
Reporter
Reportee
Comments
Date_submitted

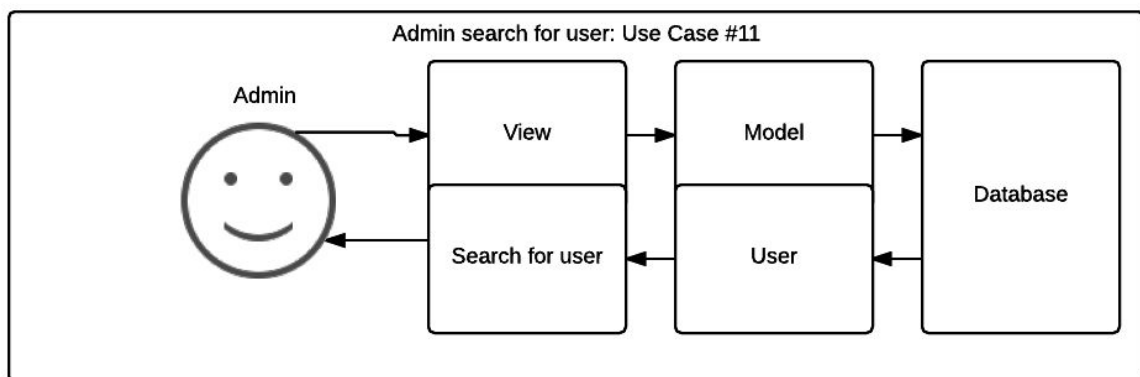
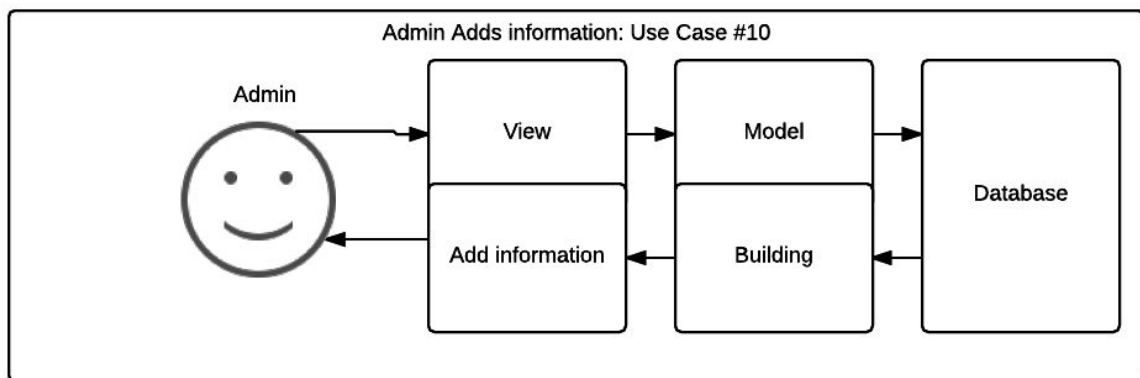
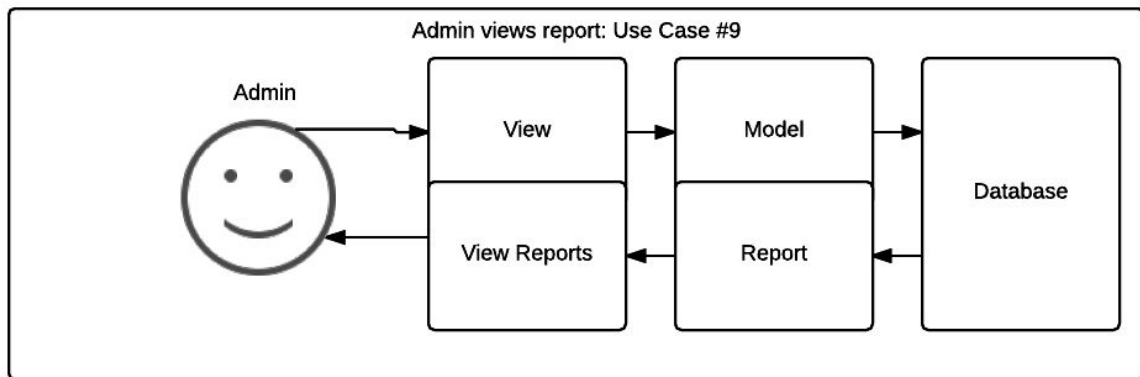
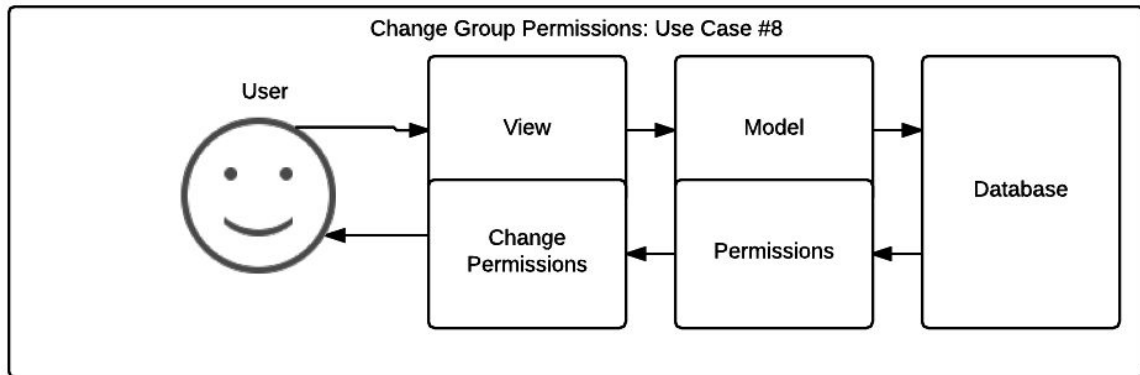
5. Requirements Matrix

