

# **Cornell Peer Group Finder**

Usability Testing (Round 2) Feedback and Analysis

---

## **Design Team:**

Abrams, Gabriel

Huang, Andy

Ruengsakulrach, Natchaphon

Wei, Xing

Zhang, Jing

Zhang, Xiangyu

Zhong, Peimin

Zhou, Tao

**November 2015**

# Overview

## Purpose and Scope

The design team will outline and create a Peer Learning Group Finder application for Mann Library visitors and staff members. The purpose of this responsive web application is to encourage group learning and team collaboration among students in Mann library. It will also provide librarians with useful room usage and meeting statistics for future library service improvements.

The system will consist of four primary functional components:

1. Admin Interface (Statistics View) allows admins to view useful room usage and meeting statistics through an admin interface.
2. Meeting Creation allows a user to create a meeting by providing meeting's title, location, time, type, and contact.
3. Meeting Editing allows user(s) to cancel the meeting or edit all basic and advanced information of the meeting.
4. Meeting Search allows users to search for meetings. The team will provide API and system documentations, which include system requirement, database ER diagram, a list of available functionalities, and RESTful API specification sheet. The client can extend the software system in the future using the API and documentations.

## Objectives and Goals

The main objective of this project is to help individuals in the Mann library to advertise their meetings or to help them to discover other meetings happening currently or in the future. The second objective is to provide the library staff with an administration interface to generate reports and manage meetings.

The project will be a success if the team is able to create an intuitive responsive web application that efficiently, securely, and intuitively implements all four of the primary functional components listed above.

# Project Requirements

To our understanding, the project requirements may be broken down into the following categories: functional requirements, usability requirements, and non-functional requirements.

## Functional Requirements

This section of the requirements outline will account for functional requirements including data management, user interface, and key functionality requirements.

As previously mentioned, the system will be built upon the understanding that there will be four key functional components: Admin Interface, Meeting Creation, Meeting Edit/Cancel, Meeting View/Search. These functions must be implemented in an intuitive, responsive, and bug free manner. To test the functionality of these components, we will run all of these functions through three rounds of usability testing, which we outline further in a later section of this document.

It should be carefully noted that usability testing forms the backbone of our development process. And therefore, we will take full advantage of the benefits of the iterative refinement method while implementing these primary functional components.

Additionally, we are carefully designing a RESTful API and documentation that meets the requirements listed below. In the detailed summary of requirements below, we have listed the requirements that we are presenting at the usability testing session. If a feature is not listed below, it will be implemented in a later iteration cycle. These features below will be tested and put up for review and modification.

### 1. Meeting Creation:

*\* denotes a required field*

**Title\*** A text entry field containing the title of the meeting

**Location\*** A popup location selection tool. We will use a hierarchical definition system as follows: (floor → room)

**Time\*** A date and time selection pane for selecting a range of time for the meeting

**Meetingtype\*** A dropdown selection field to select a meeting type (project meeting, etc.)

**Contact\*** A text entry field containing the private contact email of the creator

**Meeting Audience:** User chooses whether the meeting is “open to everyone” or “invitation only”

**Description/Details:** A text area for giving additional meeting details. Default is none.

Although the following feature was previously removed after request from the client, we are now reconsidering a feature similar to this one. This is due to a lot of careful feedback from user testers and discussion with the client.

**TeamMembers** A text field for entering team member emails. Default is none. (This field is tentative)

Functional Features that have been implemented since the previous milestone:

**Reserve-only** If a meeting creator attempts to schedule a meeting in a room and time slot that is already taken in one of the library’s previously created and populated meeting scheduling systems, the creator will be prompted with information about the conflicting meeting. During the previous iteration, we implemented one out of the three library reservation systems.

### 2. Meeting Edit:

*Credential Checking System:*

**Authentication** All meetings will require a password for editing, unless the edit request is sent by an admin user. Meeting passwords are generated upon meeting creation and emailed to the meeting creator email. After discussion with the client, we developed an intuitive password generation system that combines memorable words with simple numbers.

**PasswordReset** The system will send a password reset email with a randomly generated password to the creator email.

*Editing Options:*

**AvailableFields** All meeting fields will be editable except the meeting creator contact field. The meeting creator contact field will not be shown due to the fact that it is private information.

**MeetingCancellation** Remove the meeting from the calendar

### 3. Meeting Search:

*Search Fields:*

**Automatically Searched:** The search field will automatically search meeting title and description.

The following fields are still being implemented and are scheduled for the next iteration:

**MeetingType** Meeting type filter will show only meetings of a certain type

**Time** Allows specification of date and time range to search

**Location** Allows hierarchical searching of locations (allows searching an entire floor or a single room)

**4. Meeting List:** function to view a list of meetings in summary form. This will be used on the front page and as a way to show results of a search query.

*Visible Data:*

**Title** Meeting title

**Location** Shows floor and room (if selected)

**Time** Time range of meeting

*Filter and Sorting:*

**Number of Meetings** users may choose to view either 5, 10, or 25 current meetings

**Current Time Range** users may choose to view today's meetings or meetings from this week

We will be implementing time, date, and other filters to the search functionality for further customization of meeting views.

*Related API Functionality:*

**currentMeetings** Returns meetings shown on the meeting list page

**searchMeetings** Returns meetings based on given search options (all search options specified in the calendar search page will be available in the API)

**getMeetingInfo** Returns all data from given meeting.

**changeMeeting** Allows editing of meeting information others Other API functionality will be available for statistics and data management

*We added the following functions during the previous iteration:*

**roomInfo** Returns meeting information (picture, description, reservation system, etc if available)

**allRooms** Returns a hierarchical list of all floors and rooms with their pictures and descriptions

## Usability Requirements

We understand that usability is more than the program interface. Therefore, we focus on the entire process as the user experience. With this scope and perspective, we have identified many important usability requirements that we will be testing throughout our iterations of usability testing.

After spending time in our usability testing sessions, we have had a lot of feedback and have updated the usability requirements listed below.

### 1. Intuitiveness

All implemented features should be available to all different types of users. Every user must easily and intuitively be able to perform all of the principal program features. For instance, everyone should be able to easily create a meeting without worrying about ambiguities and confusing information.

Task	Requirement
<b>Meeting Creation</b>	User should be able to navigate creation form without tutorials, questions, or difficulties. Users should be comfortable with the web application. They should trust the interface and the security of the application.
<b>Meeting Search</b>	User should easily understand the use of filters and text to perform simple queries. Users should also be able to find specific meetings.
<b>Meeting View</b>	User should be able to open the view feature within 5s on a first attempt. User should also be able to identify all key meeting information on view page.
<b>Meeting Edit</b>	User should be able to edit a specific information field in a meeting on a first attempt. Additionally, the user should understand how to commit changes.
<b>Authentication Interface</b>	User should be able to retrieve meeting passwords and request a password via email. Users should be able to remember generated passwords.

## 2. Speed and Responsiveness

Our responsive web application is designed to be intuitive and lightweight. Thus, users must be able to perform program functions quickly without noticeable lag. For instance, a simple meeting search should return results quickly, and after creating a meeting, instantaneous confirmation should be displayed to the user.

Task	Requirement	Time
<b>Meeting Creation</b>	Creation confirmation popup	1s
<b>Meeting Creation</b>	Confirmation email	15m
<b>Meeting Search</b>	Return search results	1s
<b>Meeting View</b>	Load meeting from database	0.1s
<b>Meeting Edit</b>	Load meeting edit data	0.1s
<b>Meeting Edit</b>	Creation confirmation popup	1s
<b>Meeting Edit</b>	Confirmation email	15m
<b>Authentication Interface</b>	Authenticate	1s

## 3. Security

To ensure that the user's private information is preserved and to protect the internal systems (web application and database) from illegal or malicious inputs, the following security requirements will have to be fulfilled:

Security Feature	Description
<b>Limit size of user's text input</b>	Use text input should be limited in length to prevent the database from crashing

Security Feature	Description
<b>Store non-plain-text password</b>	MD5 hash of password will be stored in the database
<b>Validate user's input</b>	Inputs such as meeting's time range and email address must be validated prior to being stored in the database. For example, meeting's end time should not be before meeting's start time and email should be in a recognizable format <u>XXX.XXX@XXX.XXX</u>
<b>Filter spam email</b>	The system shall prevent Denial of Service spam attacks as well as prevent the system from spam mailing users.
<b>Keep creator email secure and private</b>	Do not show on meeting view or edit page
<b>Standard web security</b>	The system should mandate HTTPS
<b>Other security measures</b>	Implement many modules that protect against SQL injection, third party hacks, and other malicious behavior.

