

Senior Project

Room-o-Pedia!

Linh Le, Rachel Lee, Angie Yang
Advisor: Chris Murphy

Submitted in Partial Fulfillment of the Requirements of the BA in Computer Science
Bryn Mawr College
Spring 2022

Abstract

In Bryn Mawr College's current room draw process, there exist a multitude of issues, many of which stem from the lack of widespread information of the rooms on campus. As a result, the majority of this essential information is spread through word-of-mouth. The inadequate resources result in various inequities and anxieties, as some students have access to much more information than others, just by knowing more people. We addressed this problem by building an iOS application to display information about Bryn Mawr dorm rooms, complete with a Node Express web server and a MongoDB database, storing all the room data, including Amazon S3 links to room photos. Our iOS application allows users to filter all the rooms in the database using existing features such as dorm, air conditioning, and storage options. Users are also able to view a particular room with all its features and photos from the database, search a room using dorm and number, and save certain rooms as favorites, which appear in the favorites tab. In our user study, 88.9% of participants rated *Room-o-Pedia*'s usefulness for Bryn Mawr College students going through the room draw process as a 5/5, and the remaining 11.1% of participants rated the app a 4/5.

Acknowledgments

Linh Le:

First, I would like to express my deepest gratitude to our advisor, Professor Chris Murphy, for guiding us through this project. Thank you for your patience, understanding, and immense support! I would also like to send the world's warmest hugs to my friends and family, who were always there for me no matter what: *Con yêu mọi người nhiều lắm!* Finally, to Rachel and Angie for making it through the sleepless nights together, I appreciate you guys with all my heart!

Rachel Lee:

First and foremost, thank you, Chris! I cannot express enough how appreciative I am to have Professor Chris Murphy supporting us. His encouragement, thoughtful advice, and realistic timelines provided us with the foundations of success. We will always be CARL <3. Shout out to my friends, who patiently listened to me complain about computer science for the past 4 years. Thank you for always being there, for celebrating with me when things are good, and for reassuring me of my self-worth when things are bad. Big warm thank you to my parents and my sister, who paved the way and will never stop reminding me. And finally, a special thank you to Linh and Angie. We did it! :)

Angie Yang:

Thank you to Professor Chris Murphy, who encouraged us through every step of the way and made us feel like we could succeed. We will always appreciate your empathy and genuine care for students! Thank you to all of my lovely friends for always making me laugh and for making college that much more bearable. Thank you to my family for your overwhelming love and support. 好爱你们！！ Last but not least, thank you to Linh and Rachel. I wouldn't have chosen to suffer through senior project with anyone else <3

Contents

1 Introduction	6
2 Solution	6
3 Design	8
4 Implementation	9
4.1 Node.js, MongoDB, and AWS S3	9
4.2 iOS Application	12
4.2.1 Feature: Email Verification	13
4.2.2 Feature: Loading View	14
4.2.3 Feature: View All Rooms	15
4.2.4 Feature: Filter	17
4.2.5 Feature: View One Room	19
4.2.6 Feature: Favorite	21
4.2.7 Feature: Search	22
5 Evaluation	23
5.1 Methodology	23
5.2 Results	24
5.3 Discussion	27
6 Related Work	29
7 Conclusion	30
7.1 Future Work	30
8 References	31

1 Introduction

The existing room draw process is difficult for students given the lack of standardized information about dorm rooms. Although students have access to building floor plans on the college website, these are often misleading about room size and do not provide more detailed information such as heating/cooling and accessibility features. Much of the existing information is scattered, hard to find, and shared solely through word-of-mouth.

The existing system reveals a significant problem. The lack of centralized information causes knowledge to be largely based on who you know and what resources you are aware of. For example, a student must be acquainted with a resident of New Dorm in order to know about the centralized heating/cooling, or visit a resident of Merion to know that only some rooms have closets. Recent efforts were made by a since-graduated student working with the Residential Life Office to take photos of all dorm rooms. However, the link to this information was only spread through social media by the said student, and so, only a portion of the student body (mostly upperclassmen) even knows about this great resource. Furthermore, even the dorm floor plans, which may be the most well-known resource for room draw, can be very misleading if a student is unaware that they are not accurate (rooms not scaled relative to each other, incorrect number of windows per room, etc.). In summary, this lack of centralized, accessible, and accurate information about dorm rooms causes much confusion and unequal knowledge. For many students, it is a difficult decision to choose the dorm room in which you will spend an entire year of your college experience. Inequitable knowledge of rooms causes students more stress before the room draw selection, as well as living situation issues after selection if they were forced to make an uninformed decision.