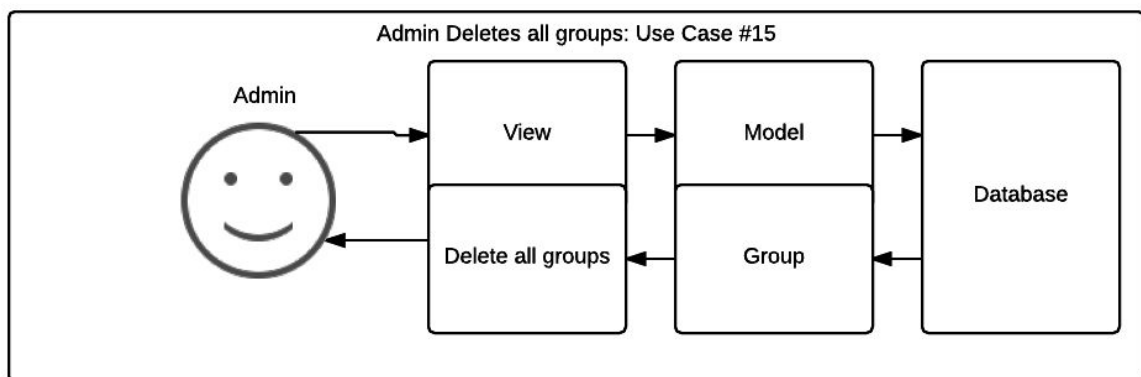
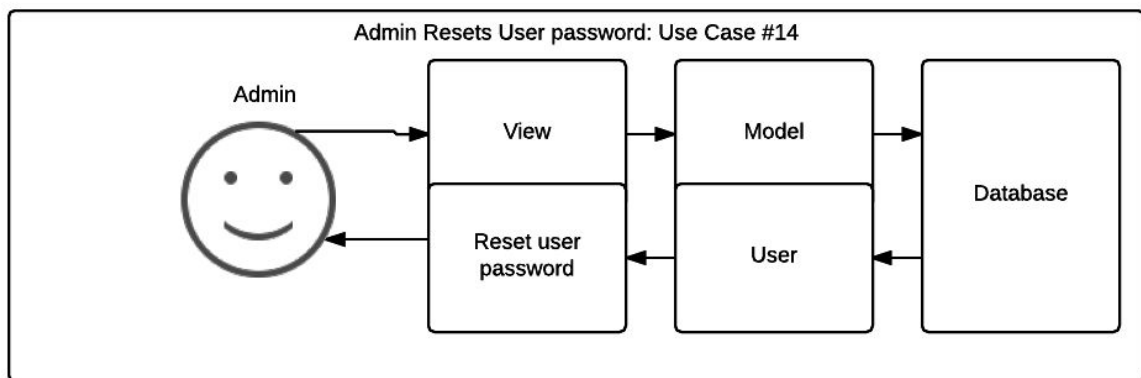
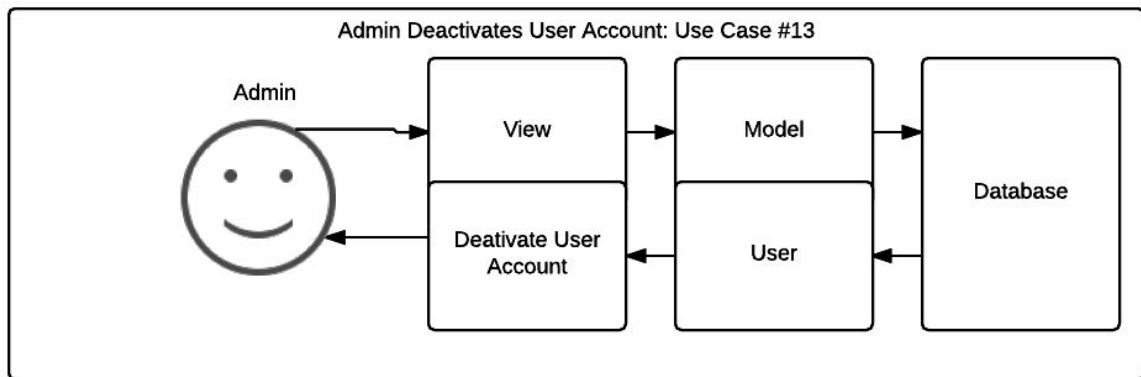
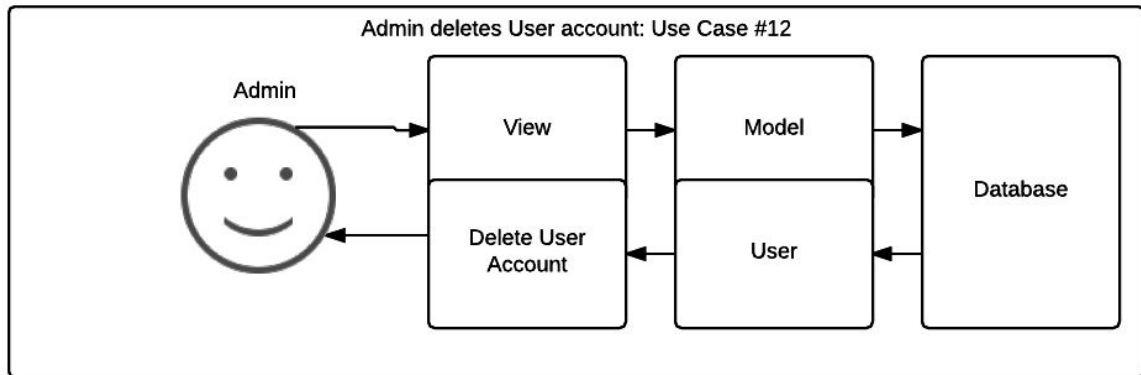
Please refer to the System Requirements Specification for details regarding the corresponding use cases.