## Non-functional Requirements

### Compatibility

The front-end application must render the user interface properly when accessed via desktop browsers, laptop browsers and mobile device browsers. The backend program must function properly when moved into Mann Library's production servers.

We will specifically be focusing on testing the web application on many different browsers and devices. Due to the popularity of many devices and browsers, we will be considering the following list in our testing and design:

**Browsers:** Google Chrome, Safari, Firefox, IE, Opera, Various Mobile Browsers (Safari, IE, ...)

**Devices:** Windows Desktops and Laptops, Mac Desktops and Laptops, iPhone, Windows Mobile, Android, iPad, devices that do not support HTML5 (test javascript support)

We understand that this may particularly affect screen and text size. As a responsive web application, we will be working hard to make dynamically sized and formatted pages to best suit the browser and device of the users.

### Maintainability and Extensibility

The backend program must support RESTful APIs in order to enable connection with other Mann Library applications and to allow the entire system to be enhanced after the initial release.

After discussion with the client, we will be creating a fully functioning responsive web application on AWS and will be handing the project to Nick at the end of the development cycle so that he can deploy the program on library servers.

## Current Systems

There is currently no Peer Learning Group Finder system available at Mann Library.

## Use Case Diagrams

We will have three different user roles. Each of these roles have different functions that they may take advantage of and therefore different use case diagrams.

Note that detailed use case specifications are listed in the following section: Use Case Specifications. A brief outline of the three user roles is found below:

### Admin:

Admin accounts are allotted by the client. These users will have the following capabilities:

- Edit any event without a password
- Change locked data fields on meetings (ex: will be able to modify a meeting's creator email)
- Remove any event from the system
- Access more detailed data from the database (ex: statistics)

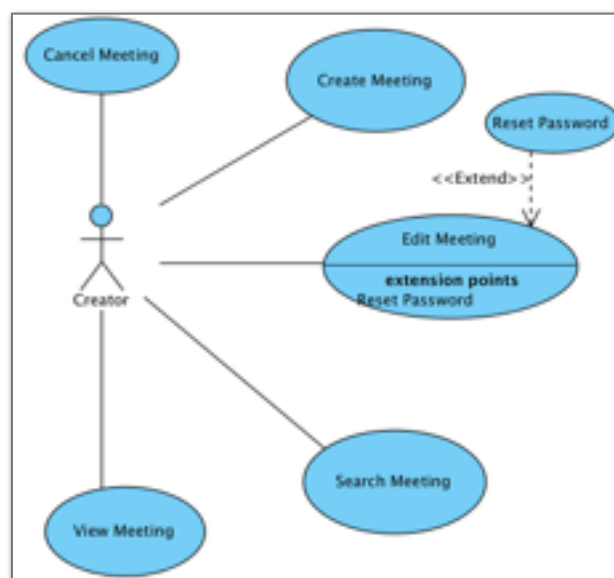
Additionally, this role will have all of the capabilities of the other two roles: Creator and Visitor.

### Creator:

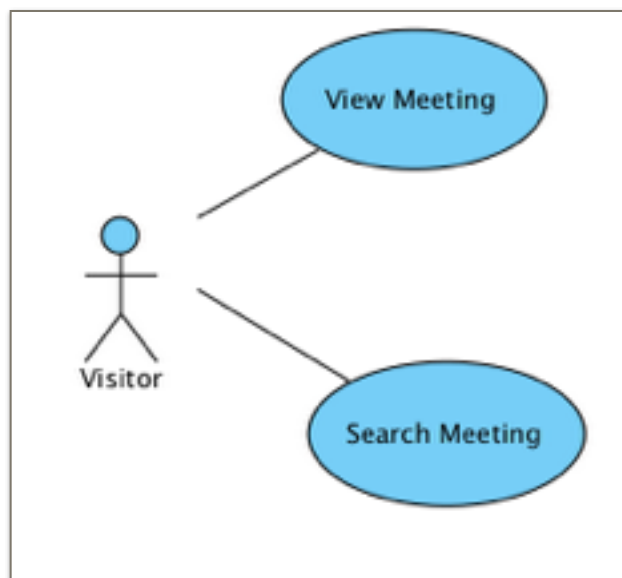
A user is promoted to Creator for the meeting(s) that they create. Thus, a visitor is promoted to Creator when they create a meeting.

- Reset the password of their meeting(s) by requesting a password reset link. Because the password will be sent to the creator email (which is unchangeable without Admin access), only creators or admins may reset the password of a meeting.

Additionally, this role will have all of the capabilities of Visitors.



Meeting Creator Use Case Diagram

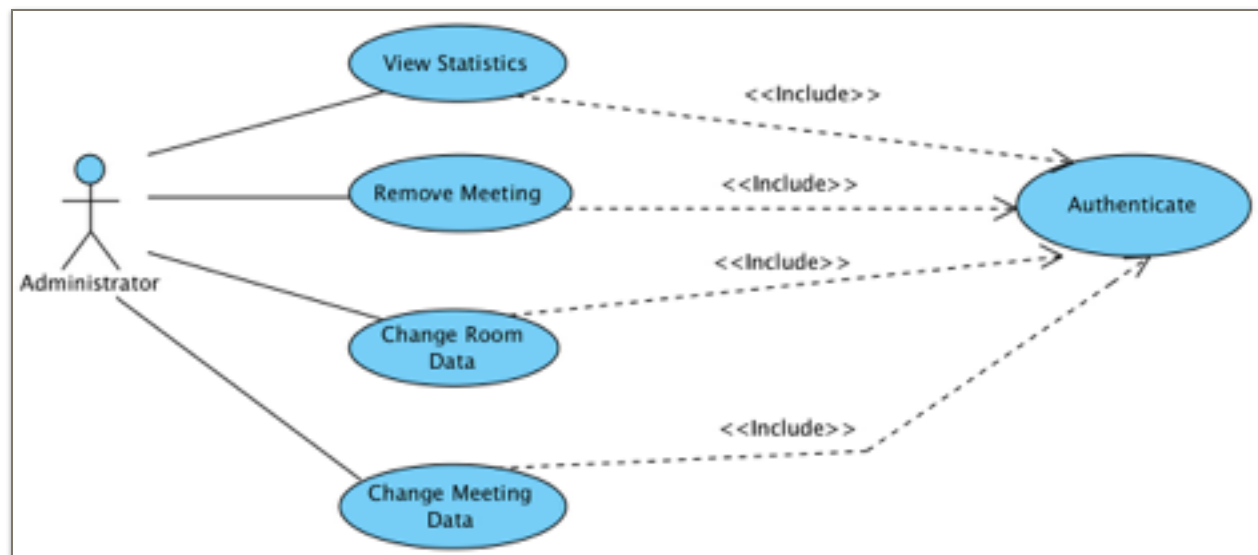


Visitor Use Case Diagram

**Visitor:**

All other users are automatically Visitors. Also, it should be noted that Creators have Visitor privileges when viewing meetings that they did not create.

- View/search for meetings
- Create meetings (they will be promoted to Creator for that meeting)
- Edit meetings with a password
- Cancel meetings with a password



**Admin Use Case Diagram**

## Use Case Specifications

**Name of Use Case:** Create Meeting

**Summary:** The creator creates a meeting by filling out a meeting creation form.

**Preconditions:** The Cornell group finder user interface must be open in a browser meeting the system requirements.

**Trigger:** The creator clicks create meeting button in the header of the web application.

**Flow of events:**

1. The creator enters all the required fields of the form and can choose to fill out the optional fields as well.
2. The creator submits the meeting creation form.
3. The server inserts the new meeting information into the database and a message indicating the successful completion of meeting creation is displayed on the browser.
4. An email confirmation is sent to the creator's email address.

