System Design Document

MyStudyGroupPlanner

Version 1.1

Client

Katie Hirsch

Team 2

Aparna V. Kaliappan
Ying Zhang
Siqi Lin
Sean Murren
Tyler Campbell

4/12/2016

MyStudyGroupPlanner
System Design Document

Table of Contents

Page

1. Changes in this Document

Appendices A and B have been added. We have updated our architectural design.

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.

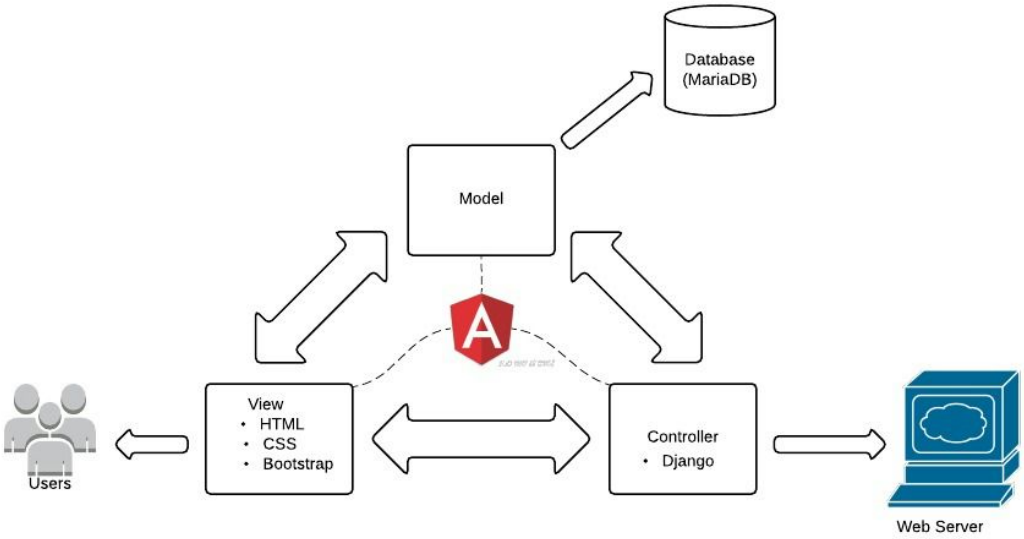


Figure 1. Architectural Design

3.2 Decomposition Description

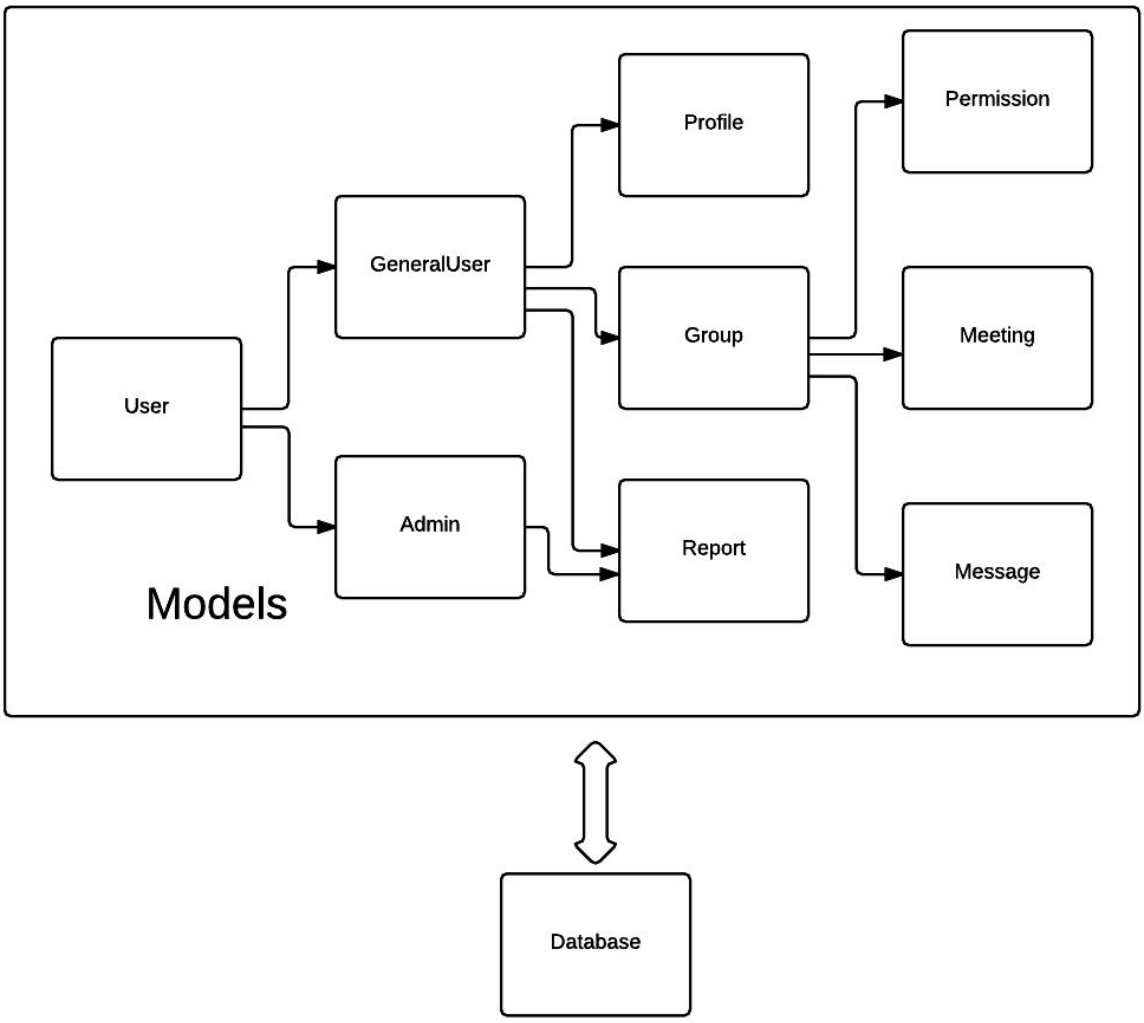


Figure 2. Models

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.

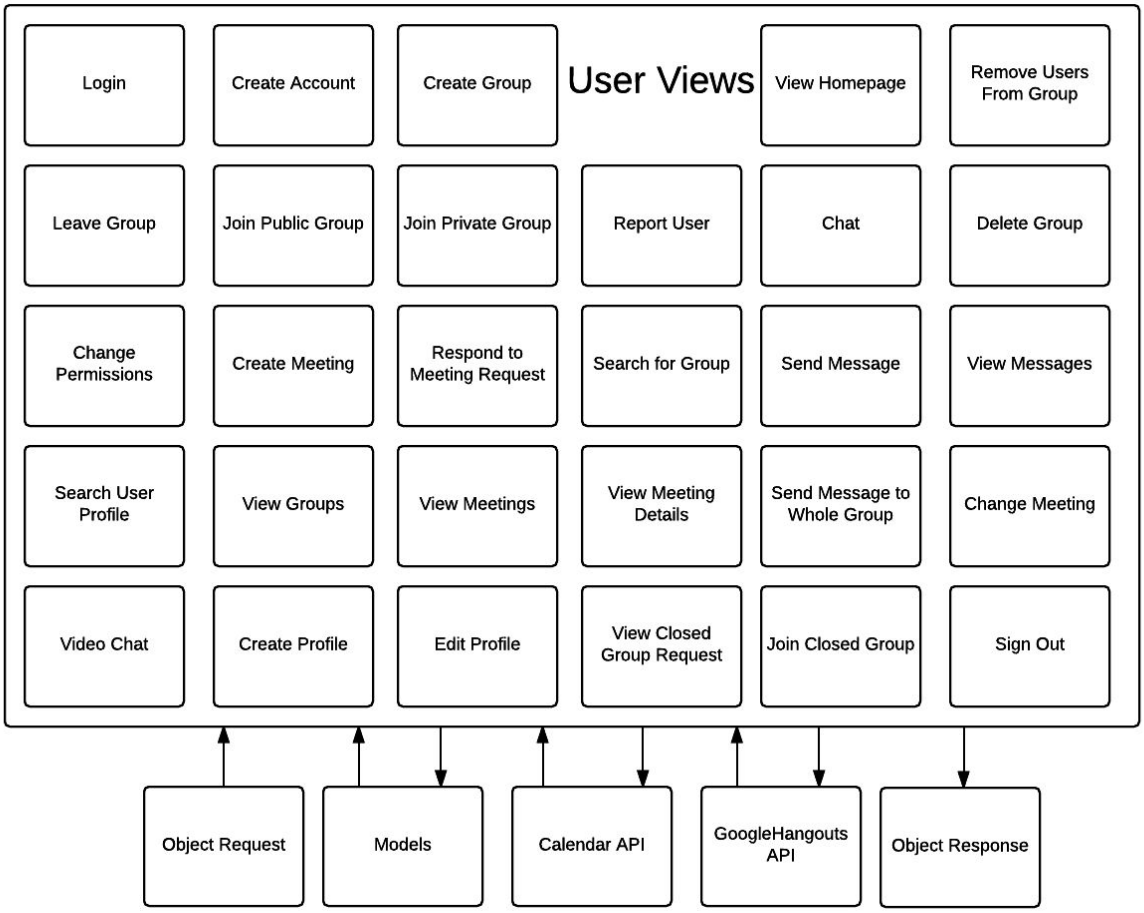


Figure 3. User Views

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.