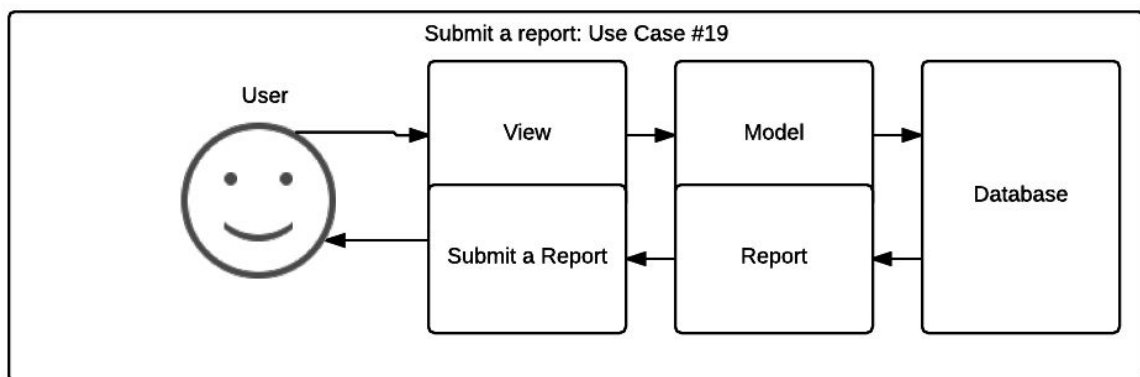
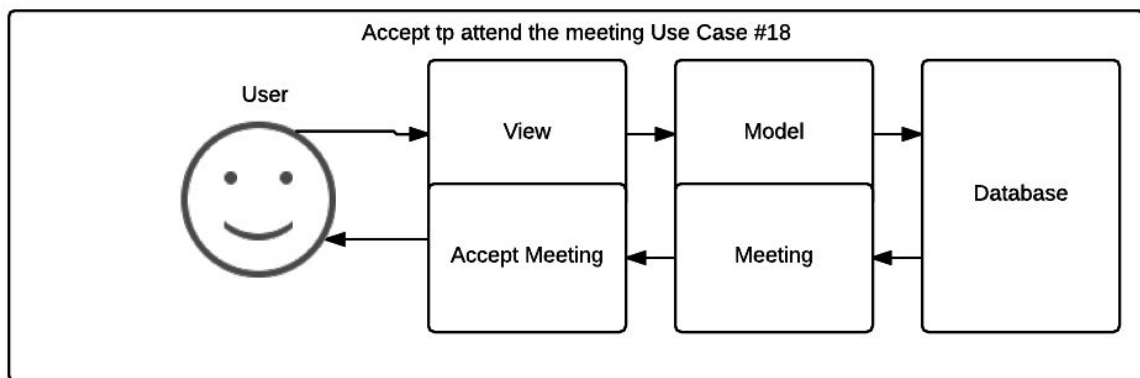
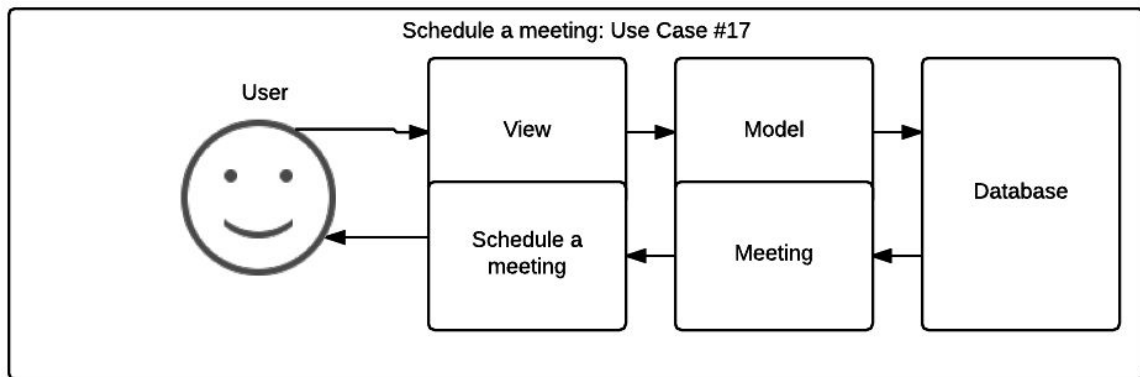
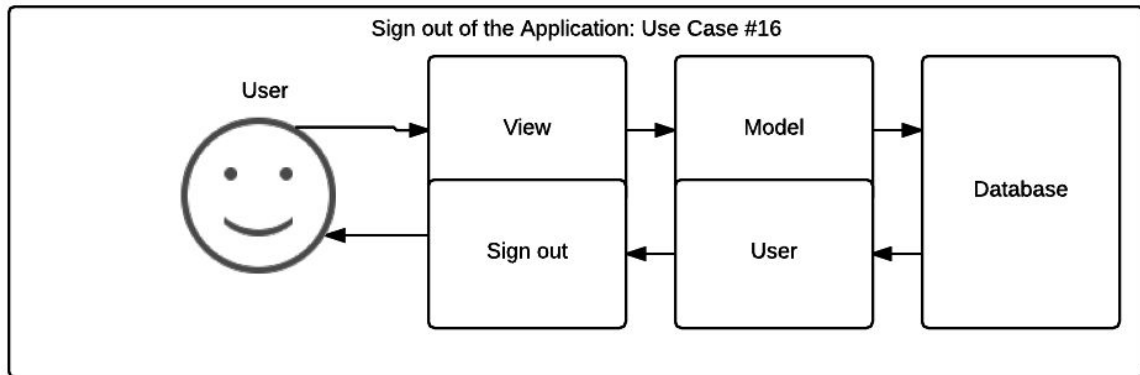
MyStudyGroupPlanner Requirements Matrix

