Design Document for MITspaces

By: Anthony Chen, Brodrick Childs, Nahom Workie

---

1. Overview

Searching for an empty room to work in is oftentimes a difficult problem to solve as a MIT student. A search for an empty location can bring your search through the Infinite and multiple floors of the student center. Our application MITspaces aims to solve this problems.

MIT spaces will be a Ruby on Rails application that displays possible study areas in the MIT community. The goal of our application is to provide a crowdsourcing platform to allow users to discover available rooms to study or work in.

User will be able to determine whether certain locations are currently occupied or not. After a location is chosen, the user will be able to check-in upon arrival and crowdsource the information about the room. This will allow other users of our service to know if the room is available or note. Similarly, users will check out when they are down with a particular room.

2. Concepts

The key concepts for MITspaces are Collaborators and Rooms. Collaborators use MITspaces to find rooms available to study in. The service connects collaborators to other who are available by connecting to their Facebook account. These Collaborations are able to check into Rooms, at which point MITspaces will track it's status.

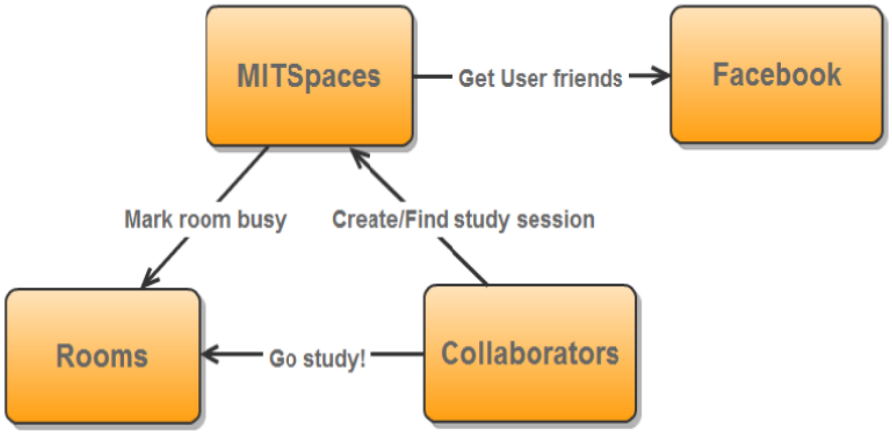

Figure 2.1: Context Diagram for MITspaces

2.1 MVP Concepts:

In the MVP implementation of MITspaces, the two main concepts are rooms and users. In the aspect of the design, our team attempted to tackle our two major challenges - authentication and mobile checkins. For the User, we implemented Facebook checkin with the assumption that our users are most likely Facebook users. For rooms, we implemented a url that modifies the availability of the room, which is saved through NFC tags and QR codes.

2.2 Final Concepts:

The final implementation of MITspaces still focuses on the same initial concepts of the project, but their names and identification has changed. The focus of the project has changed to simply allow all Users to find available study spaces or Rooms; therefore, the final concepts of our project are Rooms and Users.

3. Behavior

3.1 MITspaces Feature Description (Updated for Final Design)

1. **Secure Login** => Users can login to their accounts securely using their Facebook accounts. This enables the site to also get a list of the users friends and check if anyone of their Facebook friends also uses MITspaces.
2. **Find Room** => Users can find an empty room on campus. Using the Google Maps API users can see all the rooms on campus and their current availability. The map allows for an easy way to see exactly where a room is and when clicked users will be able to see more information about the room. Users are also able to look through a sorted table listing to discover rooms. The table presents the users with detailed information for all available and occupied rooms.
3. **Occupy Room** => Users can check into a room and have it marked as occupied for others to note. When a user checks into a room via mobile check-in the rooms occupation status immediately changes to true (i.e occupied). The color status of the pointer for the room on the campus map will also change to red to denote that it's been occupied. The map uses AJAX request to continuously update the current state of occupied and available rooms. The ability to checkout rooms with a mobile device simplifies the reservation process and aid in the usability of the service.
4. **Add Room** => Users can add a study room and have it approved by administrators. If a user has a room on campus that they study in and want to add it to the MITspaces database they can request to add a room. The user will input information such as Name, Location and Description. This request is then sent to the administrators who then have to check the room and approve it to be listed and shown on the MITspaces map.