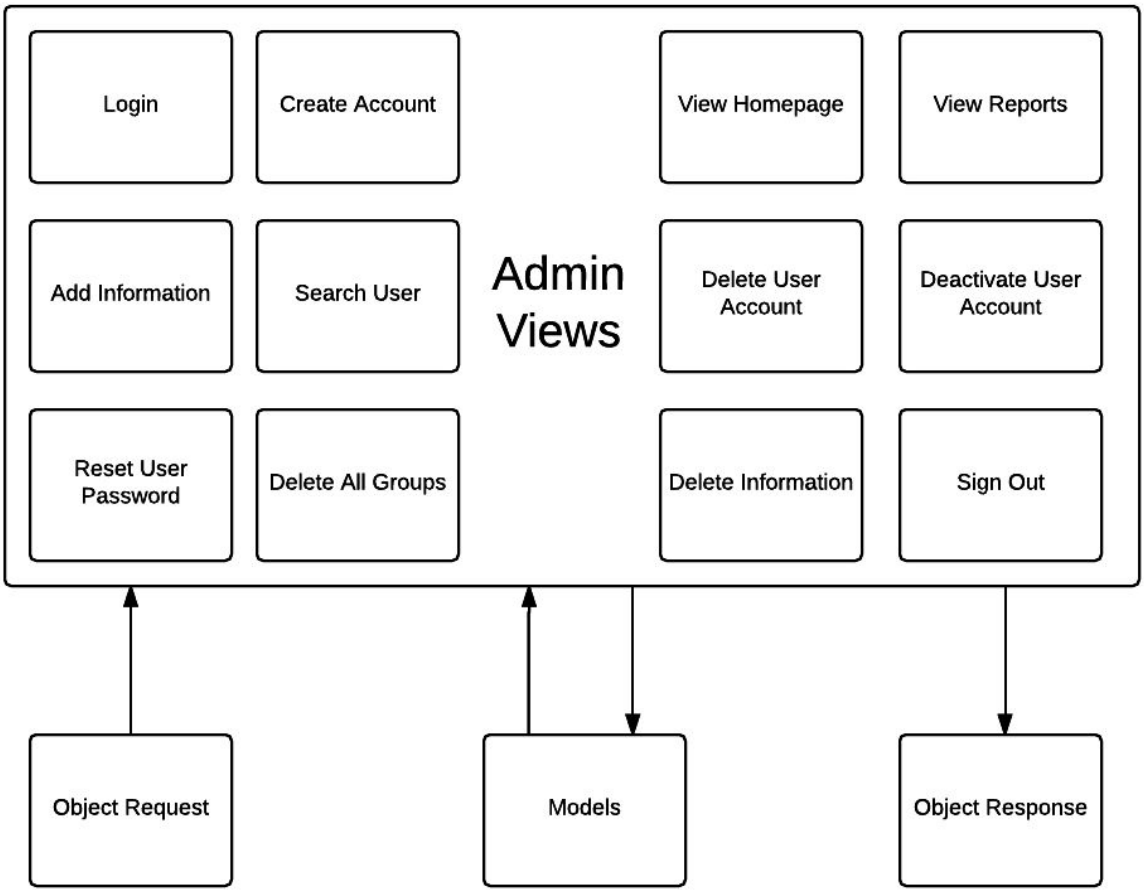
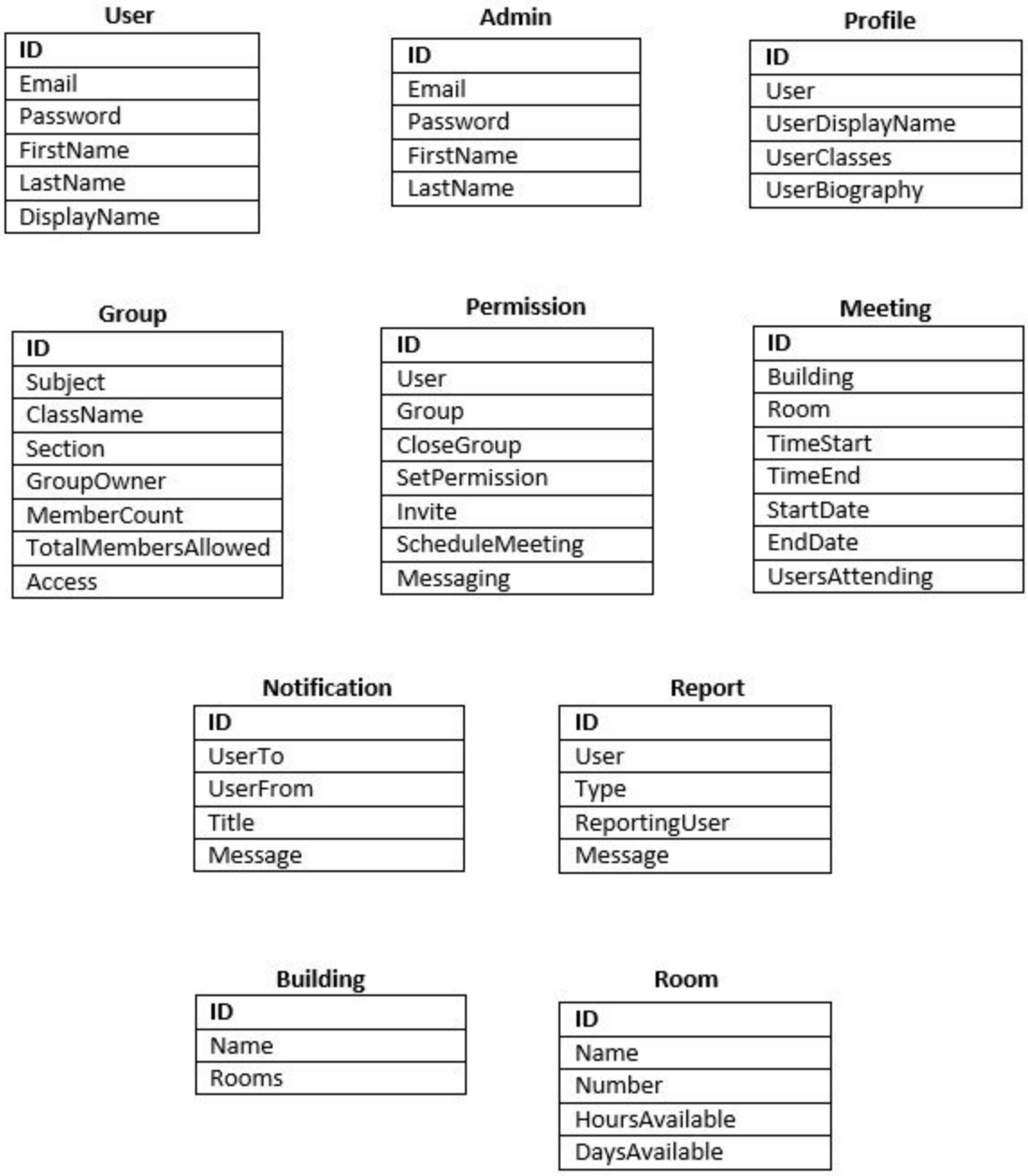