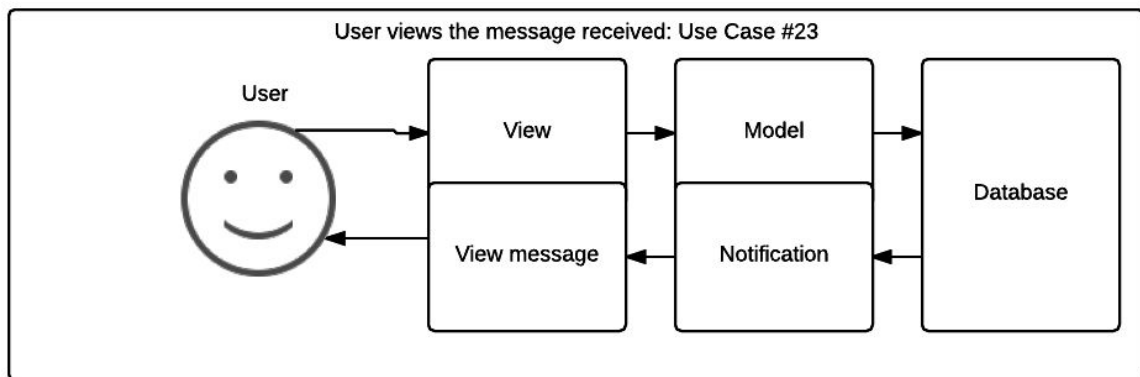
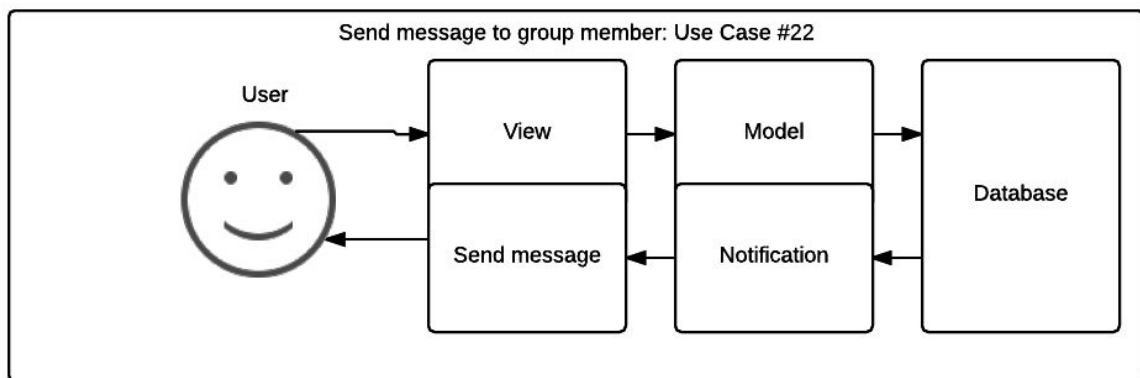
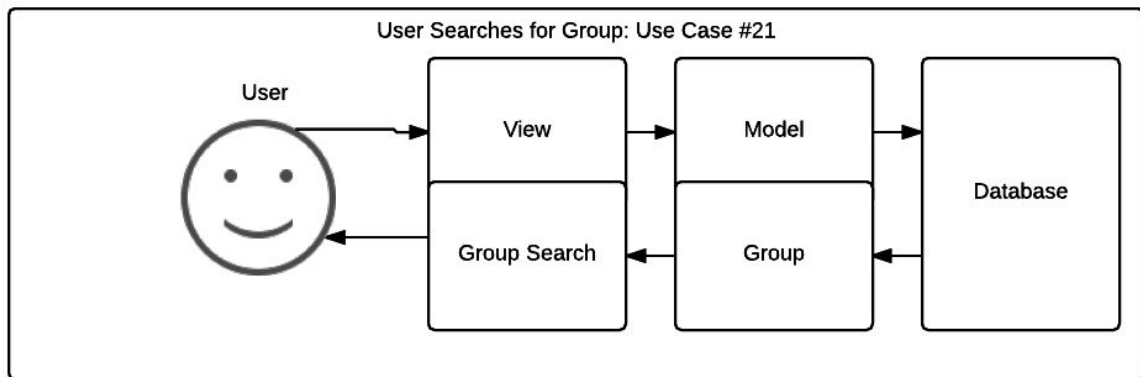
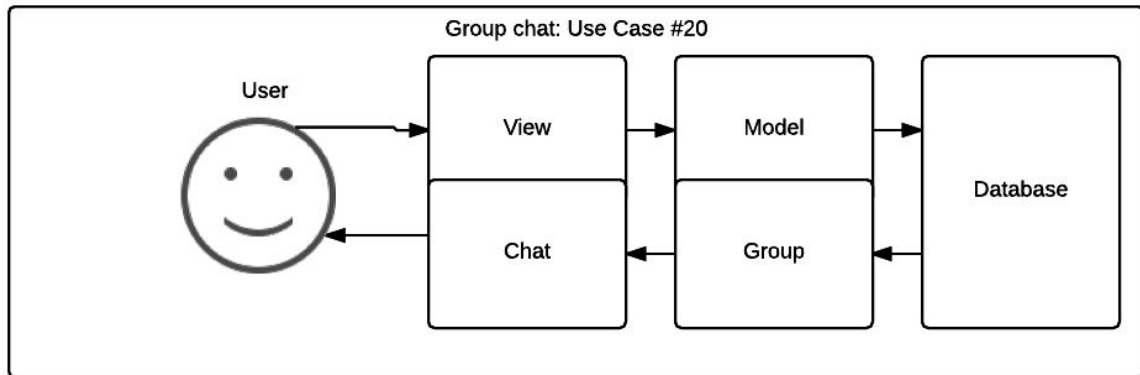