**Name of Use Case:** View Meeting

**Summary:** The creator/visitor views the list of meetings.

**Preconditions:**

1. The Cornell group finder user interface must be open in a browser meeting the system requirements.



2. There are meeting that have already been created.

**Trigger:** The creator/visitor is on the homepage.

**Flow of events:**

1. The creator/visitor tailors the meeting list by clicking the option buttons right on top of the list which allows user to decide what number of meetings to be shown within a single page during what time slots.
2. The creator/visitor clicks the highlighted link to enter the viewing section of each particular meeting.
3. The creator/visitor can choose to modify the meeting by clicking the “Modify?” button.
4. The creator/visitor can choose to share the meeting information with Facebook’s friends by clicking the Facebook share button.

**Name of Use Case:** Edit Meeting

**Summary:** The creator edits the meeting by updating the meeting creation form.

**Preconditions:**

1. The Cornell group finder user interface must be open in a browser meeting the system requirements.
2. The meeting that the creator wants to edit must be created previously.
3. For meetings that are editable, a password is required for creator authentication.

**Trigger:**

1. The creator clicks the the edit button of the navigation bar at the very top of the homepage.
2. The creator clicks the highlighted link corresponding to each particular meeting in the meeting list.

**Flow of events:**

1. If the creator uses the navigation bar, he or she will be prompted with a password to edit meeting.
2. The creator enters the password and changes any field according to the specific need.
3. The creator clicked the “Update” button to submit the new form.
4. The server updates the database with the newly-updated meeting information and a message indicating the completion of a successful update is displayed on the browser.

**Alternative path:**

1. If the creator clicks the link instead, he or she subsequently clicks the “Modify?” button and then enter the required password for editing the meeting information.
2. In the edit page, the creator changes any field according to the specific need.
3. The creator clicks the “Update” button to submit the new form.

**Name of Use Case:** Reset Password

**Summary:** The creator tries to reset the password in order to access the edit page.

**Preconditions:**

1. The Cornell group finder user interface must be open in a browser meeting the system requirements.
2. The creator must be the creator since an email regarding password reset will be sent to the creator’s email.

**Trigger:**

The creator clicks the “Forgot?” button below the password input bar.

**Flow of events:**

1. The server randomly generates a new password and sends it directly to the creator via email.

2. The creator is prompted with a message indicating that an email has been sent to his or her email address.

**Name of Use Case:** Search Meeting

**Summary:** The visitor/creator searches the information of a particular type of meetings by typing in keywords and using filters.

**Preconditions:**

1. The Cornell group finder user interface must be open in a browser meeting the system requirements.
2. The meetings that the visitor/creator wants to search for must be existent.

**Trigger:**

The visitor/creator clicks the “Search Section” button of the navigation bar at the very top of the homepage.

**Flow of events:**

1. The visitor/creator types in search keywords and uses filters to customize search experience.
2. The visitor/creator selects the number of pages to be shown within a single page.

**Name of Use Case:** Authenticate

**Summary:** The administrator passes authentication by providing the correct username and password.

**Preconditions:**

1. The Cornell group finder admin interface must be open in a browser meeting the system requirements.
2. The person who access the admin interface must be an administrator of the system.

**Trigger:**

The administrator has typed in the login information needed to access the system.

**Flow of events:**

1. The server validates the administrator’s input and send a message back to the browser indicating the success or failure of the login.

**Name of Use Case:** View Statistics

**Summary:** The administrator accesses detailed data from the system’s database.

**Preconditions:**

1. The Cornell group finder admin interface must be open in a browser meeting the system requirements.
2. The user must pass through the administrator authentication.

**Trigger:**

The administrator inputs the search keywords and selects different filters.

**Flow of events:**

1. The administrator chooses different options to tailor the search result in terms of different aspects of data that is of particular interest.

**Name of Use Case:** Remove Meeting

**Summary:** The administrator removes any meeting from the system.

**Preconditions:**

1. The Cornell group finder admin interface must be open in a browser meeting the system requirements.
2. The user must pass through the administrator authentication.

**Trigger:**

The administrator selects the delete option provided by each meeting that has been created.

**Flow of events:**

1. On the server side, the entire record associated with the meeting is deleted from the database.
2. The server sends back a message to the browser indicating the success removing of the meeting.

**Name of Use Case:** Edit Meeting Data

**Summary:** The administrator is authorized to edit any meeting event even if it is locked and password protected.

**Preconditions:**

1. The Cornell group finder admin interface must be open in a browser meeting the system requirements.
2. The user must pass through the administrator authentication.

**Trigger:**

The administrator selects the highlighted link of the meeting that he or she wants to edit.

**Flow of events:**

1. The administrator enters the edit page, and is authorized to change any data fields.
2. The administrator confirms the modification by clicking the complete button.
3. The server updates the database and sends a message back to the browser indicating the success update of meeting information.

**Name of Use Case:** Change Room Data

**Summary:** The administrator is authorized to change the room data.

**Preconditions:**

The Cornell group finder admin interface must be open in a browser meeting the system requirements.

The user must pass through the administrator authentication.

**Trigger:**

The administrator selects the option of changing room data.

**Flow of events:**

1. In the room data edit page, the administrator can alter room data fields.
2. The administrator clicks the update button.
3. The server updates the database and sends a message back to the browser indicating the success update of room information.

## Scenario 1 - Create Meeting

**Purpose:** This scenario describes the process of entering information and posting meetings

**Individual:** A meeting creator (i.e. teaching assistants, senior students, group study leader)

**Equipment:** A computing device with internet browser and network connections

**Scenario:**

1. Meeting creator turns on computing device and opens browser
2. Meeting creator types the URL to access the Cornell Group Finder homepage
3. The backend server fetches the homepage layout sends it to the browser for display
4. Meeting creator clicks on the text-box that says "Click to Create a Meeting!"
5. The text-box will expand into a form with labels and fields
6. Meeting creator enters the following information into the form: meeting title, meeting contact email, meeting location, meeting location description, meeting start and end time, meeting type, meeting description and meeting privacy preference
7. After completing the form, the meeting creator clicks the "Done" button
8. The front-end application invokes the backend program to add the meeting. Upon serving the request, the backend will send the status of the transaction (i.e. success or failure) to the front-end application
9. The browser displays confirmation response or error messages to the meeting creator
10. The meeting creator sees the response and closes the Cornell Group Finder webpage or repeats steps 4 to 9 again if there is any error

## Scenario 2 - View Current Meetings

**Purpose:** This scenario describes the process of viewing a list of current meetings and the details of any particular meeting on that list

**Individual:** Any person interested in finding current meetings (i.e. students, group study members, library administrator)

**Equipment:** A computing device with internet browser and network connections

**Scenario:** (We will use "student" to represent the individuals in this scenario)

1. Student turns on the computing device and opens browser
2. Student types the URL to access the Cornell Group Finder homepage
3. The backend server fetches the homepage layout sends it to the browser for display
4. From the homepage, student selects the number of meetings to show per page and selects a time window (i.e. day, week or customized window)
5. The front-end application sends a request to the backend, receives a response with a list of meetings that fits the student's time window of interest and displays the list of meetings
6. Student clicks on a cell in the list view to select a particular meeting
7. The front-end application switches to a new page that shows all the detail information associated with that particular meeting

## Scenario 3 - Search Meetings

**Purpose:** This scenario describes the process of searching a list of meetings that matches certain conditions and finding the details of any particular meeting from the resulting subset

**Individual:** Any person interested in finding meetings (i.e. students, group study members, library administrator)

**Equipment:** A computing device with internet browser and network connections

**Scenario: (We will use "student" to represent the individuals in this scenario)**

1. Student turns on the computing device and opens browser
2. Student types the URL to access the Cornell Group Finder homepage
3. The backend server fetches the homepage layout sends it to the browser for display
4. From the homepage, student selects the search meeting tab
5. The front-end application switches to the search page user interface
6. Student enters some search criteria (i.e. time, location, meeting type)
7. The front-end application sends a request to the backend, receives a response with a list of meetings that fits the search conditions and displays the list of meetings
8. Student clicks on a cell in the list view to select a particular meeting
9. The front-end application switches to a new page that shows all the detail information associated with that particular meeting