5. **Delete Room (Mods)** => Administrators can delete rooms. Users can have a room to be deleted from MITspaces if students can't study in the room or if multiple users request that the room be deleted. Ultimately, this decision is made by the administrators who have the final word in the approval process.
6. **Find Friends** => Users can find their friends and if they have occupied a room. Using the Facebook API allows MITspaces to access a user's friend list. If the user's friends are occupying a room, when a user clicks on an occupied room MITspaces will show which friends are occupying the room when it displays more information about the room. The service shows all of a person's friends who are also signed into the service.

3.2 MITspaces MVP Features:

1. **Secure Login** => Users can login to their accounts securely using their Facebook accounts. This enables the site to also get a list of the users friends and check if anyone of their Facebook friends also uses MITspaces.
2. **Find Room** => Users can find an empty room on campus. Using the Google Maps API users can see all the rooms on campus and their current availability. The map allows for an easy way to see exactly where a room is and when clicked users will be able to see more information about the room.
3. **Occupy Room** => Users can check into a room and have it marked as occupied for others. When a user checks into a room via mobile check-in the rooms occupation status immediately changes to true (i.e occupied). The map uses AJAX request to continuously update the current state of occupied and available rooms.
4. **Add Room** => Users can add a study room and have it approved by administrators. If a user has a room on campus that they study in and want to add it to the MITspaces database they can request to add a room.

Object Model

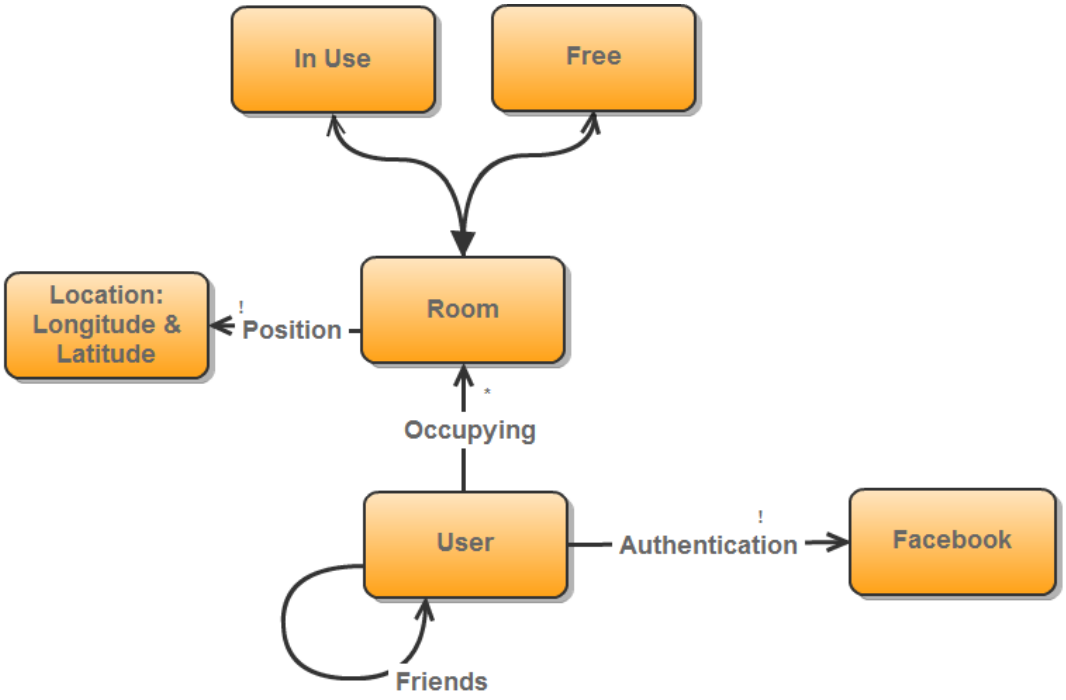

Fig 3.1: Object Model for MITspaces

Security

In this aspect of the project we handle security from a couple angles. Our largest concern is to make sure that our users are using the service in the manner in which it was intended. Since our application is built on the idea of crowdsourcing, we need make sure users are authenticated and they are submitting relevant information.