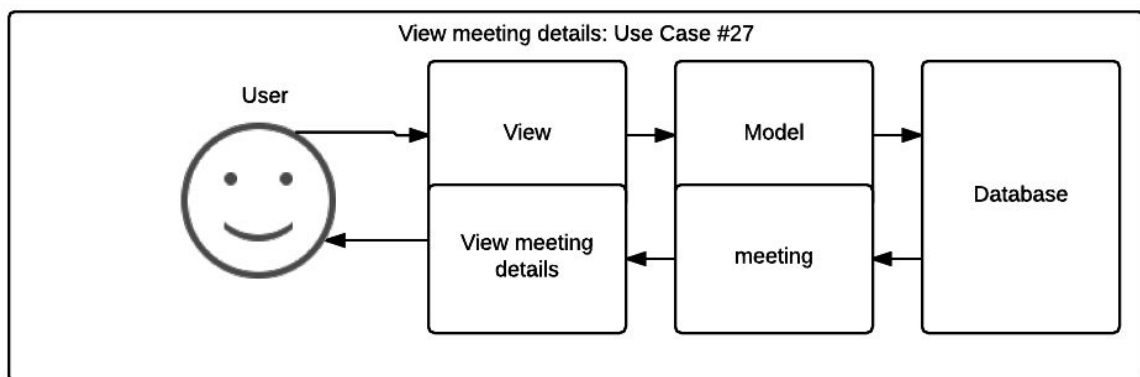
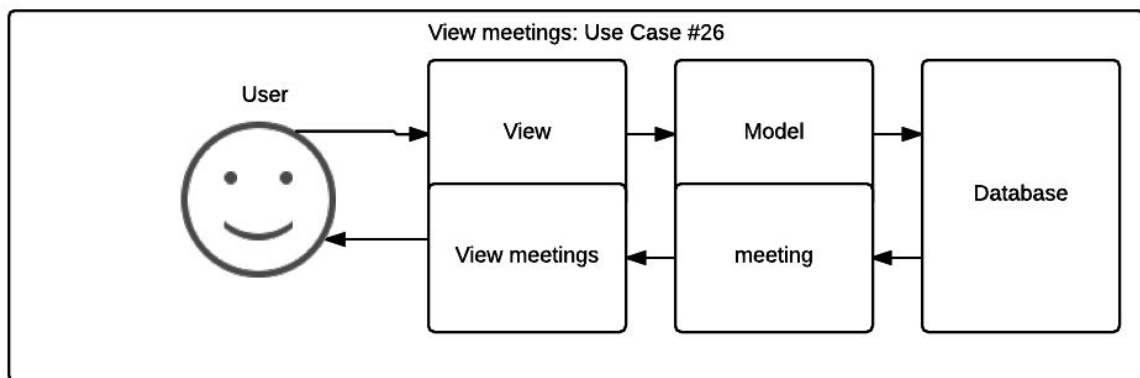
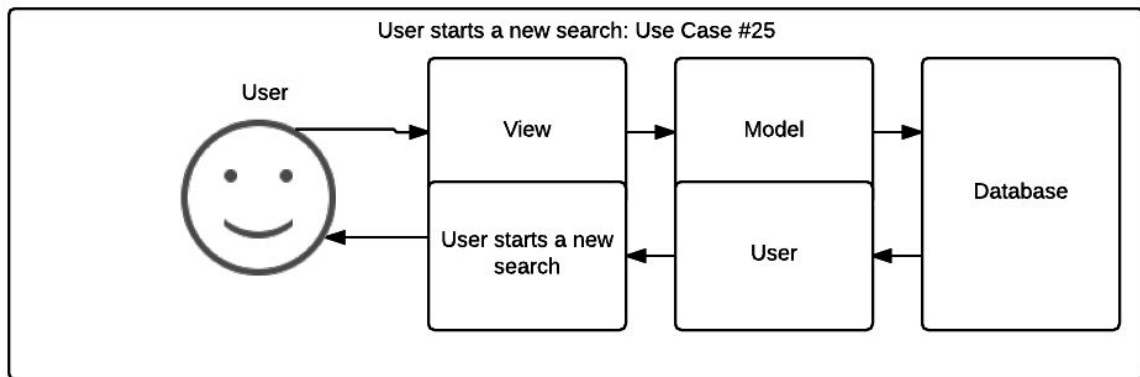
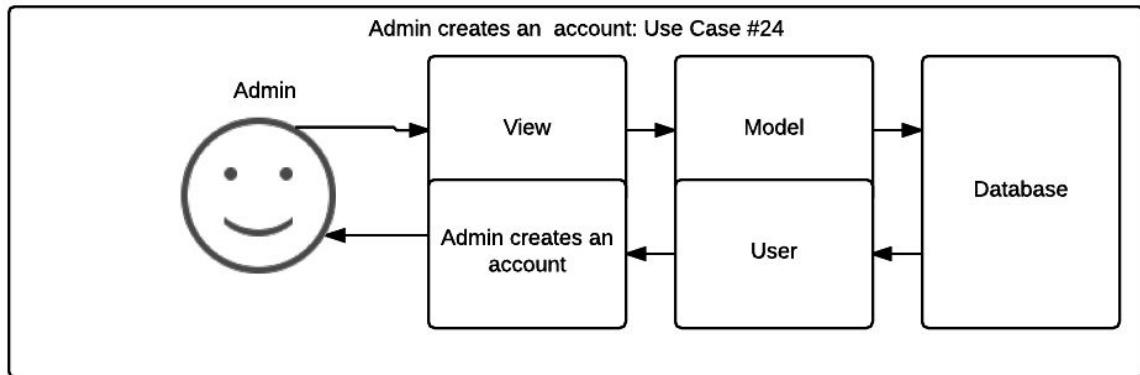