## Scenario 4 - Edit Meeting

**Purpose:** This scenario describes the process of editing an existing meeting by the creator

**Individual:** A meeting creator (i.e. teaching assistants, senior students, group study leader)

**Equipment:** A computing device with internet browser and network connections

**Scenario:**

1. Meeting creator turns on computing device and opens browser
2. Meeting creator types the URL of the meeting share link that was emailed to him/her upon meeting creation to access the meeting details page
3. The server fetches the meeting details page layout and sends it to the browser for display
4. Meeting creator clicks on the button that says "Modify"
5. A pop up will appear to prompt the meeting creator to enter the password associated with that particular meeting
6. Meeting creator enters the password into the text-box of the pop up and receives authentication to edit the meeting
7. The server fetches the meeting edit page layout and sends it to the browser for display
8. Meeting creator can edit the following information from the page: meeting title, meeting location, meeting time, meeting type, meeting description and meeting privacy preference
9. After completing the edit, the meeting creator clicks the "Done" button

10. The front-end application invokes the backend program to edit the meeting. Upon serving the request, the backend will send the status of the transaction (i.e. success or failure) to the front-end application
11. The browser displays confirmation response or error messages to the meeting creator

## Scenario 5 - Gathering Meeting Statistics

**Purpose:** This scenario describes the process of gathering some statistics regarding all the meeting records in the database

**Individual:** Mann Library Administrator

**Equipment:** A computing device with internet browser and network connections

**Scenario:**

1. Administrator turns on the computing device and opens browser
2. Administrator types the URL to access the Cornell Group Finder administrator interface
3. Administrator enters login credentials to authenticate with the system
4. Upon successful login, the administrator interface homepage will be displayed
5. Administrator inputs the conditions that matches the statistics he/she is looking for
6. Front-end application makes a request to the backend which triggers a file export transaction that contains all historic meeting data that fits the pre-select conditions

## Revised Design

### User Interface

We have distilled the interface into four sections associated with different key functions: meeting create, view/search, edit/cancel, and statistics or admin view.

These key functions have remained consistent since our preliminary design decisions as discussed with the client. The implementation of these functions has changed in various ways as we have accounted for user feedback and feedback directly from the clients. In short, we redesign our web application to implement more features while remaining simple, clean, professional, and concise.

### Admin Interface

Now that our responsive web application has undergone a full iteration of development and testing, the client has indicated that we are ready to begin developing an administrative interface that allows for privileged access and statistics gathering.

Because we have previously discussed preliminary requirements for this interface, we have designed a very simple framework to start the interface. We understand that this is just our first implementation and that we will be revising this component drastically throughout this iteration. The client's function requests will be top priority as well as the security of the authentication and editing system.

To our understanding, the administrative interface will serve two main purposes:

1. **Meeting editing and override**

Admins should be able to edit any meeting without the meeting password. They will also be able to change the meeting creator email (a feature that is unavailable to meeting creators and visitors).

2. **Statistics querying**

Admins will have an interface that will provide information about current, past, or future meetings upon request. We are still working with the client to develop a list of important pieces of statistics that will be useful to librarians. This is why the interface will be revised throughout this iteration. Also, our design goal is to make an interface that is both intuitive and easy to modify if librarians want to get new pieces of information out of the statistics interface in the future. We will be working directly with Nick to ensure this goal is achieved.

Our very first version of the interface is shown below:

## Peer Learning Group Finder – Admin Interface V1

### Administrative Interface


To edit a meeting, enter a meeting URL or ID :

### Statistics Interface

Query shortcuts:

...or enter a custom query:

Search Text:

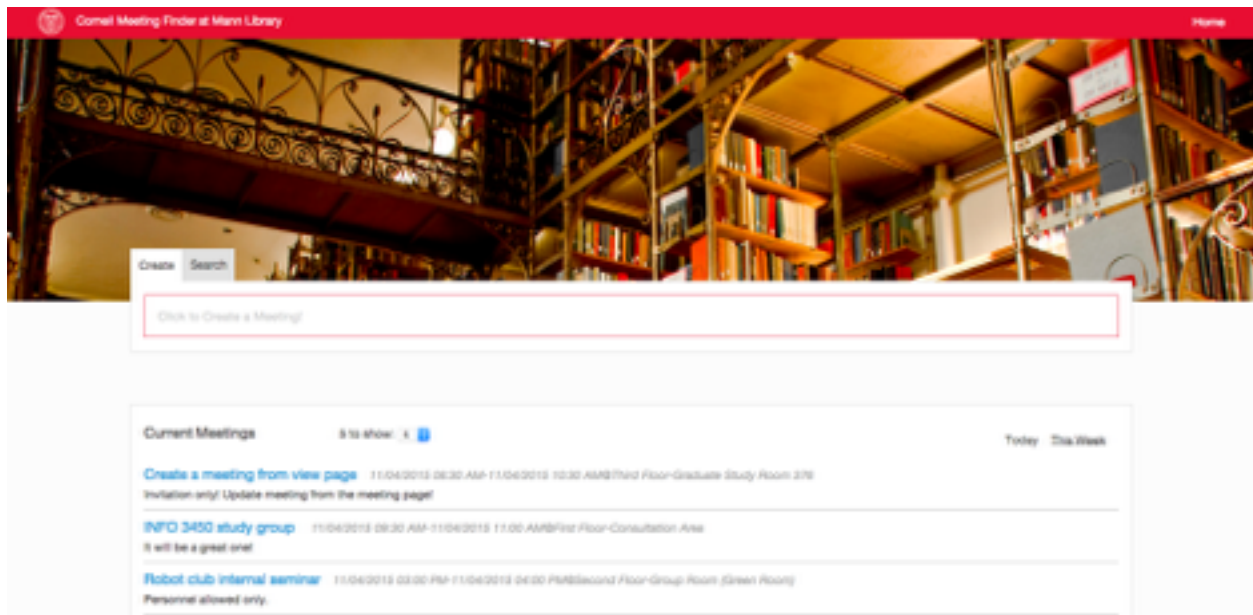
Meeting Type:  

Number of Meetings:	Most Common Floor:	Most Common Room:	Average Duration:	Most Common Meeting Type:
0	None	None	0 minutes	undefined

## Home Page

We got some feedback on our user interface from user tests: *Simple but not beautiful, Site itself needed to feel more legitimate.*

To achieve a more professional and clean interface, we used the same color scheme as the other library sites, and we added a Cornell logo with a library photo to match the general templates used by Cornell University.



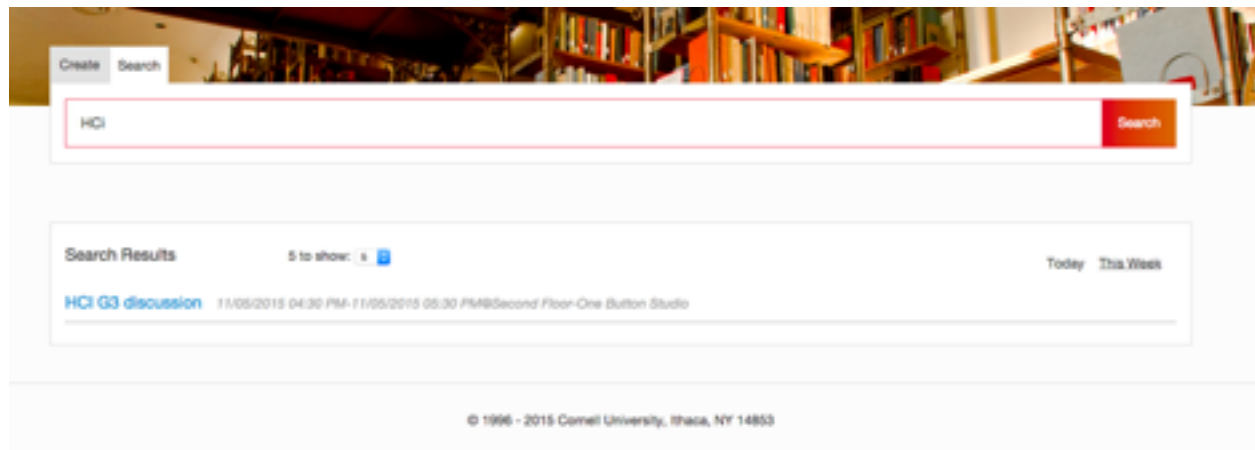
And also keep the structure of our site as before: show a list of meetings will happen today by default. You can change to view all the meeting that will happen in this week.