Figure 4. Admin Views

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

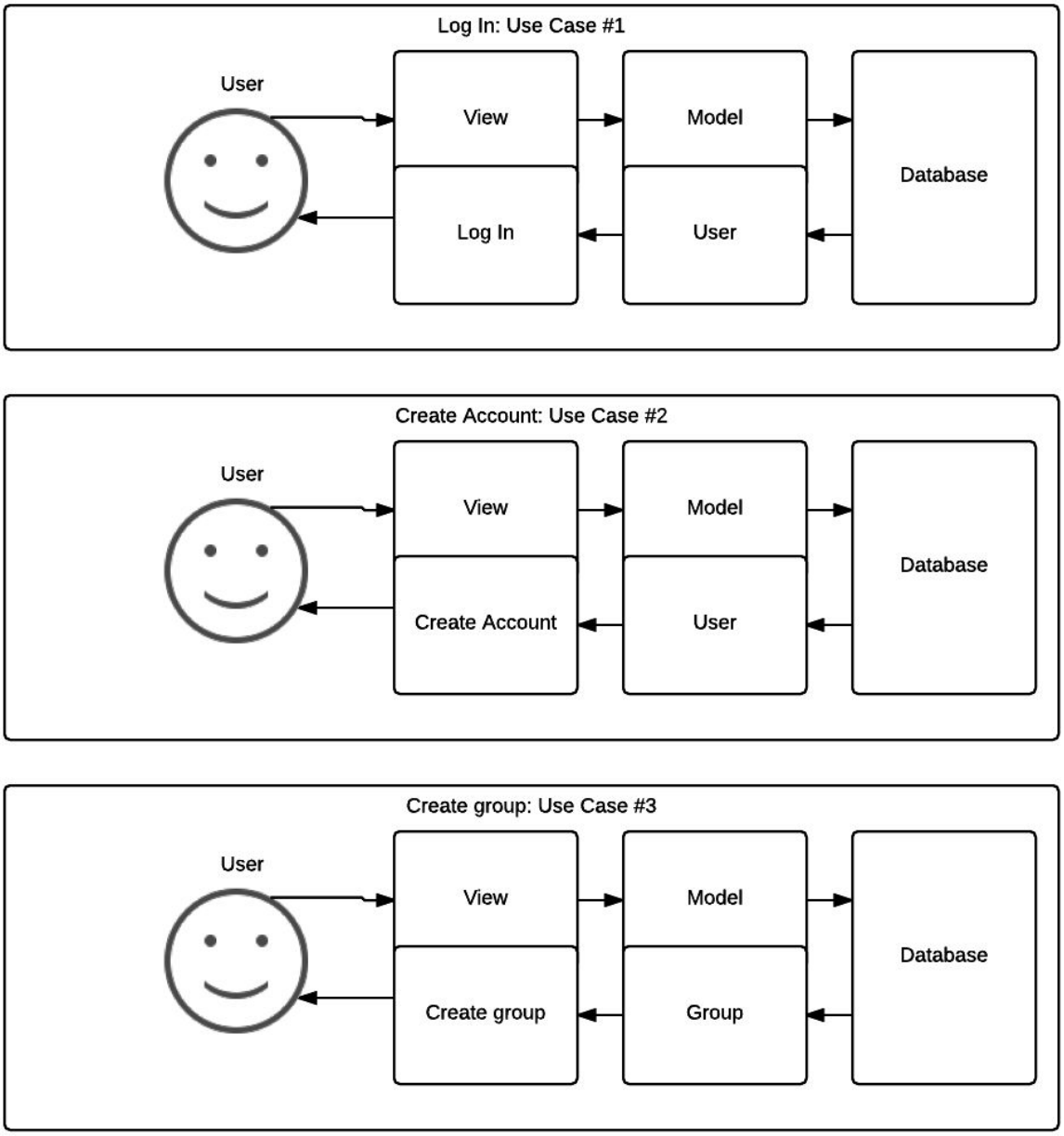
All application data will be kept in a MariaDB database called MSGP. We will be using the CherryPy Python web framework to create the database. The database will store data about each user and admin, group information, permissions, and the data of each notification and report created.

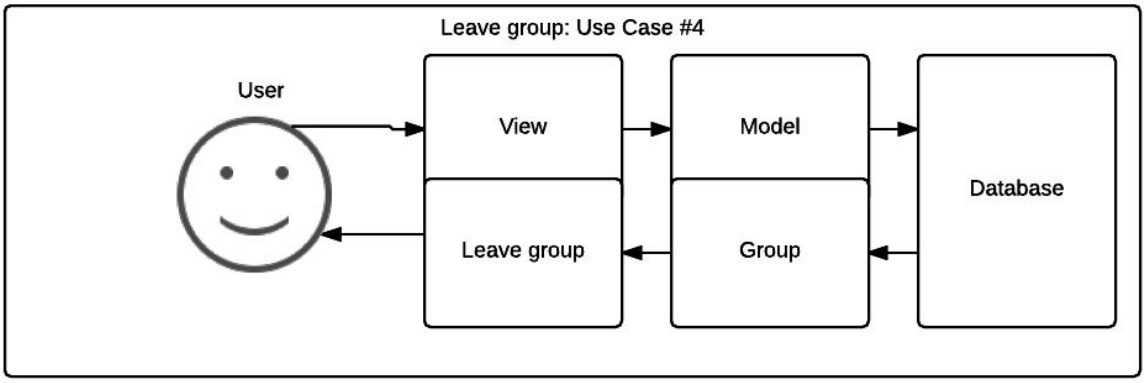


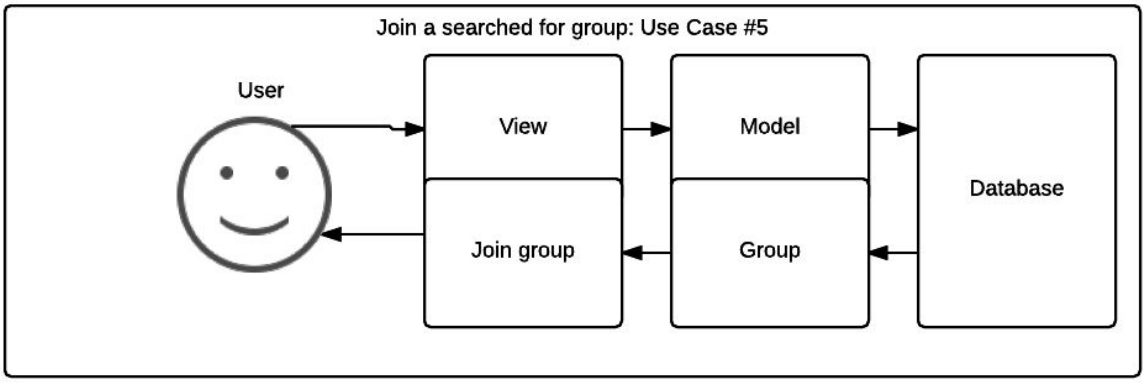
5. Requirements Matrix

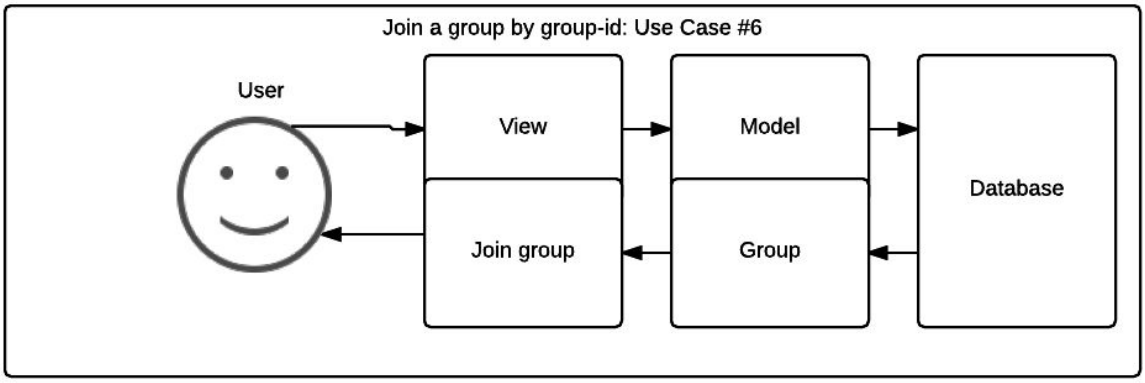
Please refer to the System Requirements Specification for details regarding the corresponding use cases.