In order to solve this problem and assist students in making more informed decisions during the room draw process, our project was developed to create a convenient app that centralizes and publishes dorm room data. This was a challenging task to accomplish due to the scattered nature of dorm room information across multiple sites. Important room features are not documented anywhere, and can only be discerned by manually going through hundreds of images. In addition to gathering the data, our app also presents it in an accessible and easy-to-navigate format for students. Students have varying background knowledge and understanding of the dorms, so clarity was a critical consideration for our project.

Centralizing dorm room data in a convenient and accessible app greatly benefits Bryn Mawr College students who are looking to live on campus in the following year. Our goal was to make room draw information more equitable and thus improve residential living for Bryn Mawr College students.

2 Solution

To approach this problem, we implemented an iOS app for Bryn Mawr College residential students that provides various information about dorm rooms in an accessible, easy-to-use format in order to assist students in making informed room selections during the room draw process.

Our iOS app provides various main features necessary for students to make their informed decision on a dorm room, including browsing through all rooms in the database, viewing specific rooms and associated information (e.g. accessibility access, heating/cooling, or flooring installation), narrowing down room options based on different needs through a filter-by-tags system, and searching directly for a particular room.

For students participating in the room draw process, collecting all of the aforementioned detailed information on each room has continuously proved to be both time-consuming, difficult, and incredibly frustrating, let alone categorizing them into lists of potential options. By centralizing all data to one place and allowing the app users to filter through the readily available database with a single feature tag or a combination of tags, the app guarantees to lessen the time and effort put into this task by each student while ensuring equal access to the information within the student body.

Aside from providing users with frequently sought-after information, the app provides multiple photos of each room for easier visualization of room size, layout, and orientation. This grants app users a clearer look into each room to avoid any unnecessary misunderstandings previously caused by not-to-scale and vague floor plans. Users are also able to “favorite” certain rooms so they can save them for future reference. This feature will allow students to build their own list of potential rooms they would want to live in, further assisting their room draw experience.

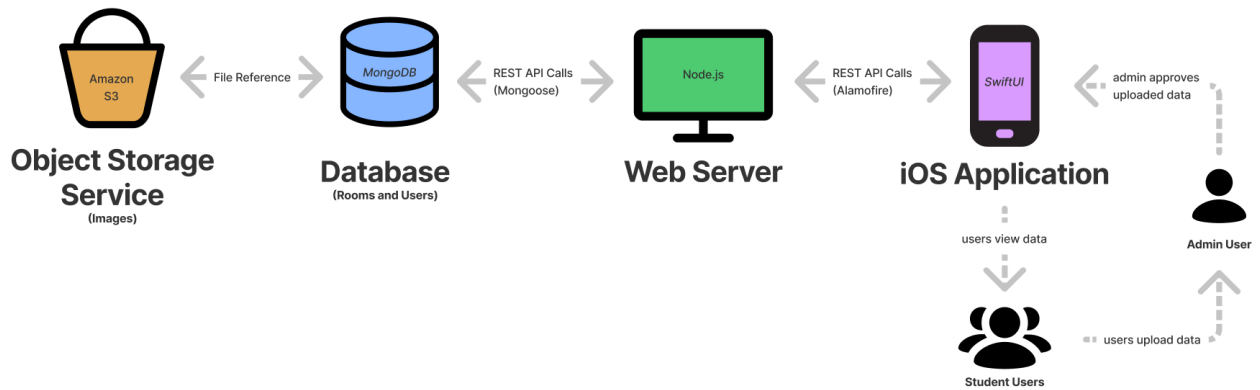
Drawing from our personal experiences and the experiences of our peers as seniors who have dealt with room draw during previous years, we aimed to fill in the gaps of misinformation. This app is particularly helpful for underclassmen who have less experience on campus and no idea where to look for room information. It condenses all of the scattered information into one place so that necessary information is easily accessible and students do not have to carve out time to scour the internet and ask countless people for an idea of what rooms may look like.