## Search Page

Much of the usability testing feedback agreed that meeting searching was unintuitive and difficult. Therefore, we worked hard to remove many of the bugs in the search system. We also implemented search by keyword for meeting titles and descriptions. We will be working throughout this iteration to implement more search features such as filters, times, dates, and locations.

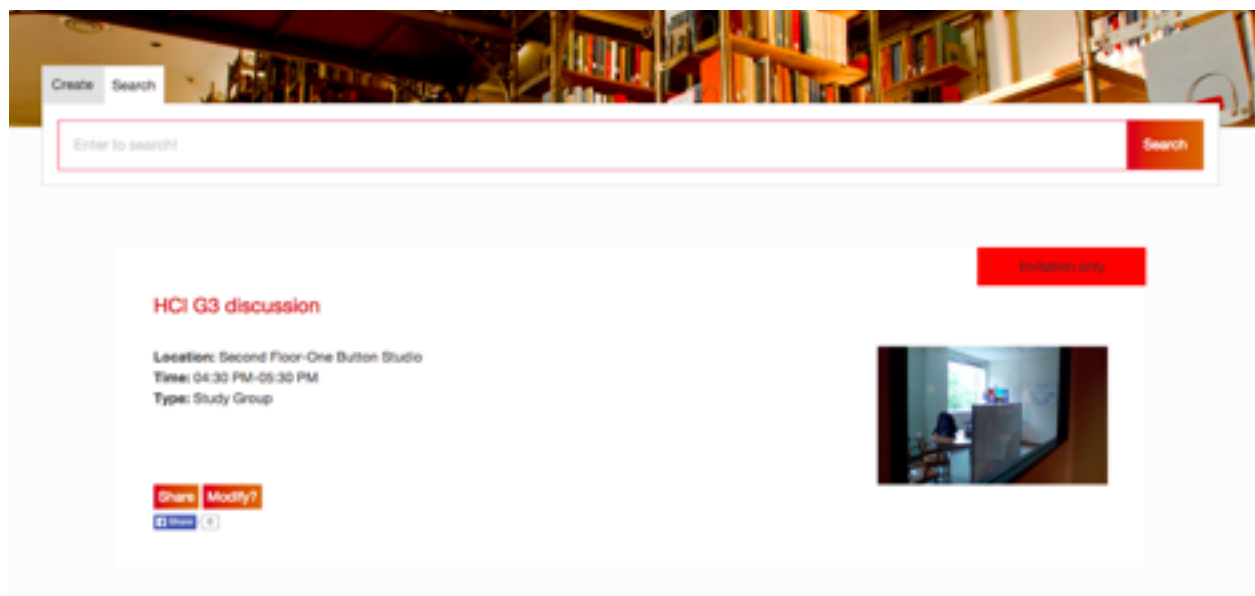


Another important piece of feedback that we received from usability testing users and from the client was the general difficulty when looking for the search function. To combat this issue, we redesigned the interface so the search function would be available from all pages on the website. By placing the search button in a central location, we hope that users will more easily find the search button.



## View Page

The view section has been revised with a more clean and intuitive style. In order to keep the create section and search function easy to find, we keep the header, create, and search group buttons in view section. We also consider the fact that people may use the search function more than the meeting create function while in the view page. For this reason, we keep the search function active by default in view section.



Besides all important and basic information about the meeting, we also will show a picture of the room if we have one which will give user more information about the room and location.

Perhaps one of the most unanimous pieces of feedback we received from users was the addition of the “invitation only” tag as shown on the top right hand corner of the view page. This tag is vital because it shows users that the meeting is either “open to everyone” or “invitation only”

Now you still can share the meeting with your friends through the share button or click the Facebook to share this meeting on your Facebook. And of course, if you are the creator of the meeting, you can go and update the meeting information if you want.

## Edit



Since we generate a memorable password for every meeting, you will need a password to edit a meeting. The authentication section is clean and simple. Due to feedback from users, we added a cancel button to make sure you can go back to the home page if you enter this section by accident.

And if you click you forgot the meeting password, we will send the creator of the meeting an email with a newly generated password.

If you enter the correct password, you will enter the edit section and can update the information you want.

Create a meeting from view page

\*Indicates a required field

Contact Information:

\*Email:

Location:

Choose:

Directions:

Details:

\*When:   to

\*Type:

Audience (who's invited):

Everyone can find your meeting, but we'll mark it "Invitation Only" if you choose.

\*Choose: ☐ Open to Everyone [Mark as Invitation Only](#)

Meeting Description:

Invitation only!

Update meeting from the meeting page!

The structure and style for the edit form will very similar to creation to keep the consistent in our website. You just need to edit the information you want to update, and click the update button, then we will update it for you. An important update to this interface is the locking of the creator email field. This field will only be editable by admins.

## Functionality

The sections above reflect the outline of the web application page by page. Abstracting away the pages of this application reveals the following important functions of our web application.

## Meeting Create

This important functionality will allow a meeting creator to input relevant information and create a meeting. Our preliminary user interface is shown below:

Each of the fields above is summarized briefly below:

**Title:** A text entry field containing the title of the meeting

**Location:** A button that brings up a location chooser. We will use a hierarchical definition system as follows: (floor → room)

**Time:** A date and time selection pane for selecting a range of time for the meeting

**Type:** A dropdown selection field to select a meeting type (project meeting, etc.)

**Contact:** A text entry field containing the public contact email of the creator

**Location Details:** A text area for giving additional location details. Default is none.

**Audience:** User chooses whether the meeting is open to the public

**Description/Details:** A text area for giving additional meeting details. Default is none.

There are additional functions that we have outlined with the client:

**Reserve:** only check If a creator selects a room marked "reserve-only" by the system, the user will be notified if the room has already been reserved (according to the library's scheduling API). Information about pre-scheduled meetings will be shown, however the meeting creator will still be able to create the meeting.

**Visual Interface:** Upon selecting a room in the location section, we will show an image and description of the room, provided that media is available.

## Meeting View and Search

Because the search and view functionalities of the responsive web application are very closely related (search brings the user to the view function), we will be speaking about them simultaneously.

The search function is the most important way to explore the meetings all user have created. You can type anything into the search field, and the view function will list the results we find that match your search. We will support several filters during the search, like location, time range, meeting type, and so on.

After discussion with the client, we will be starting with the following features as the base of our search function:

**AutomaticallySearched:** The search field will automatically search meeting title and description

**MeetingType:** Meeting type filter will show only meetings of a certain type

**Time:** Allows specification of date and time range to search

**Location:** Allows hierarchical searching of locations (allows searching an entire floor or a single room)

**PublicMeetings:** Checkbox allowing searching for only open meetings

The list of meetings shown in the view section can be resized (a user may select to see fewer or more results) and each of the meetings will be linked to the meeting view page.

The meeting view page is a sub-functionality of the view feature. This function will allow a user to view specific details about a given meeting. Because each meeting will have a unique meeting ID, each meeting will also have an associated meeting view link. Our current design of the meeting view page is shown below:

## Meeting Edit

The meeting edit feature is almost identical to the meeting create feature in the sense that all of the same input fields will be available for edit (title, description, location, time, etc). There is one key exception to note: the meeting creator email will not be editable. This is to ensure the security of the meeting edit system. Due to the fact that meeting passwords are reset via email, this field must remain secure in order to maintain the security of the meeting password.

If a meeting creator must change the email associated with a meeting, they must contact a system admin (who has the privileges to edit any information in any meeting).