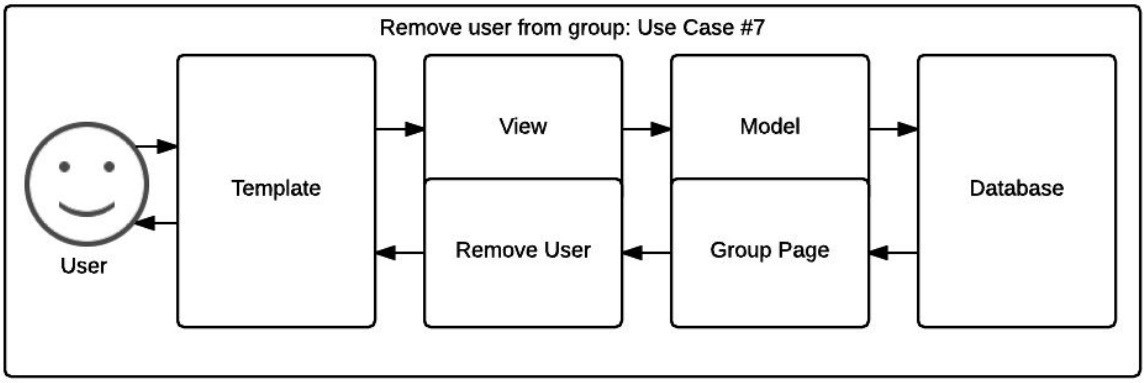
MyStudyGroupPlanner Requirements Matrix

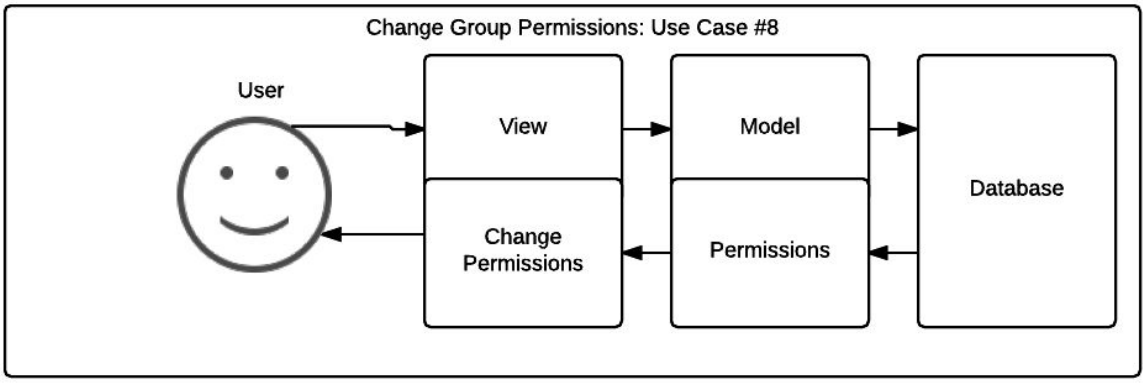


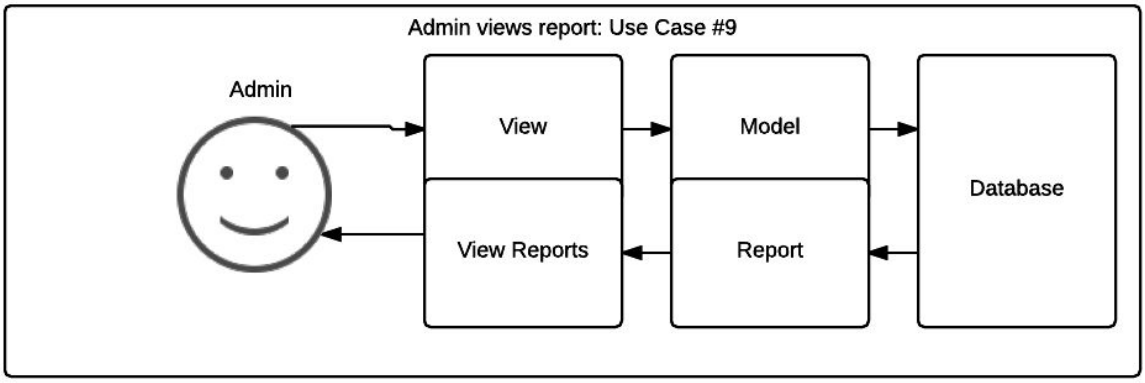

Leave group: Use Case #4
User
View
Model
Database
Leave group
Group

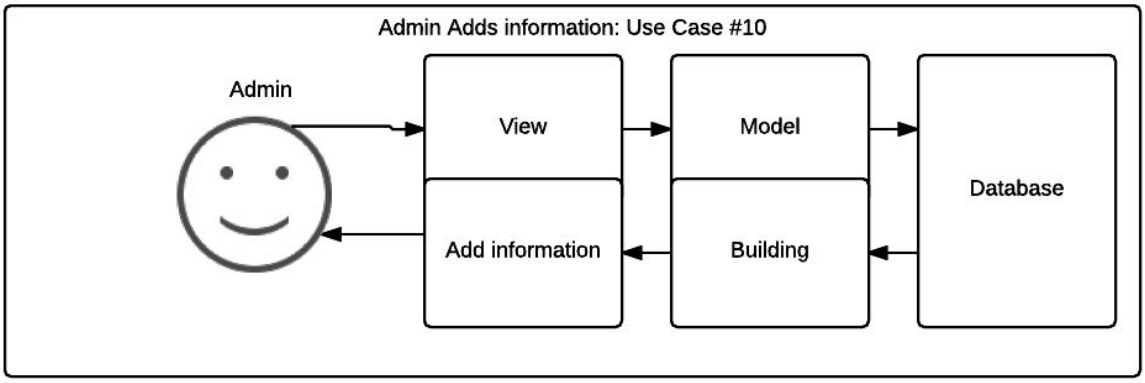

Join a searched for group: Use Case #5
User
View
Model
Database
Join group
Group

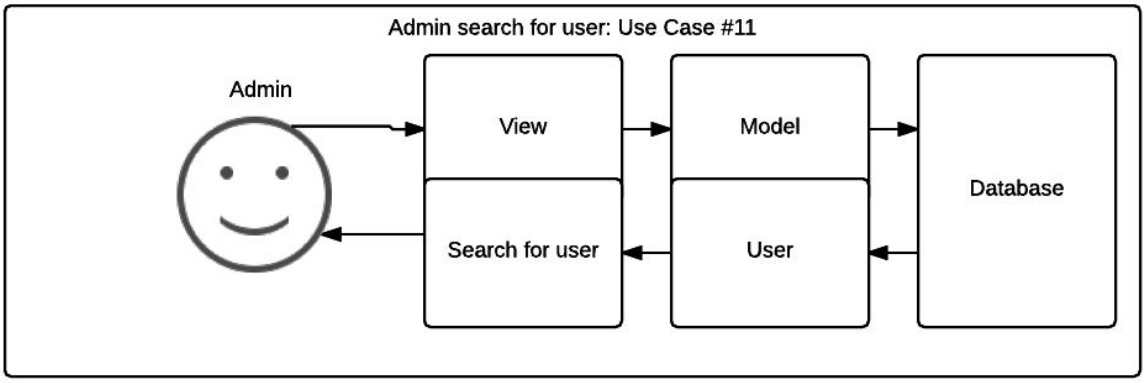

Join a group by group-id: Use Case #6
User
View
Model
Database
Join group
Group

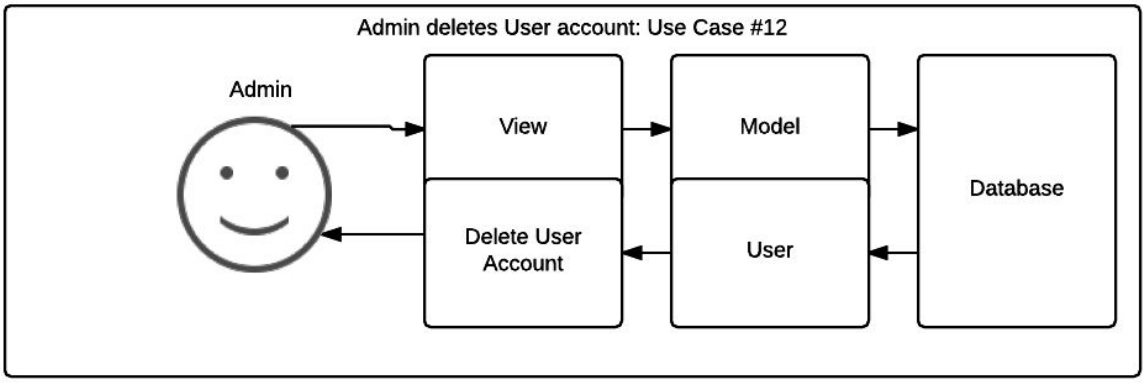

Remove user from group: Use Case #7
Template
View
Model
Database
User
Remove User
Group Page

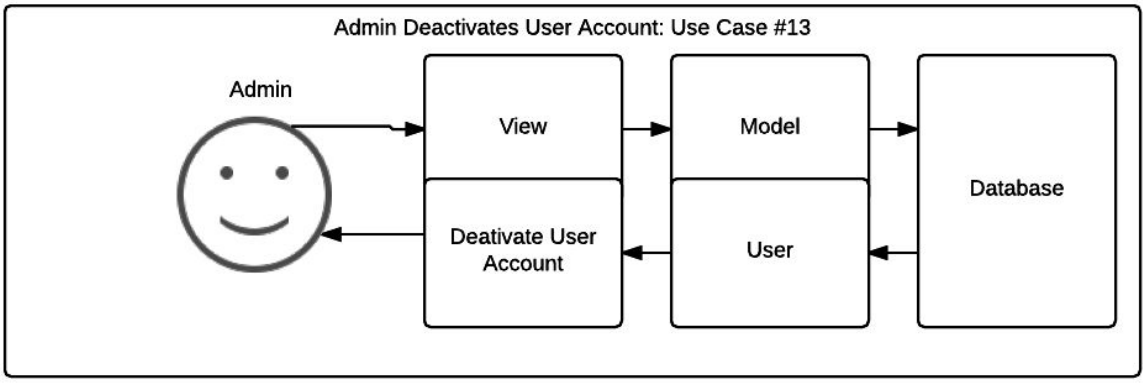

Change Group Permissions: Use Case #8
User
View
Model
Database
Change Permissions
Permissions