Information about rooms was populated in the database based on data gathered from a recent student project with the Residential Life Office, including photos, occupancy, floor, features, etc., and additional information can be found by speaking to other Bryn Mawr College students. The room objects are created, retrieved, updated, and deleted through HTTP requests, and all data is viewable via the iOS frontend. With consideration of the limited timeframe, we developed the app on a small scope of only certain rooms from three dorm buildings, so as to put more emphasis on technical functionality and performance.

Since the app contains detailed photos of each dorm room, it is intended to be used only by current Bryn Mawr students. We carried out an authentication system with Bryn Mawr emails in the implementation period, with which students can input their school email, and only after the app verifies the email will users have access to the app. Users are generally not expected to be exposed to any risks in terms of security and privacy using our app, since the input email addresses are not saved and cannot be accessed by anyone from the server/database side, including the developers.

Lastly, we designed the app interface to be accessible to the most number of users by using clear visual and textual cues, as well as a color palette with sufficient color contrast.

3 Design



(Fig. 1 Architecture diagram: how the components relate to each other, and how they will communicate and interact.)

Our project design is composed of three main components: database, web server, and client-side iOS Application. First, our data source is a MongoDB NoSQL database, storing collections of room and user objects. The database communicates with the file storage service, Amazon S3, in order to retrieve URL references of the uploaded room photos, and then store them in the associated room objects. Next, the database interacts with our second component, the web server. Our web server is built upon Node.js and uses the Express framework. It is responsible for the backend of our project, connecting the user-facing application to data from the database. The web server is connected to the database using a Mongoose connection. Mongoose is an Object Data Modeling (ODM) library for MongoDB and Node.js, managing relationships between data, providing schema validation, and allowing translation between objects in code and their representation in MongoDB. The web server executes RestAPI calls to get the room and user data from the database.

Finally, our third component is an iOS mobile application called *Room-o-Pedia*. This is the user-facing component, which is responsible for displaying room data and allowing users to interact easily with the data. The app is built using the SwiftUI toolkit written in the Swift programming language. The iOS app connects to the web server using an Alamofire connection. Alamofire is a widely used HTTP networking library also written in Swift. Alamofire requests utilize the web server API calls in a clean and easy-to-read way. Together, these three major components allow us to accomplish our solution.

We decided on this project design because our goal is to present dorm room data in the most accessible way. Thus, we first considered user-facing application options: web app, Android mobile app, or iOS mobile app. A mobile app is the most portable and quickly available to users. We decided on an iOS mobile app over an Android app based on our assumption that more BMC students use iPhones. Based on our app's main features of searching and filtering, it made sense to use MongoDB for our database in conjunction with a web server. MongoDB has strong query performance with fast results and is highly scalable. We considered not including a web server to simplify the design, but using a web server improves our ability to perform

complex queries and retrieve data. In terms of connection, Mongoose and Alamofire are commonly used technologies that are known to work well with our other technologies. Our project design is effective as it was carefully considered from all angles, informed by best practices, and with help from our advisor.

4 Implementation

Our goals for the user interfaces of the app included an all-room listing view, individual room views, favorite room list view, and search and filter-by-tags functionality. Contrast color palette and text labels are utilized, taking user accessibility into consideration.









In terms of source control and technology set-ups, we monitored the project via two Github repositories, one for the backend web server, and another for the frontend Xcode project in Swift.

4.1 Node.js, MongoDB, and AWS S3

The MongoDB database is populated with the RoomWithPhotos Schema, each with various room features and an array of references in URL form to its photos in the Amazon S3 bucket. This model allows us to get all data associated with one room at once through the app quickly and conveniently.