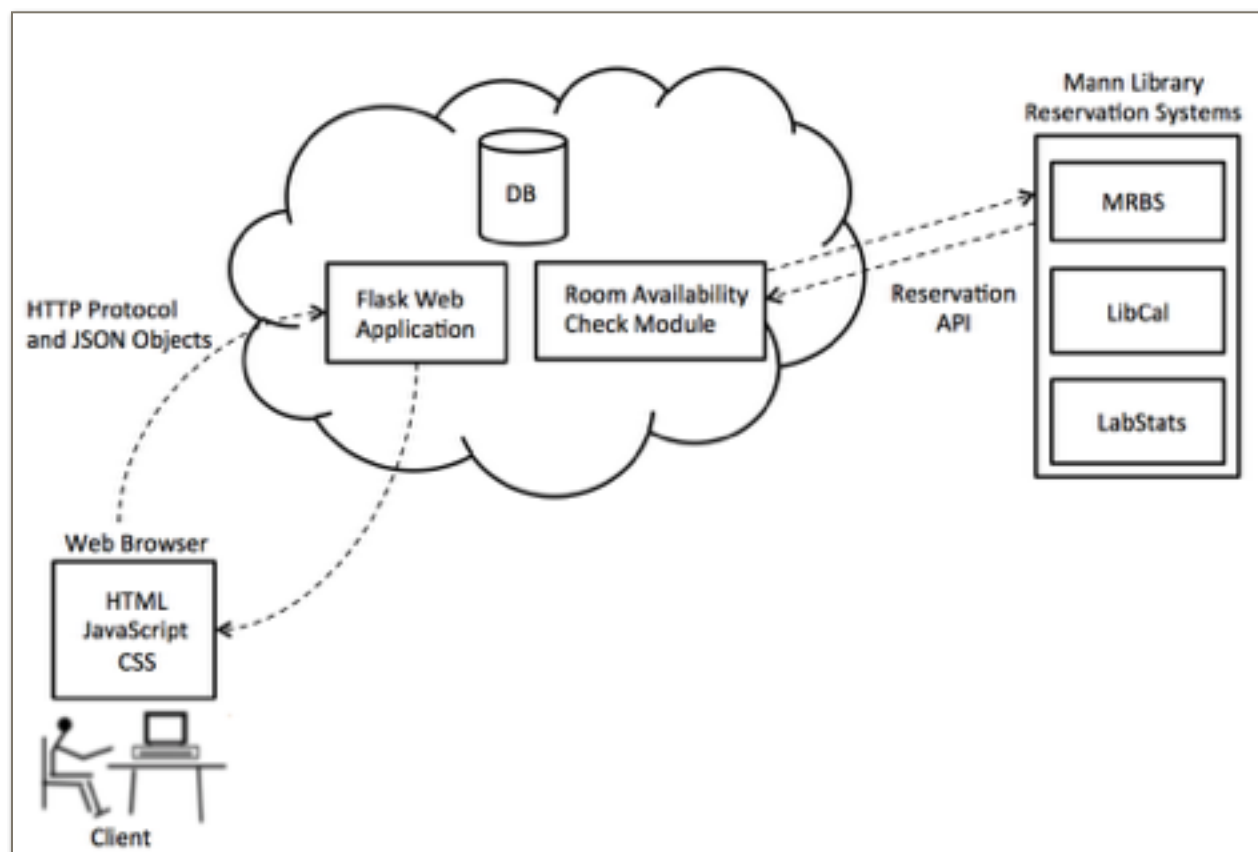
The preliminary design for meeting edit is shown below:

## System Design

### Architectural Design

Cornell Peer Group Finder's system is based on the client/server architecture, as illustrated in the diagram.

On the client's side, the user will connect to Cornell Peer Group Finder via a web browser, which will execute HTML5 and CSS. Javascript will be used to create interactive pages, improve user experience, and verify user's inputs. The client communicates with and sends data in JSON object format to the server through an application protocol HTTP .

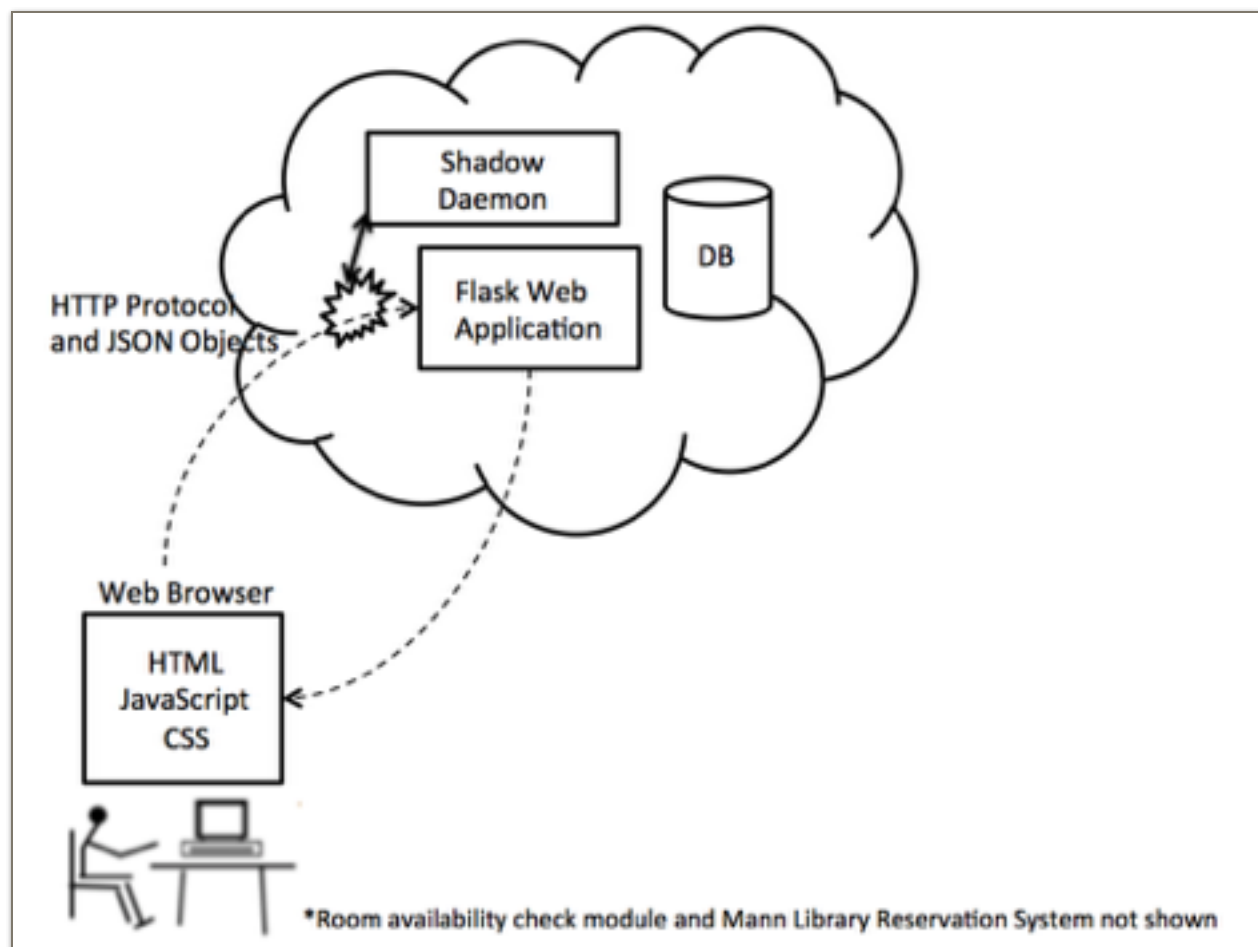


On the server's side, the RESTful Flask web application and database will be hosted on the testing environment, Amazon Web Services. The responsive web application performs input verification and connects to the MySQL 5 database, which stores information about meetings. Additionally, it will send HTTP requests to the Mann Library server to obtain information about room's availability.