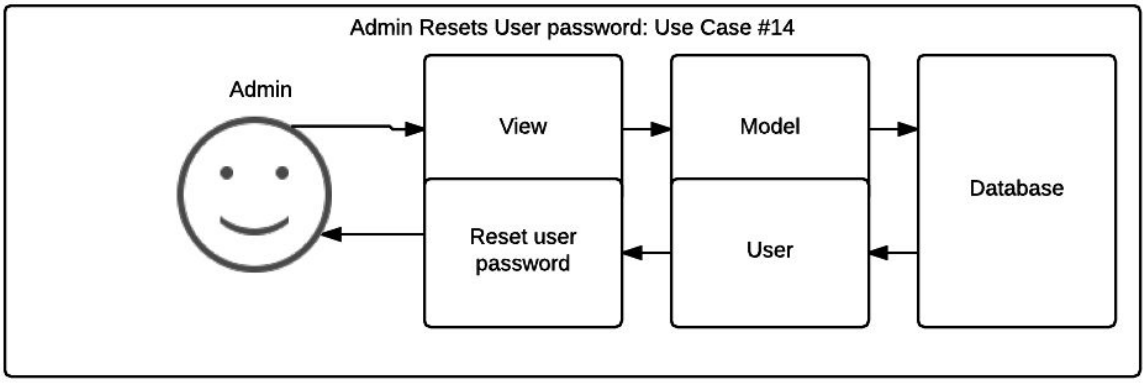

Admin views report: Use Case #9
Admin
View
Model
Database
View Reports
Report

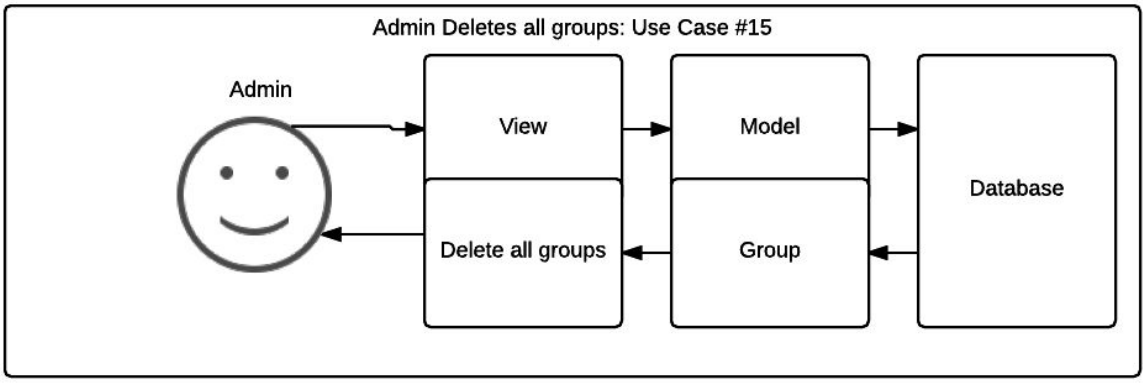

Admin Adds information: Use Case #10
Admin
View
Model
Database
Add information
Building

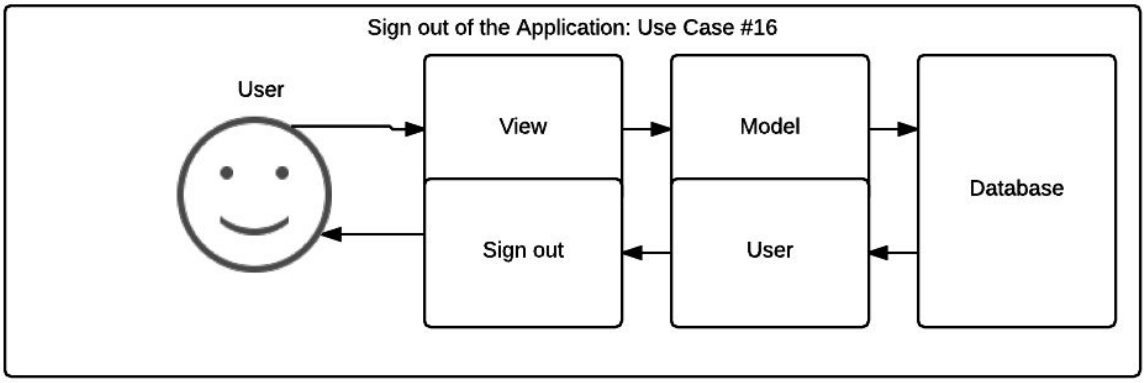

Admin search for user: Use Case #11
Admin
View
Model
Database
Search for user
User

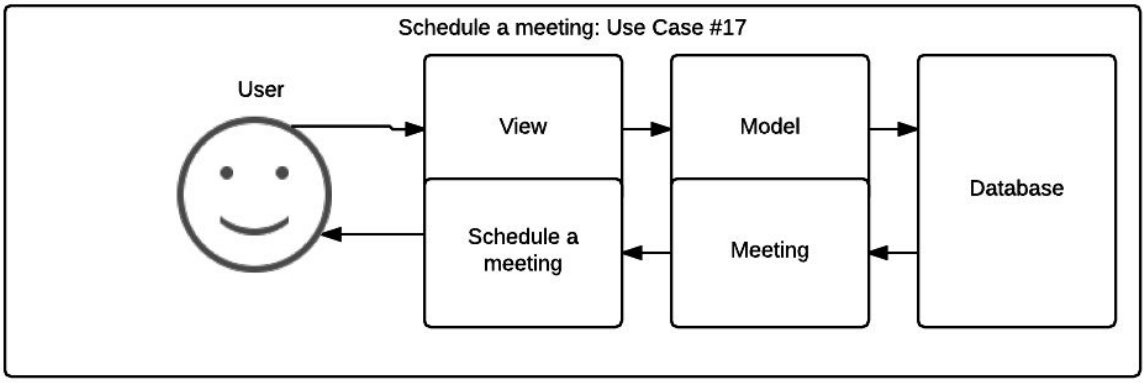

Admin deletes User account: Use Case #12
Admin
View
Model
Database
Delete User Account
User

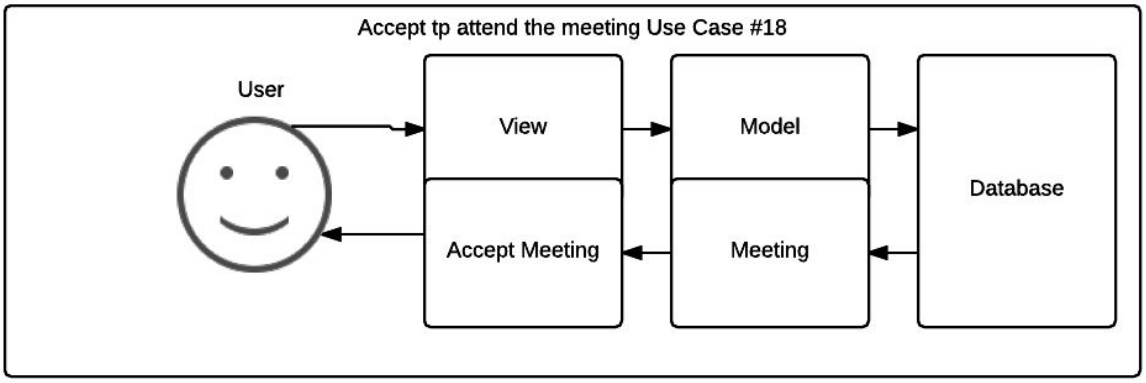

Admin Deactivates User Account: Use Case #13
Admin
View
Model
Database
Deativate User Account
User

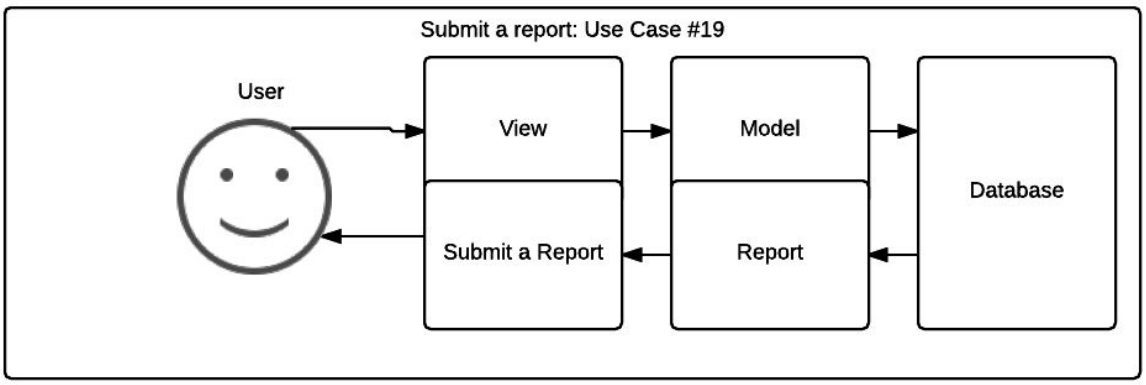

Admin Resets User password: Use Case #14
Admin
View
Model
Database
Reset user password
User