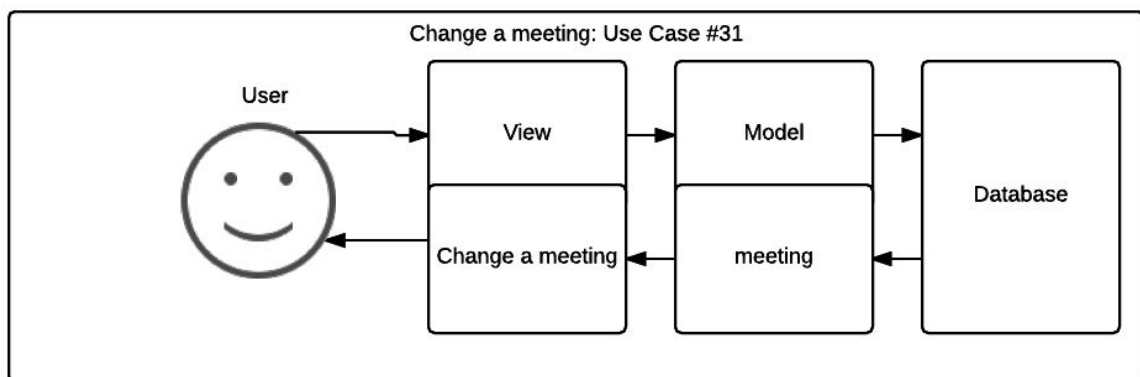
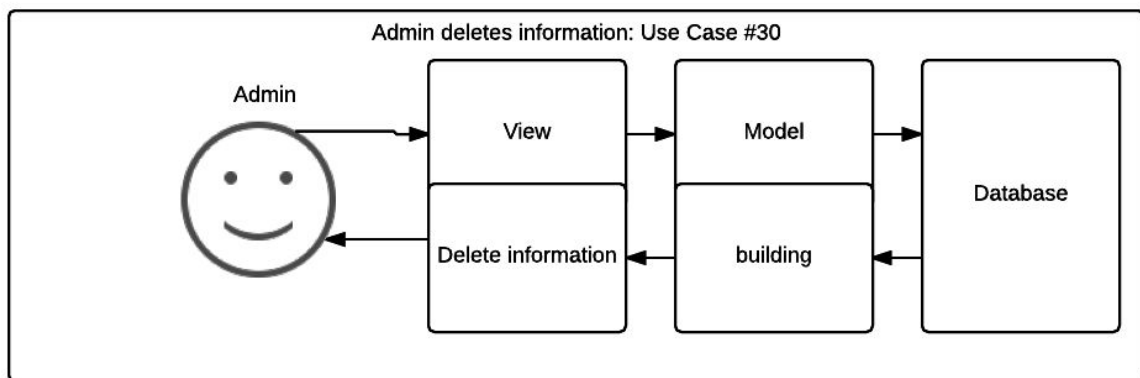
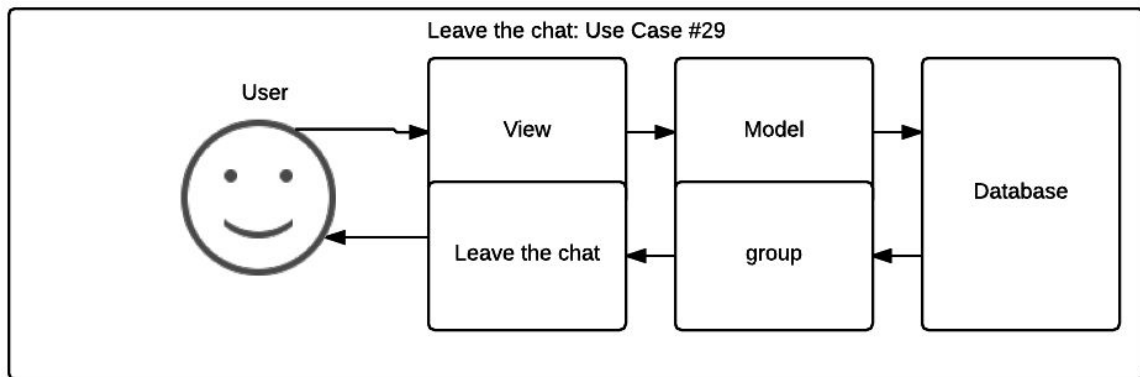
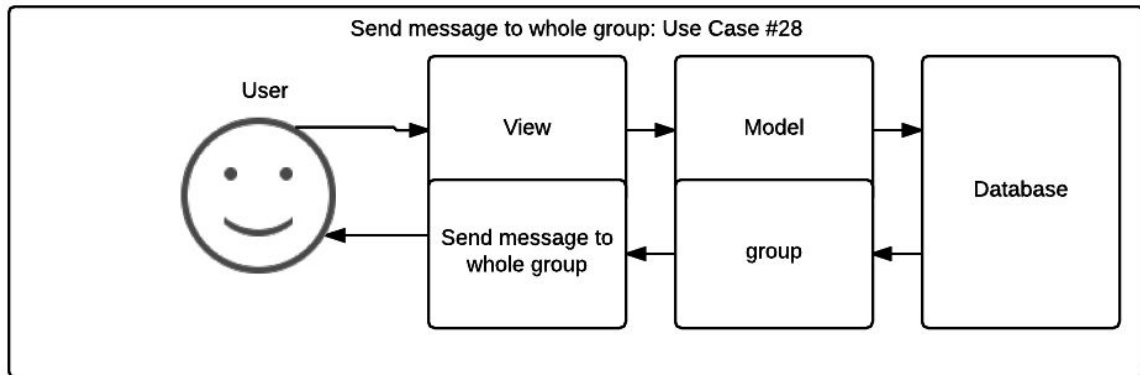