Also on the Amazon Web Services is the Room Availability Check Module. Using the API and keys provided by Mann Library, the module connects to the three Mann Library reservation systems: MRBS, LibCal, and LabStats. Each of reservation systems stores availability information of different types of rooms. Currently, the web application and the module are developed in parallel. In the future, the module will be integrated with the web application.

# Web Security Plan

Security has been the team's focus throughout the second iteration phase. To prevent modern attacks on web applications, the team has informed the client of its plan to deploy Shadow Daemon, a web application firewall.



Shadow Daemon is a free open source software, released under the license GPLv2. It supports Flask Web Application Framework, which eases the integration task with the current system. Moreover, it allows user-defined white- and blacklisting, the design of which will be discussed with the client. The above diagram explains the role of Shadow Daemon. The daemon intercepts HTTP requests and filters out malicious parameters to prevent common attacks such as SQL injections, Denial of Service, spams, cross-site scripting, code injections, XML injections, and many more. Only requests that are deemed safe by the daemon will be processed by Flask Web Application.

## Meeting Reservation System Integration

In this milestone, we have started to integrate our backend server with the existing library room reservation system. Currently there are three different library room reservation systems that are used by the Mann Library administrators: MRBS, LibCal and LabStats. The communication protocol between our server and the library systems makes use of the RESTful API with HTTP requests and JSON responses. When there is a meeting creation request for a reserved-only room, the backend system will send a request to one of the three library reservation systems with the room ID as well as the meeting time. The library reservation systems will reply and indicate whether there has already been a reservation for that room during that particular time. By integrating these two components, we can inform the users on the room availability and remind them to make a reservation with the library system or prompt them to select a different time.

## Database Design

We have been making major changes to our Database systems to better accommodate the feedback from our first round of user testing, clients' suggestions, and changes from front-end team. The most updated Database Entity-Relation Diagram is shown in the figure on the following page.

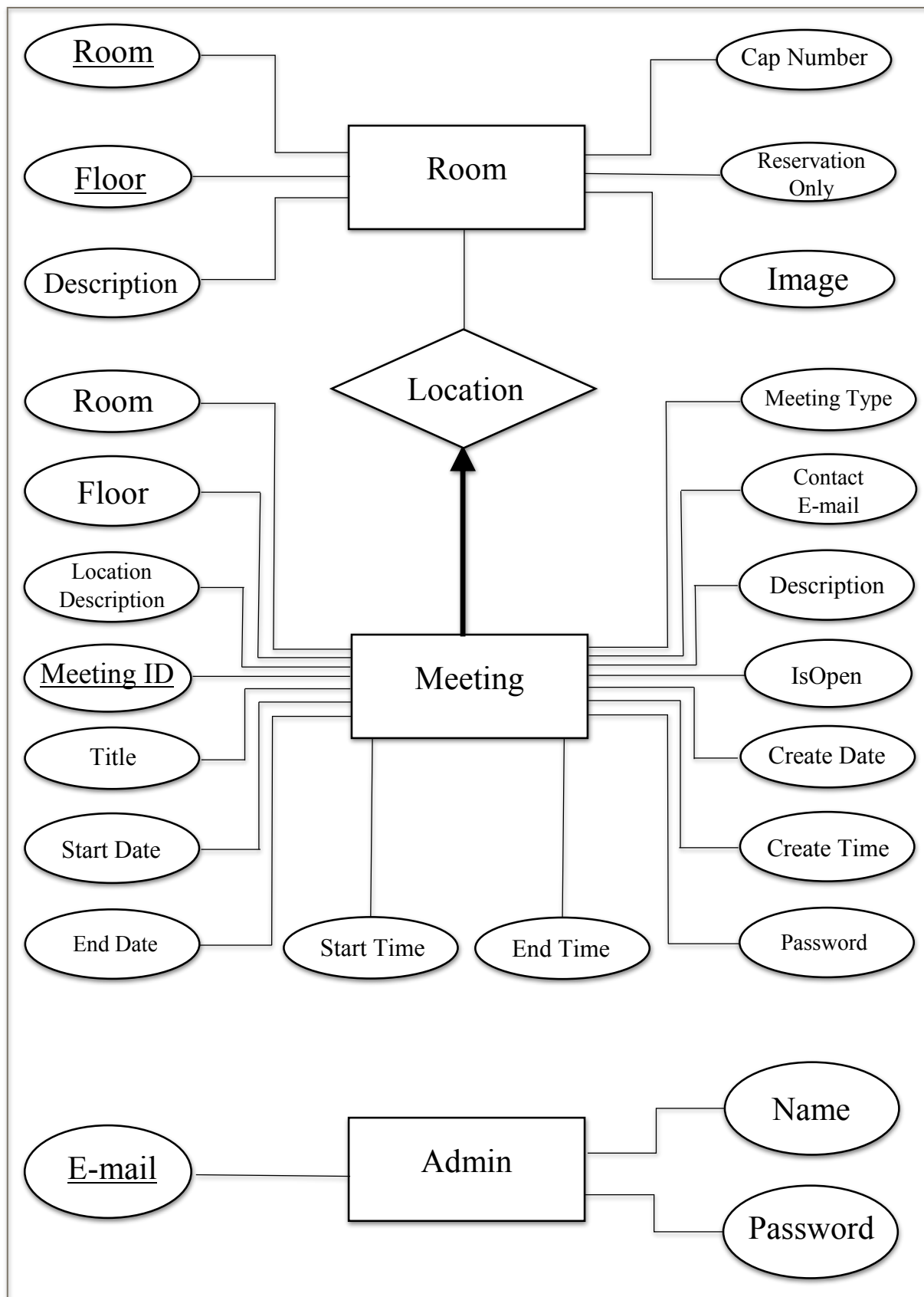
We kept most of the structures of the ADMIN table, the ROOM table and the one-to-many relationship between ROOM and MEETING tables. However there are a few changes to the MEETING table:

1. Location Description (LOC\_DESC) is a newly added attribute to a meeting, as in the latest version of the user interface, users are allowed to add additional direction instructions to help others better find the location.
2. Start time and end time are split into START\_DATE, START\_TIME and END\_DATE, END\_TIME. We are making this change to better accommodate the clients' needs as administrators, who might wish to obtain statistical information of meeting frequency at a certain time period of a day. Regarding this specific search, we introduced additional B+ tree indexing on both (START\_DATE, START\_TIME) pair and START\_TIME itself so that pulling results of those queries is efficient.
3. CONTACT\_NAME field is removed. Per the request of our clients, we no longer ask for users' names to better keep the users' privacy.
4. IS\_EDITABLE field is removed. After several discussion with the clients, we have come to the agreement that the edit operation is either authorized to a user who knows the meeting

password (who is, by default, the meeting creator) or an administrator. Therefore all the meetings are technically editable and this field becomes unnecessary.

5. CREATE\_DATE and CREATE\_TIME fields are introduced to record the timestamp when meeting is created in the database. With these fields we will be able to support more searching options, sorting operations and statistical information for the users and administrators.