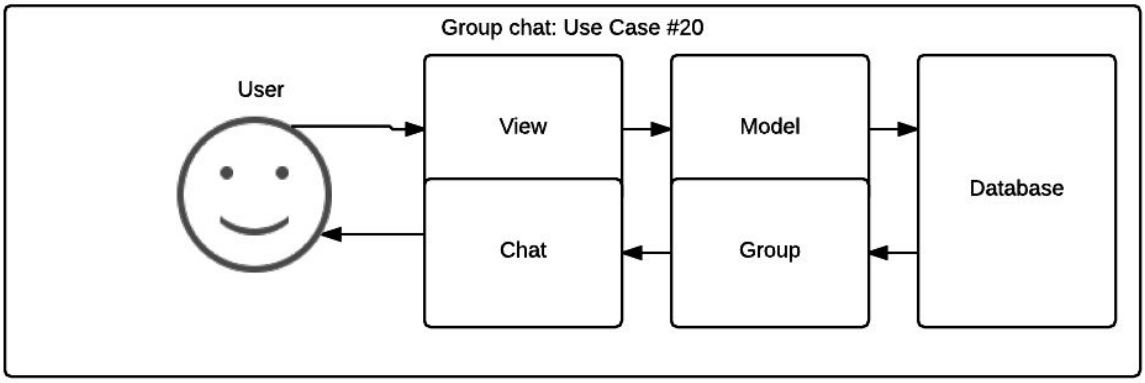

Admin Deletes all groups: Use Case #15
Admin
View
Model
Database
Delete all groups
Group

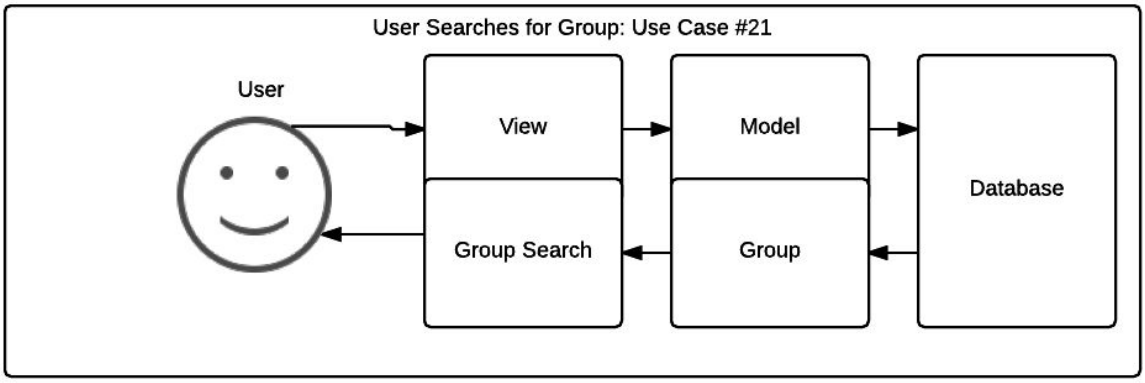

Sign out of the Application: Use Case #16
User
View
Model
Database
Sign out
User

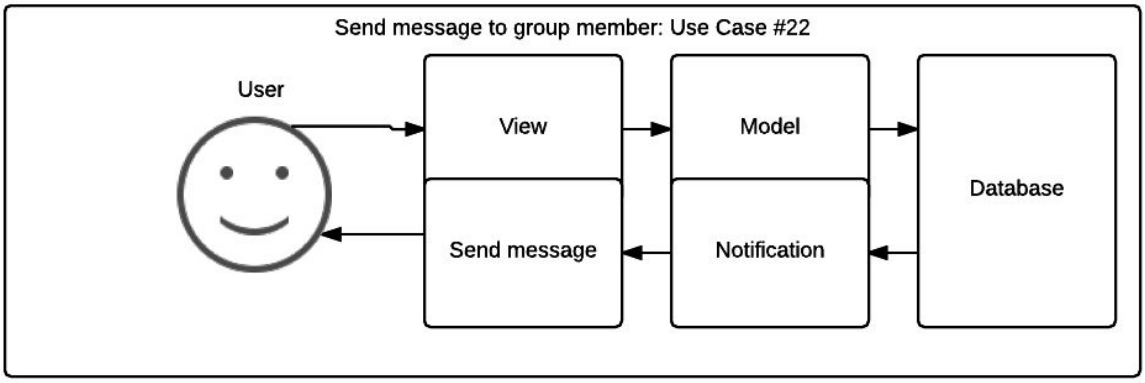

Schedule a meeting: Use Case #17
User
View
Model
Database
Schedule a meeting
Meeting

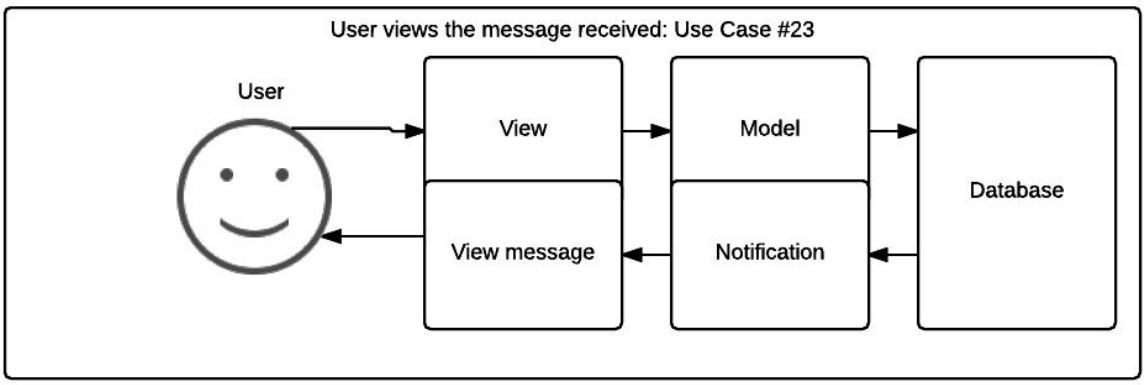

Accept tp attend the meeting Use Case #18
User
View
Model
Database
Accept Meeting
Meeting

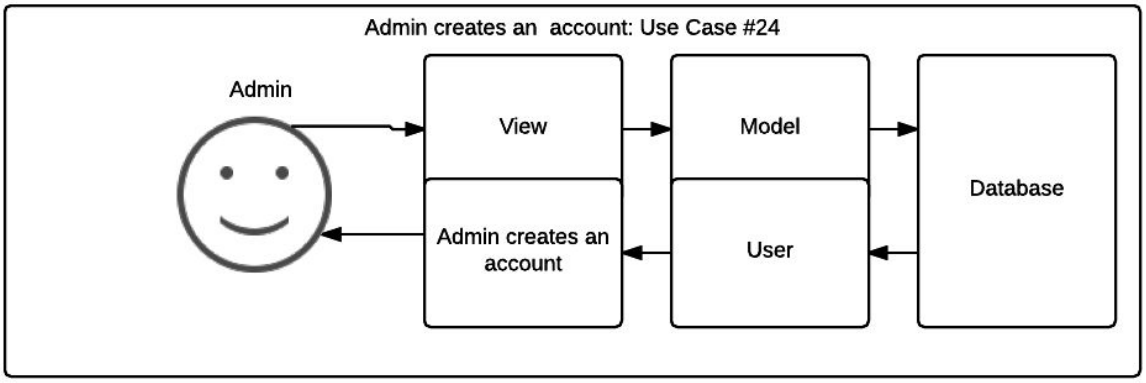

Submit a report: Use Case #19
User
View
Model
Database
Submit a Report
Report

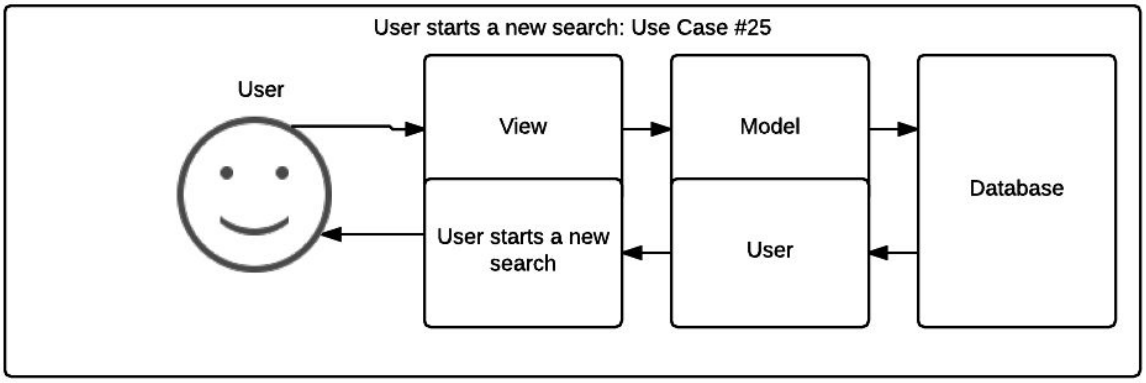

Group chat: Use Case #20
User
View
Model
Database
Chat
Group

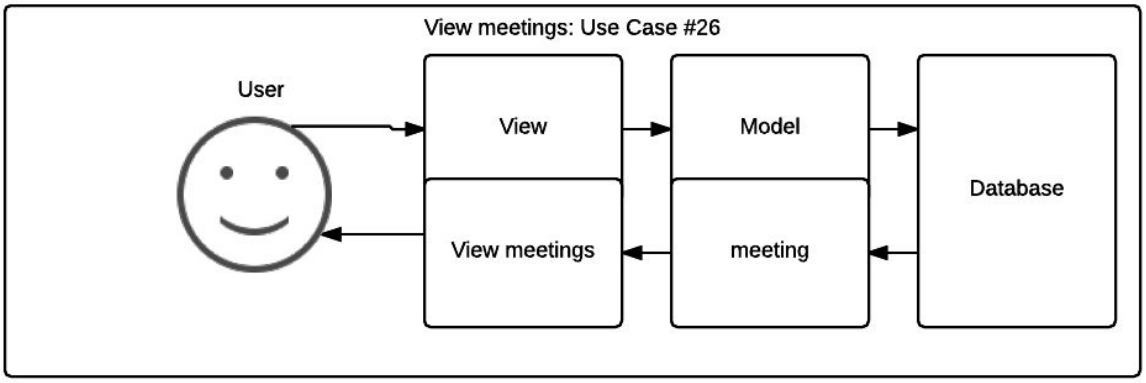

User Searches for Group: Use Case #21
User
View
Model
Database
Group Search
Group