Login: Login is handled via Facebook, so we make sure the user is who they say they are that way. They will also be asked to confirm their MIT email address. Our goal with login is to be strict enough that we are only allowing users from the MIT community.

False Reporting: One of the usability issues that we encountered has been false information. Part of what makes MITspace successful is it's ability to provide the most up to date information for all the members of the MIT community. In order to account for this, users must be with a set radius of the room they are planning to check out. Also the use of mobile reservation encourages users to only checkout a room while they are there. Finally, there is a maximum time one checkin can last, at which point the user must re-checkIn. The use of these three security features best allows MITspaces to provide its users with correct information.

Cross site scripting/Access Control: The secure actions, like adding a room and checking out a room check to make sure the user has the proper authority and is able to make the action in the backend, so they are secured this way. If people try to send requests to the site with forged parameters to make or delete a note, it makes sure this user is logged in and has the proper credentials to perform the actions. Furthermore, the use of moderators adds an additional level of security in this aspect. Although the use of moderators increases the amount of time before a new room is submitted, it drastically improves the quality of information presented by our service.

User Interface

The figure below demonstrates the wireframe views for MITspaces and the flow between them. The focus of our interface is the main dashboard. On this screen, a user can locate where his or her friends are currently studying and which rooms are currently available or occupied.

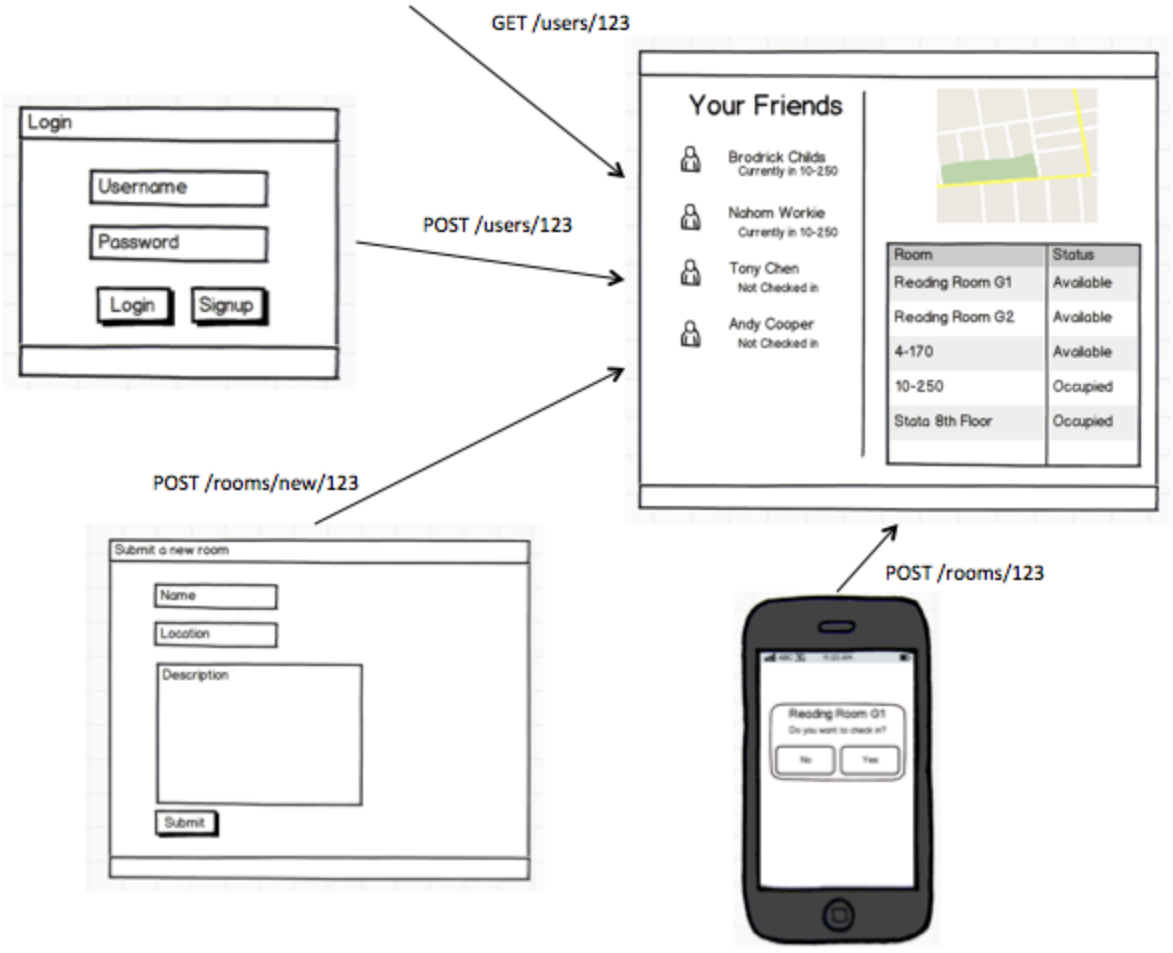

Figure 3.2: Wireframes for MITspaces

4. Challenges

One challenge was how to authenticate users. We want to make sure they're MIT students, so we need to send an email to an @mit.edu account to verify, but we also want the user's friends. Our options are listed below:

- Limited Authentication: utilize a rails gem and allow users to simply create an account with our application. This approach allows any user to create an account.
- Strict Authentication: force users to sign in with an mit.edu email address with veritification. This approach guarantees that we reach MIT students only, but requires a longer verification process.

We decided Facebook authentication was the best way to do this. Facebook will easily allow us to handle login and get a user's friends. We can easily send an email after the user has connected their account to facebook in order to authenticate them as an MIT student. Even without MIT email authentication, we don't expect non-verified users to pose much of an issue since they can't do anything without physically being in the room to check out

Due to the importance of our users receiving the most updated information on the status of the rooms, we needed to decide the best approach to keeping them informed. Our choices are listed below:

- Constant ajax refreshes: continuously submit ajax requests to the server to maintain that the user has the most updated information. This approach would provide users with the most information, but it would require sending many requests.
- Manual refresh: Force the user to refresh the page if they want to see new information. This drastically reduces the number of requests made to our server, but limits the information they receive.

In order to improve the user experience, we plan on implementing an approach that refreshes the pertinent information in periodic time intervals. This way, the number of requests is reduced, but the users are still receiving updated information.

Another challenge is how to implement the room checking out.

- Best approach: Place QR codes or NF patches in the actual rooms, and force users to check out the rooms when they get there (and force them to check out when they leave.)
- Simple approach: If QR codes prove impractical, we could just have the users check out rooms by visiting the website.

We decided that we will talk with the MIT administration to determine whether or not QR codes

are feasible, if not we will use the simple approach since it keeps the functionality within our site and has fewer external variables.

After a user is finished studying in a room, MITspaces hope to gather this information the moment he/she leaves. This data is oftentimes a very unreliable piece of data. Its often that a person is in a rush, and may not have the time to check the room back into our system. Our solutions for these challenges are:

- GPS based: recording a user's location, and forcing a room to become available once the original user leaves a designated radius. One of the issues with this approach is the need for a user to remain with our service. If he/she is not on the site, then the location cannot be tracked. This approach would be much more usable in the case of a targeted mobile app.
- Crowdsourcing: Allowing other users to make a room as available in a similar manner that they can checkout rooms. If they are in the area, then they can report a room to be empty. This approach, however userful, may be open to users submitting incorrect information. As an example, someone could report a room to be available while someone is working there.
- Setting a timeout: This approach would just set a maximum time that a user can stay checked in, at which point he/ she would need to re-checkout the room. This approach forces all rooms to have correct information based on the maximum time limit. The tradeoff with correct information is usability, and through user testing the best optimal balance can be discovered.

The decision we decided to implement is the timeout approach. This decision allows MITspaces to maintain updated information without security threats. In discussion, we decided that a hybrid approach of many of the choices may be the most usable solution, but at this moment providing correct information is more important. Our goal is to provide users with the correct information. A users going to an unavailable room marked as available is much worse for usability than an available room marked as unavailable.