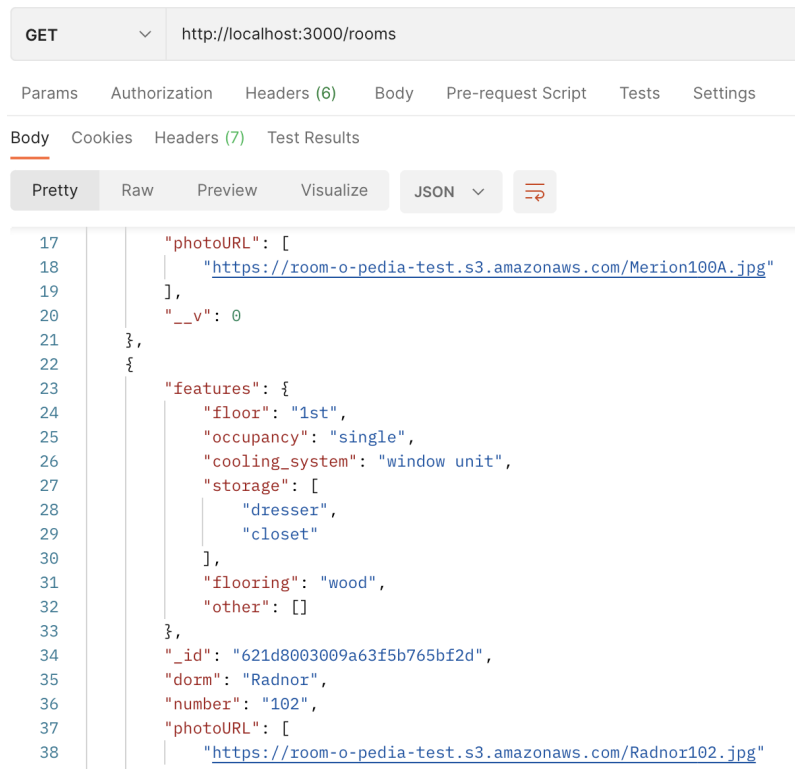
```
  _id: ObjectId("621d8003009a63f5b765bf2d")
  dorm: "Radnor"
  number: "102"
  features: Object
    floor: "1st"
    occupancy: "single"
    cooling_system: "window unit"
    > storage: Array
      flooring: "wood"
    > window_directi... : Array
    > other: Array
  photoURL: Array
    0: "https://room-o-pedia-test.s3.amazonaws.com/Radnor102.jpg"
```

(Fig 2. Example room object currently stored in MongoDB)

<input type="checkbox"/>	Name ▲	Type ▼	Last modified ▼	Size ▼
<input type="checkbox"/>	 Merion100.jpg	jpg	February 28, 2022, 21:10:37 (UTC-05:00)	3.1 MB
<input type="checkbox"/>	 Merion100A.jpg	jpg	February 28, 2022, 15:31:26 (UTC-05:00)	3.1 MB
<input type="checkbox"/>	 NewDorm208.jpg	jpg	February 28, 2022, 21:20:01 (UTC-05:00)	3.1 MB
<input type="checkbox"/>	 NewDorm305.jpg	jpg	February 28, 2022, 21:22:49 (UTC-05:00)	3.3 MB
<input type="checkbox"/>	 Radnor101.jpg	jpg	February 28, 2022, 21:13:02 (UTC-05:00)	3.2 MB
<input type="checkbox"/>	 Radnor102.jpg	jpg	February 28, 2022, 21:07:58 (UTC-05:00)	3.0 MB
<input type="checkbox"/>	 Radnor103.jpg	jpg	February 28, 2022, 21:09:06 (UTC-05:00)	3.2 MB
<input type="checkbox"/>	 Radnor201.jpg	jpg	February 28, 2022, 21:18:32 (UTC-05:00)	3.0 MB

(Fig 3. Example photos of rooms currently stored in Amazon S3 bucket)