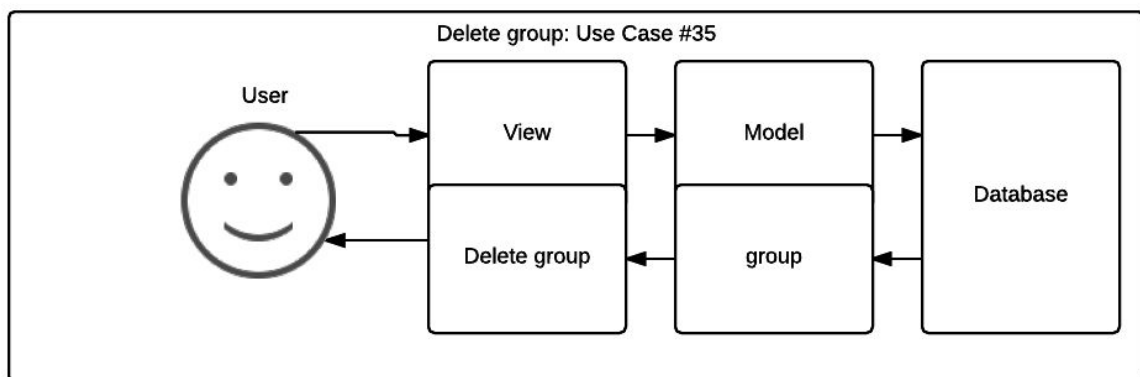
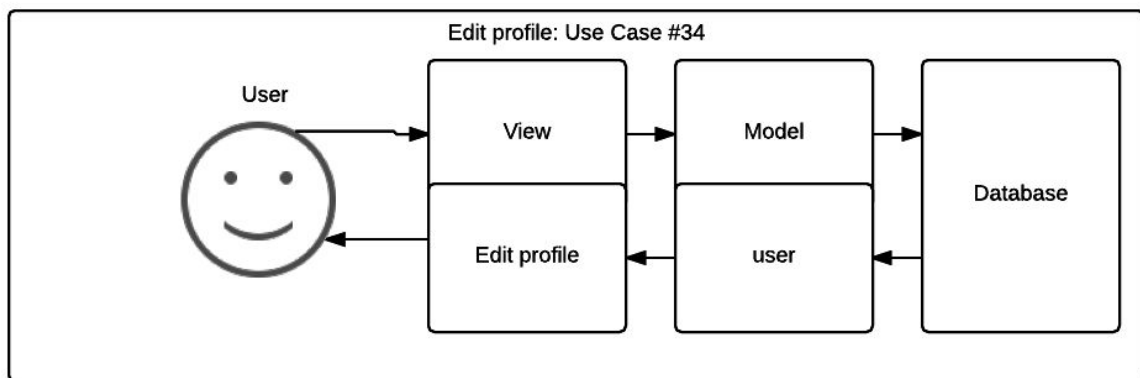
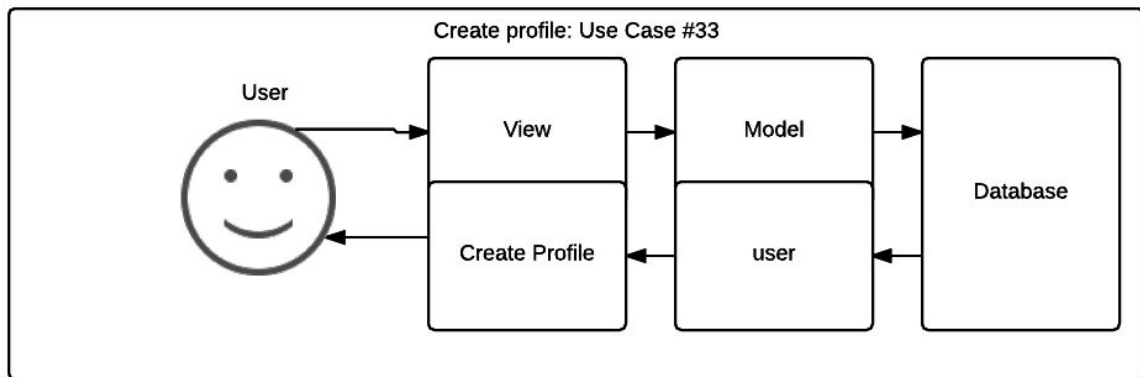
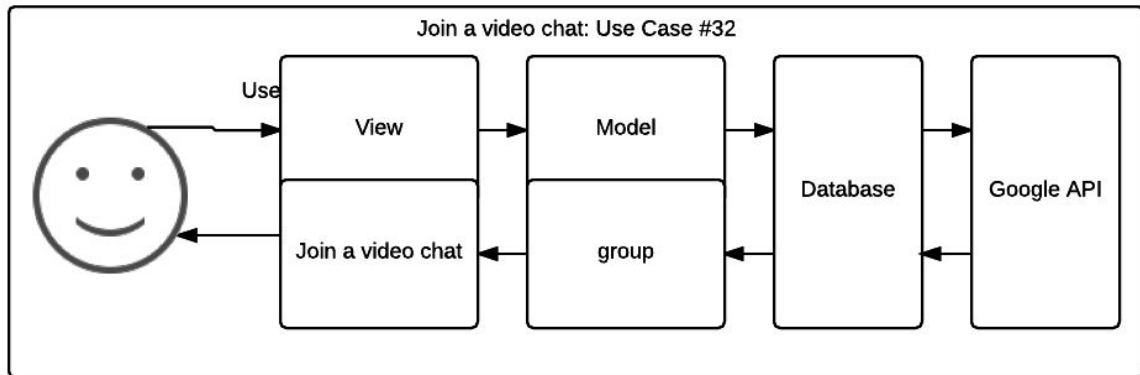