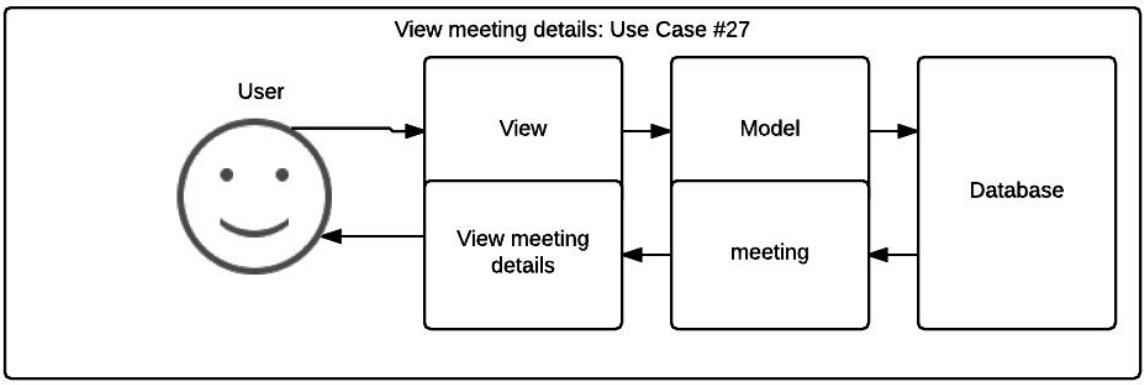

Send message to group member: Use Case #22
User
View
Model
Database
Send message
Notification

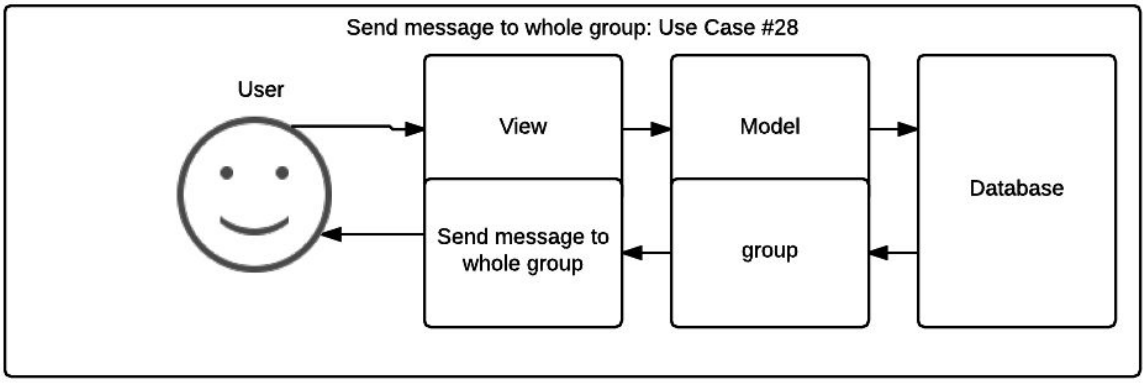

User views the message received: Use Case #23
User
View
Model
Database
View message
Notification

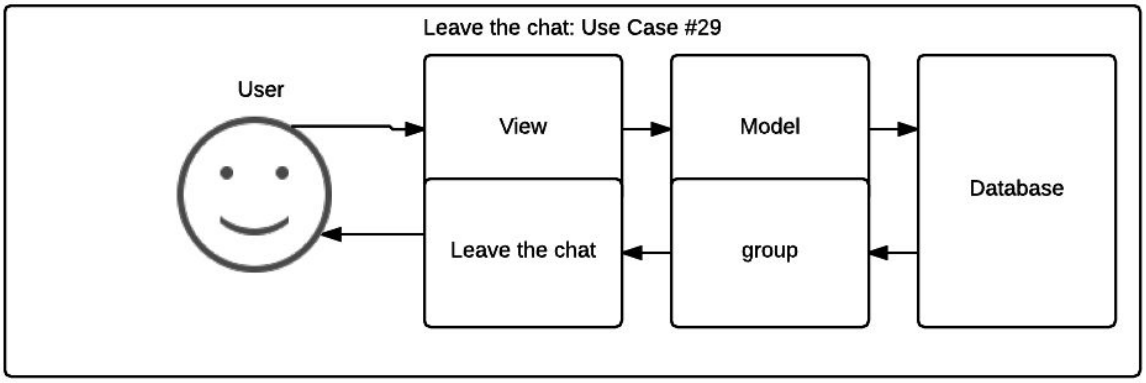

Admin creates an account: Use Case #24
Admin
View
Model
Database
Admin creates an account
User

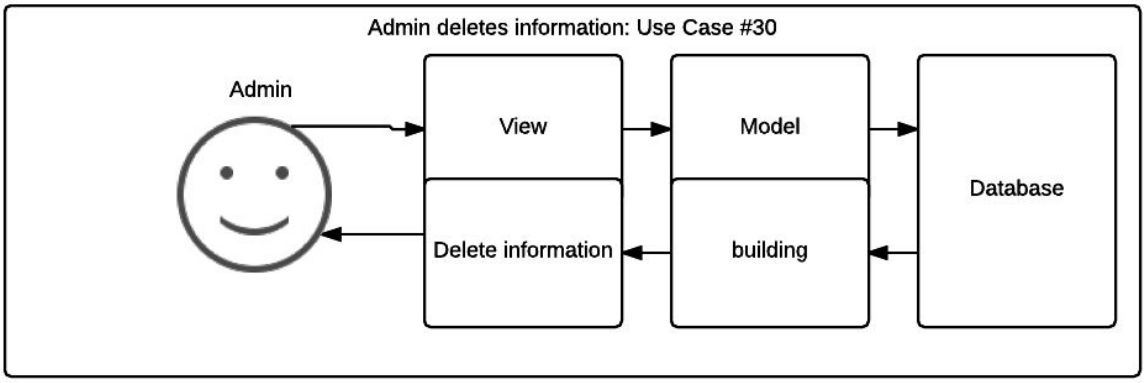

User starts a new search: Use Case #25
User
View
Model
Database
User starts a new search
User

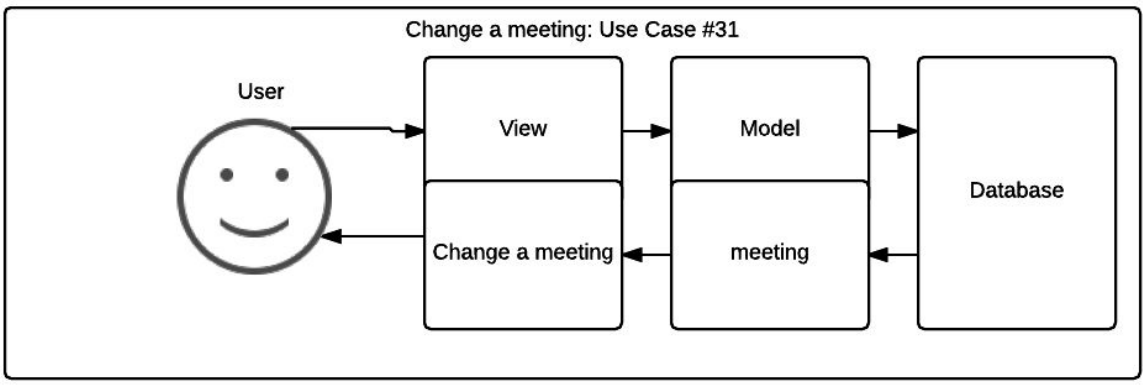

View meetings: Use Case #26
User
View
Model
Database
View meetings
meeting

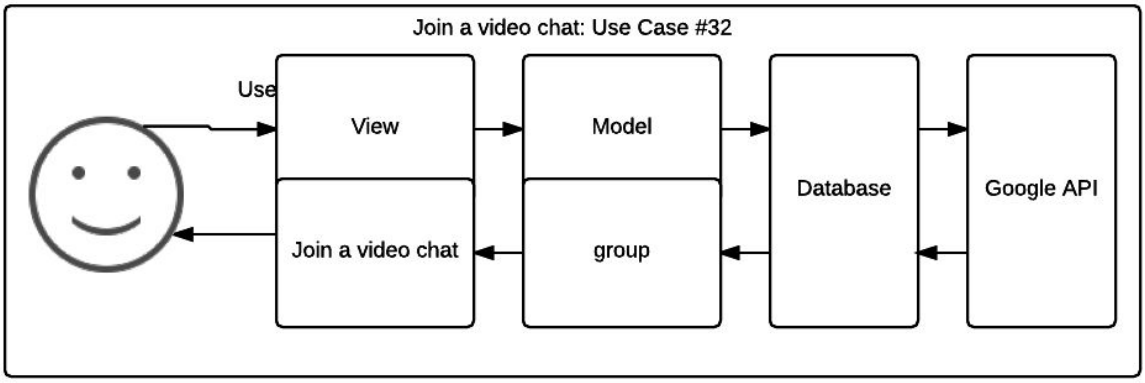

View meeting details: Use Case #27
User
View
Model
Database
View meeting details
meeting