## Application Programming Interface (API) Design

1) This API gets all the meeting records in the database (status: complete)

**API Method:** HTTP GET

**API Name:** "meeting"

**Inputs:** None

**Outputs:** An array of JSON objects and each object contains all the fields of a single meeting

2) This API gets all the rooms records in the database (status: in progress)

**API Method:** HTTP GET

**API Name:** "getRoomDetails"

**Inputs:** None

**Outputs:** A JSON object containing 5 fields for the different floors and each field will have an array of JSON objects that contain all the fields of a single room belonging to that floor

3) This API gets a particular meeting detail information (status: complete)

**API Method:** HTTP GET

**API Name:** "getMeetingDetails"

**Inputs:**

Parameter Name	Type	Required?
id	string	yes

**Outputs:** A JSON object that contains all the fields of a single meeting as well as the room description and room image associated with the location of that meeting id

4) This API creates a meeting record and inserts it into the MySQL database (status: complete)

**API Method:** HTTP POST

**API Name:** "createMeeting"

**Inputs:**

Parameter Name	Type	Required?
title	string	yes
locationFloor	string	yes
locationRoom	string	yes
locationDesc	string	no
timeFrom	string	yes
timeTo	string	yes
meetingType	int	yes

contactEmail	string	yes
isOpen	boolean	yes
description	string	no

**Outputs:** A JSON object that contains the status of the request and if success, the meeting id of the newly created meeting

5) This API finds out if a room is reserved only, checks if there is a conflict and returns the URL of the room reservation site if the room is still available. (status: incomplete)

**API Method:** HTTP GET

**API Name:** "isReservable"

**Inputs:**

Parameter Name	Type	Required?
locationFloor	string	yes
locationRoom	string	yes

**Outputs:** A JSON object that contains a boolean field to indicate whether this timeslot is still available. If true, the object will also contain a field with the URL link to the reservation system

6) This API edits one or more existing information in a particular meeting (status: complete)

**API Method:** HTTP POST

**API Name:** "editMeeting"

**Inputs:**

Parameter Name	Type	Required?
id	string	yes
title	string	no
locationFloor	string	no
locationRoom	string	no
locationDesc	string	no
timeFrom	string	no
timeTo	string	no
meetingType	int	no
isOpen	boolean	no
description	string	no

**Outputs:** A JSON object indicating the status of the request (success/failure)

7) This API searches all current and future meetings based on a set of conditions (status: in progress)

**API Method:** HTTP GET

**API Name:** "searchMeeting"

**Inputs:**

Parameter Name	Type	Required?
keyword	string	no
locationFloor	string	no
locationRoom	string	no
timeFrom	string	yes (if timeTo is set)
timeTo	string	yes (if timeFrom is set)
meetingType	int	no
isOpen	boolean	no

**Outputs:** An array of JSON objects that satisfies the search conditions and each object contains all the fields of a single meeting

8) This API resets the meeting edit password for the creator by sending an email with a reset link (status: incomplete)

**API Method:** HTTP GET

**API Name:** "resetPassword"

**Inputs:**

Parameter Name	Type	Required?
id	string	yes

**Outputs:** A JSON object indicating the status of the request (success/failure)

9) This API authenticates the editor in order to proceed to meeting edit (status: complete)

**API Method:** HTTP POST

**API Name:** "authPassword"

**Inputs:**

Parameter Name	Type	Required?
meeting_id	string	yes
password	string	yes

**Outputs:** A JSON object that contains a boolean field to indicate a match or no match of the passwords for this authentication transaction. If there is a match, it returns all the field information associated with that meeting id.

10) This API authenticates the library administrators in order to proceed to the administrator interface (status: incomplete)

**API Method:** HTTP POST

**API Name:** "authAdmin"

**Inputs:**

Parameter Name	Type	Required?
username	string	yes
password	string	yes

**Outputs:** A JSON object that contains a boolean field to indicate a match or no match of the passwords for this authentication transaction

## Usability Testing

Our next iteration of development is centered around a second round of usability testing. We are currently working with the client to develop dates and times for usability testing with students and librarians. These half hour time slots are tentatively schedule for November 16th and November 17th.

We are also working on a set of usability testing tasks that are centered around use case scenarios. We will be building upon our previous tasks so we can compare new results with feedback from the first round of usability testing. In this manner, we can use our results from the first round of usability testing as a benchmark for the second round of testing.

In addition to the scenarios originally proposed in our use case scenarios, we are introducing another scenario: add meeting that is in conflict with another meeting. To best simulate this situation, we will ask the user to schedule a meeting for a specific time and room. This room and time will be chosen to conflict with a previous reservation on either the library reservation systems or on the peer group learning finder application. Because the tester will not previously know that the meeting will be in conflict, we will be directly testing the system's ability to alert users to meeting conflicts and direct users to meeting conflict resolution strategies.

Our preliminary set of usability testing scenarios may be found below. We are still working with the client and we will be taking their feedback into account while conducting this second round of usability testing. It is important to note that the task descriptions have not changed however we are testing new features within the task scenarios.

We have bolded testing features that are new to this round of usability testing.

Task #	Task Description	Front-end features	Back-end features
1	Set up a study group meeting just for your PSYCH 1103 section somewhere on the second floor for early next week.	i. Meeting creation interface ii. User sharing message confirmation iii. <b>Confirmation message after submit</b> iiii. <b>Intuitiveness of date, time, and location selection system</b>	<b>i. Room availability check with Mann Library Reservation System</b> <b>ii. If room is unavailable, suggest alternative time slots on the same day</b> iii. Email confirmation sent to the meeting creator iv. Successful update to the database
2	Notify the rest of the students in your section about the details of the meeting you just created	i. Share button for public link ii. Public link should direct user to current meeting iii. <b>Test Facebook share</b>	i. Identify and display meeting details from the public link
3	Find an upcoming meeting (open to the public) that you would like to join	i. Meeting view and search interface ii. Meeting list on front page iii. <b>Visibility of “invitation only” tag</b>	<b>i. Keyword search based on meeting title and description and time range</b>
4	Edit the PSYCH 1103 meeting that you created earlier and change the start time to 7 pm based on feedback from the rest of your group	<b>i. Authentication interface including cancel button</b> ii. Meeting edit interface	i. Authentication handling <b>ii. Room availability check with Mann Library Reservation System</b> <b>iii. If room is unavailable, suggest alternative time slots on the same day</b> iv. Email confirmation sent to the meeting creator iiv. Successful update to the database

## Schedule & Future Plan

The team's design phases have been designed to be driven by usability testing. Therefore, after conducting the usability test, the team will discuss with the client the team's observations and possible improvement to the front-end and back-end features. The team will then proceed according to the third iteration plan as outlined in the feasibility report.

As we have just concluded our first iteration, our previous schedule is listed below:

- Front-end

- Add and improve HTML pages according to improvements that we discussed with the client after the usability testing ✓
- Connect more HTML pages with back-end REST endpoints. ✓
- Back-end
  - Modify database schema and RESTful API to accommodate with front-end changes ✓
  - Implement more back-end functions for REST endpoints ✓
  - Enhance web application security by adding
    - input verification ✓
    - email spam filtering (*in progress*)

Note that we are still working with the client to develop an email spam filtering system that balances security with requested features. Therefore, this point will also appear on our third iteration.

Our **second iteration plan** is listed below:

Time frame: **November 9th - November 29th**

Our next-round of usability testing will be scheduled at the beginning of the next phase and we will start with collecting and analyzing the feedbacks from the usability testers to consider in what way we can further improve the interface of the web-based system and what are some of the new features that we should definitely consider adding to our application.

It's important to note that this time we will be focusing on creating various use case scenarios for the testers in order to obtain more valuable feedback information regarding their interactions with the system. In addition to the testing, one of the most essential features that we are planning to incorporate into the system is spam detection since our ultimate goal has always been to build a easy-to-use, robust, and secure system for our client. Aside from a security update, we will be also planning to polish our search functionality by adding some advanced search filters such as room floor and meeting time. As for the admin interface and its corresponding functionalities, we will be working closely with the client to implement an admin system that is functional and intuitive.

To be more complete with our usability testing process, we will also be working with librarian testers in order to test both the responsive web application and the admin interface.

#### Front-end

1. Perform another round of user testing and implement revisions
2. Check in with clients and clarify changes with user testing evaluators
3. Add and modify application according to improvements that will be discussed with the client after the usability testing
4. Connect the rest of the HTML pages with back-end REST endpoints.
5. Update admin interface and statistics functionality

#### Back-end

1. Continue integration with all reservation systems
2. Add meeting conflict detection (some rooms allow multiple meetings)
3. Modify RESTful API to accommodate with front-end changes
4. Add statistics and data mining system for admin interface
5. Implement more back-end functions for REST endpoints
6. Enhance web application security by adding email spam filtering functionality
7. Improve the performance of search functionality
8. Add maintenance functionality, polish API and documentation

We will proceed with our plan as outlined above, tentatively starting with usability testing on November 17th.

## Live Demos and Other Documents

For live demos of our current preliminary design and other important documents relating to our project, we have assembled a project hub page at the following url:

[groupfinder.gabeabrams.com](http://groupfinder.gabeabrams.com)

There, you will find links to much of the important material that we are currently working with.