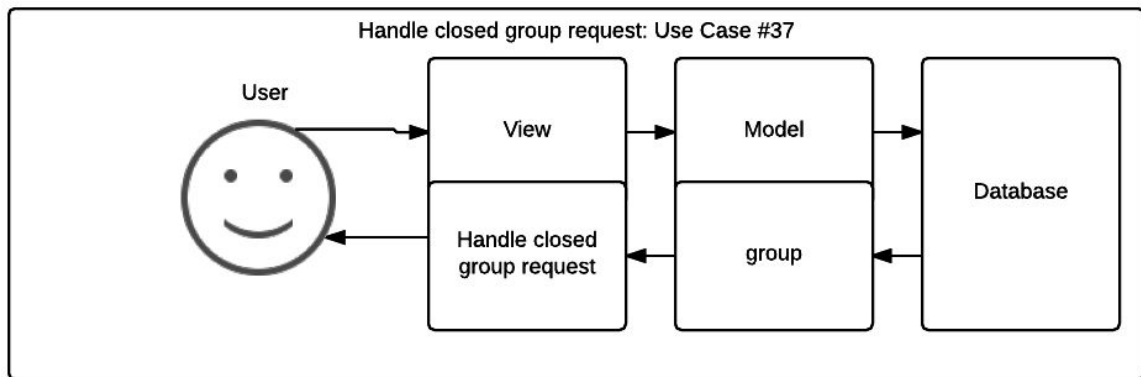
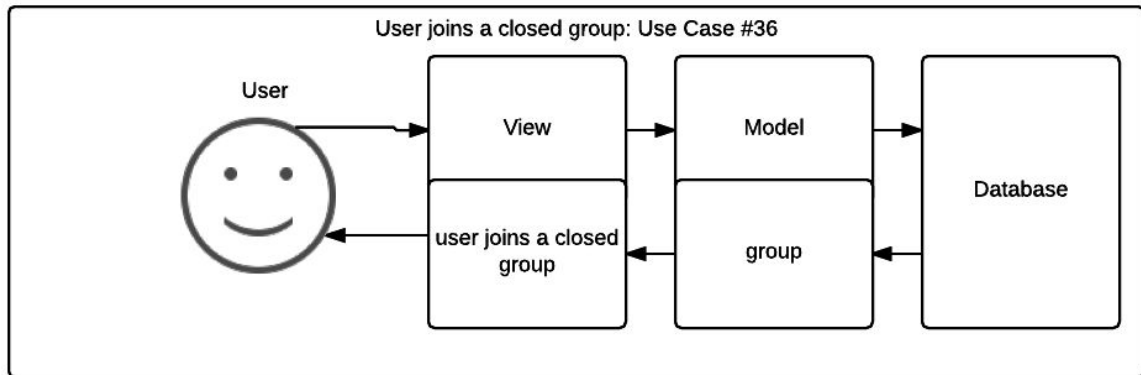












Appendix A - Agreement Between Customer and Contractor

The customer and the development team agree that the MyStudyGroupPlanner application performs all of the high-priority functionalities described in this System Requirements Specification. Use cases with lower priority will be addressed during future development of this application.

In the case that there are any future changes to the requirements outlined in this document, the development team will make these changes and provide the customer with updated hardcopy and softcopy versions to be read, approved, and signed.

Customer Comments:

Customer Signature

Print Name

Date

Signature

Development Team Signatures

Print Name

Signature

Print Name

Signature

Print Name

Signature

Print Name

Signature

Print Name

Signature

Date

Date

Date

Date

Date

Appendix B - Team Review Sign-off

All members of the MyStudyGroupPlanner development team have reviewed this document and agree on its content and format. Any disagreements about this document are documented below.

Development Team Signatures

Print Name

Date

Signature

Comments:

Print Name

Date

Signature

Comments:

Print Name

Date

Signature

Comments:

Print Name

Date

Signature

Comments:

Print Name

Date

Signature

Comments:

Appendix C - Document Contributions

- **Group**
 - Discussed and drew the diagrams of the Architectural Design and the Models by hand
- **Tyler Campbell**
 - Section 2: Introduction Section
 - Section 5: Requirements Matrix
- **Aparna Kaliappan**
 - Section 3.1: Architectural Design description
 - Section 3.2: Decomposition Description diagrams and descriptions of the Models, User Views, and Admin Views
 - Appendices A and B
- **Sean Murren**
 - Section 4: Persistent Data Design
- **Ying Zhang**
 - Architectural Diagram
- **Siqi Lin**
 - None