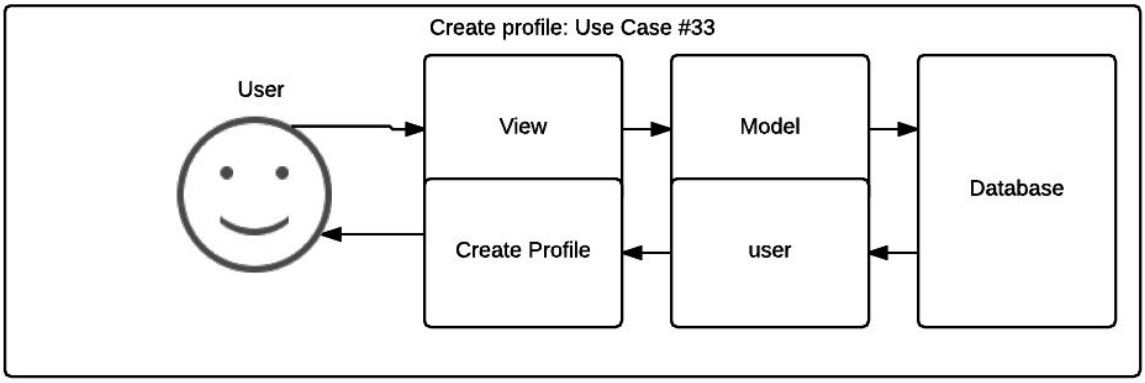

Send message to whole group: Use Case #28
User
View
Model
Database
Send message to whole group
group

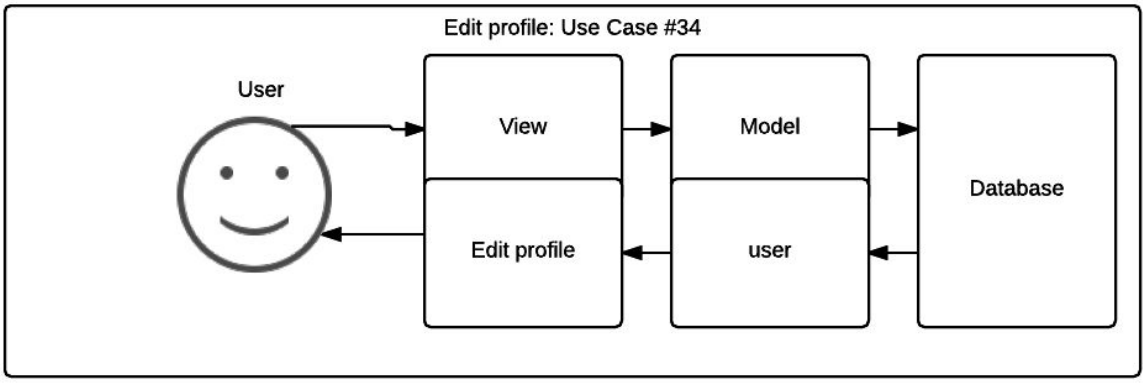

Leave the chat: Use Case #29
User
View
Model
Database
Leave the chat
group

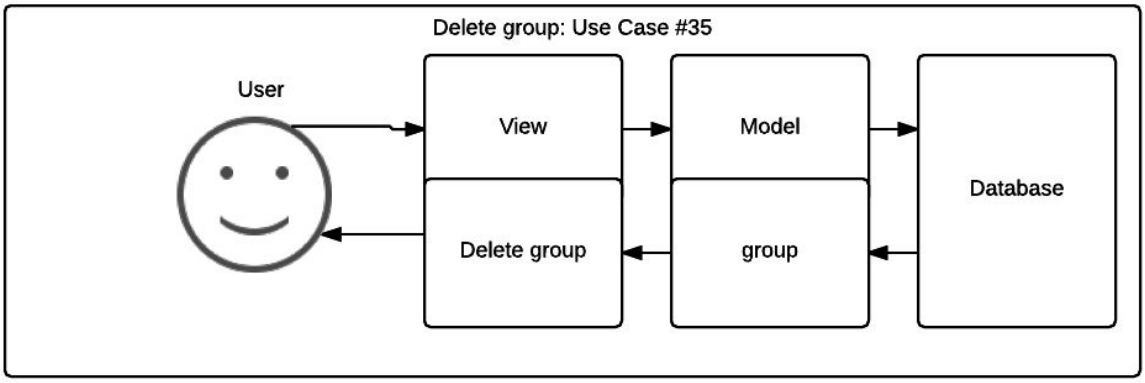

Admin deletes information: Use Case #30
Admin
View
Model
Database
Delete information
building

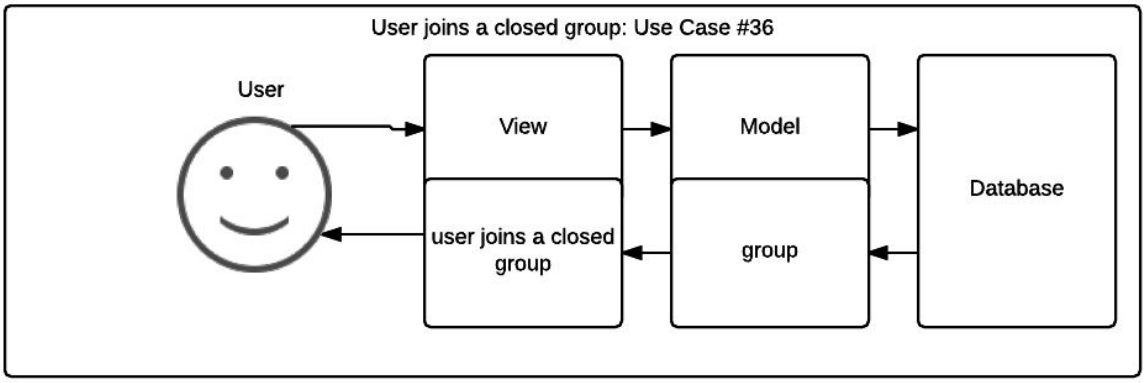

Change a meeting: Use Case #31
User
View
Model
Database
Change a meeting
meeting

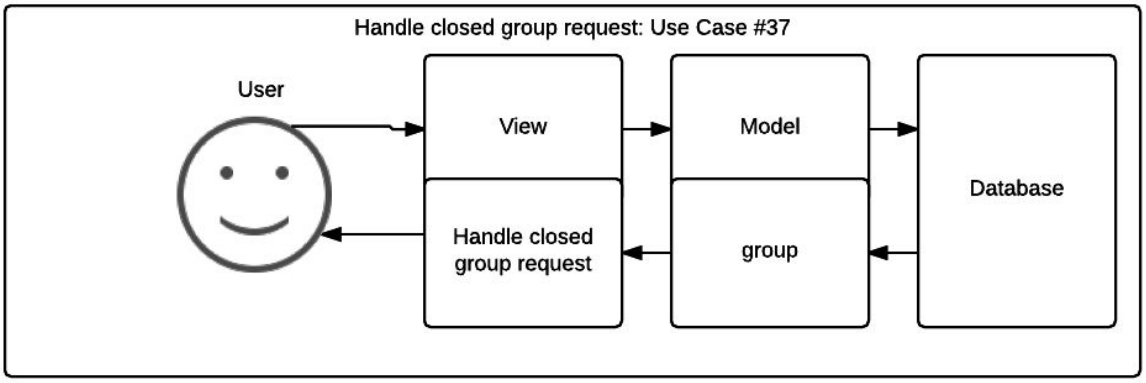

Join a video chat: Use Case #32
Use
View
Model
Database
Google API
Join a video chat
group


Create profile: Use Case #33
User
View
Model
Database
Create Profile
user


Edit profile: Use Case #34
User
View
Model
Database
Edit profile
user


Delete group: Use Case #35
User
View
Model
Database
Delete group
group


User joins a closed group: Use Case #36
User
View
Model
Database
user joins a closed group
group


Handle closed group request: Use Case #37
User
View
Model
Database
Handle closed group request
group

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	Date
_____	
Signature	
_____	_____
Print Name	Date
_____	
Signature	
_____	_____
Print Name	Date
_____	
Signature	
_____	_____
Print Name	Date
_____	
Signature	
_____	_____
Print Name	Date
_____	
Signature	

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**