The database is connected to the Node.js server via Mongoose, allowing us to create a room, retrieve all rooms, update information or delete a specific room from the server side as needed. On the backend side, *multer* is used to handle multiple image file uploads. As a room is created, the uploaded photos in form-data will automatically be stored in the S3 bucket, with their locations returned as an array to be saved in MongoDB. Then the connection between the database and our iOS frontend is handled via Alamofire in the APIFunctions class (details on this in Section 4.3).



```

17      "photoURL": [
18        |   "https://room-o-pedia-test.s3.amazonaws.com/Merion100A.jpg"
19      ],
20      "__v": 0
21    },
22    {
23      "features": {
24        "floor": "1st",
25        "occupancy": "single",
26        "cooling_system": "window unit",
27        "storage": [
28          |   "dresser",
29          |   "closet"
30        ],
31        "flooring": "wood",
32        "other": []
33      },
34      "_id": "621d8003009a63f5b765bf2d",
35      "dorm": "Radnor",
36      "number": "102",
37      "photoURL": [
38        |   "https://room-o-pedia-test.s3.amazonaws.com/Radnor102.jpg"
39      ]
40    }
41  ]

```

(Fig 4. Endpoint testing: GET request interacts with the database, returns info about all rooms)

POST

http://localhost:3000/create_room_with_photo

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

<input checked="" type="checkbox"/>	features[storage][]	closet
<input checked="" type="checkbox"/>	features[flooring]	carpet
<input checked="" type="checkbox"/>	features[other]	
<input checked="" type="checkbox"/>	photoURL	NewDorm305.jpg

Body

Cookies

Headers (7)

Test Results

Pretty

Raw

Preview

Visualize

JSON

```

1  {
2    "dorm": "New Dorm",
3    "number": "305",
4    "features": {
5      "floor": "3rd",
6      "occupancy": "single",
7      "cooling_system": "central",
8      "storage": [
9        "dresser",
10       "closet"
11     ],
12     "flooring": "carpet",
13     "other": [
14       ""

```

PUT

http://localhost:3000/add_photoUrl/621d837d009a63f5b765bf3d

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Body

Cookies

Headers (7)

Test Results

Pretty

Raw

Preview

Visualize

JSON

```

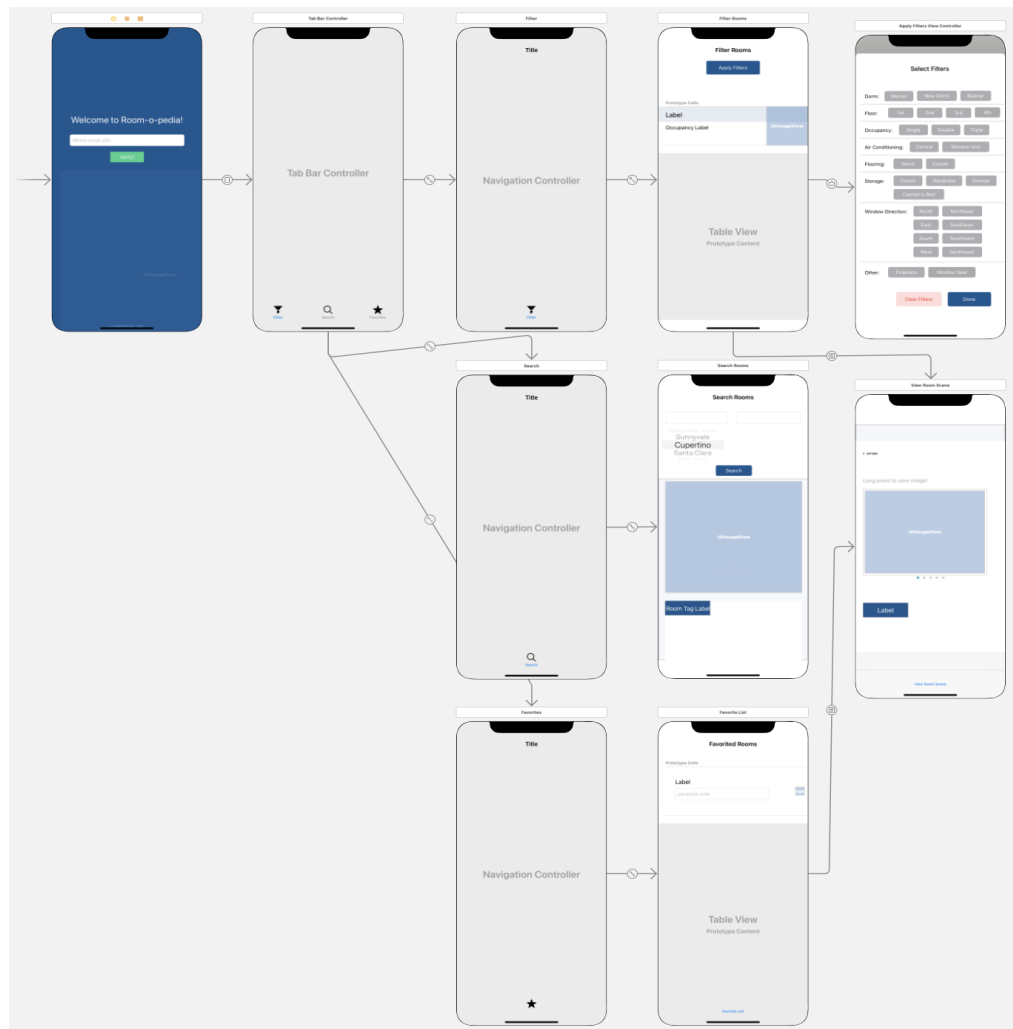
1  {
2    "room": {
3      "features": {
4        "floor": "3rd",
5        "occupancy": "single",
6        "cooling_system": "central",
7        "storage": [
8          "dresser",
9          "closet"
10       ],
11       "flooring": "carpet",
12       "window_direction": [
13         "southeast"
14       ],
15       "other": [
16         ""
17     ]
18     },
19     "_id": "621d837d009a63f5b765bf3d",
20     "dorm": "New Dorm",
21     "number": "305",
22     "photoURL": [
23       "https://room-o-pedia-test.s3.amazonaws.com/NewDorm305.jpg",
24       "https://room-o-pedia-test.s3.amazonaws.com/NewDorm305(1).jpg"
25     ],
26     "_v": 0
27   },
28   "message": "Room updated."
29 }