5. Reflection

<u>Peer Review</u>

Brodrick Childs (about Tony): Tony worked a lot on the front end of the site, and did a good job there. The one thing I can think to give him as constructive criticism is to prioritize what parts he works on first, since he has a tendency to fixate on small details sometimes, and lose sight

of the larger picture. At the same time, he took charge of team meetings and taking updates, and did a great job there.

Brodrick Childs (about Nahom): Nahom took charge of maps and the first presentation. Struggled a bit to catch up to Tony and me sometimes since we've had more rails experience, so I would suggest that he perhaps have spent a little more of his time individually reviewing the code so that he might understand the code base better in team meetings. It was nice that he originally took charge of getting the initial maps code out of the way since Tony and I are averse to dealing with maps after doing it in previous projects.

Nahom Workie (about Brodrick): Brodrick did an amazing job with the ajax calls throughout our application. This really helped our site functionality and also improved the usability aspect of our site. One suggestion I would have would be to spend more time better organizing the features and feature planning because this would really help create a more streamlined process instead of creating confusion regarding who's working on what. However, he was really essential to the team especially with regards to integration with Facebook and usability through AJAX.

Nahom Workie (about Tony): Tony took charge of team organization and meetings as well as cleaning up the front-end views of the site. This really helped our site looked clean and easy to use. A suggestion I would have would be to focus more time on feature completion rather than spending so much time on styling details. However, Tony was a great teammate and really helped manage our timeline for completing our various features, and in doing so we were never really pressed on time for any of the deadlines.

Tony Chen (about Brodrick): Brodrick did a great job working on the Facebook aspect and ajax calls throughout the application. He was always available to work and was always on time to all of our meetings. One aspect that I would give as constructive criticism is to spend additional time planning some of the features that need to be completed. Overall, however, he was a critical piece to our team and a terrific DJ.

Tony Chen (about Nahom): Nahom was a great teammate and really helped get our product to completion. The addition of Google Maps to MITspace was his responsibility and he did a great job. One criticism would be to spend more time reviewing his code. As the team member with the least rails experience, he was oftentimes a little behind on understanding the code base. Nonetheless, his creativity and attention to details helped our team stay on track and create a very usable application.

Evaluation

Overall, MITspaces was an extremely successful project from a team planning perspective. Our object model and context model that we initially designed before any coding remained exactly the same throughout the entire implementation. Surprisingly, the features we initially wanted in our design were all implemented. Our team worked extremely well during the design aspect of project. Working in a team environment gave us more opinions to assess the timeline to create some of the features we initially wanted.

Furthermore, another aspect that aided in our project was set weekly meetings with each other to finish features. This allowed us to meet right before the assignments were due and enabled us to set goals for the week ahead. As a result, our team was never pushed to finish features for any of the deadlines. Each addition to our application was planned well in advance, and our time management to complete them worked out really well.

Lastly, the use of our priority list was the last tool that aiding in our planning as a team. After the MVP, we discovered that the number of aspects that needed to be completed where very hard to keep track of as. In order to keep track of it all, we met to create a priority list for all the features we could implement. This document, planned early, helped keep our team on track with implementing the most critical pieces to our application.

Lessons Learned

The difference between prototype and product:

Our team started the project with high aspirations of pushing this project into production for the general MIT community; however, through this goal we recognized the difference of launching a product that work compared to one that works well. We found ourselves constantly thinking of how to improve the security of our application in the case that users don't use it with the intended purpose. For example, in an ideal situation, users would always check a room back in after using it. In production, however, we believed that many people would forget to do so, thereby causing a security challenge even without a malicious intent. Situations like these are extremely abundant, and through this project, we learned how difficult it is to truly launch a product.

Planning Early (*Design Document Included*):

Planning early is always helpful, and we definitely saw the rewards of this lesson throughout our project. As a result of early and frequent meetings, our planning was extremely effective and we were able to implement all the features we initially set out to accomplish.

Getting an Outside Opinion:

As a team become more and more immersed in the project, the harder it becomes to keep an outside perspective on what is essential to the project. Our TAs, Helen and Shirish, served as these perspectives. For example, our initial MVP aspirations did not focus on the critical pieces of our application, and Helen was able to guide our goals to focus on the key aspects of our application before implementing some of the added features.