```

(Fig 5 & 6. Endpoint testing: POST and PUT requests to interact with the database, create a new room object, or update a currently available room)

4.2 iOS Application

An overview outlining the architecture and every scene of the iOS app can be found on our Main storyboard (Fig 7.). The initial arrow points to the first screen, the login page. After a user's email is verified, the app automatically transitions into displaying all the rooms in the database, with an option to filter. From that screen, users can navigate to two other tabs on the bottom of the screen, one being the search room feature and the other being the favorite room feature. In the search tab, students can scroll to their desired dorm, type in a room number, and click the search button. This will then yield the pictures and tags of the desired room, given that it is in the database. From the view all rooms page, students are able to favorite rooms, which will then show up under the favorite tab. Besides the login page, students are able to navigate these tabs from any screen within the app. From the view all rooms page, students can also click into any singular room, which will then segue into a screen to display one room's information including dorm, number, pictures, and tags. With the overall flow of the app in mind, we will describe in detail each feature of the app.



(Fig 7. Main storyboard on XCode showing all screens)

4.2.1 Feature: Email Verification

The application begins with a login screen, which asks for a Bryn Mawr College email. This screen is linked to the main storyboard and the LoginViewController, in which *authorizeEmail()* reads in the email typed by users, checks if it is in a valid Bryn Mawr email format, and either lets them proceed to the app or sends an alert based on the check result.



In LoginViewController:

viewDidLoad(): This function is executed when the screen first loads. It applies formatting to elements on the screen, including User Email UITextField and the Verify UIButton.

authorizeEmail(_sender: UIButton): This function is triggered by the Verify UI Button. It checks whether the user's text input in User Email UITextField matches the syntactical pattern of Bryn Mawr College student email addresses. If the text input does not match, then an alert is displayed. If the text input does match, then the app segues to the View All Rooms screen.

4.2.2 Feature: Loading View

Upon passing the Bryn Mawr College email verification, the app will automatically navigate to the filter rooms screen. Initially, a loading view with a buffering animation covers most of the screen. During this time, API calls are used to fetch and load all room data from the database.



Filter Count Text View
Rooms Count Text View

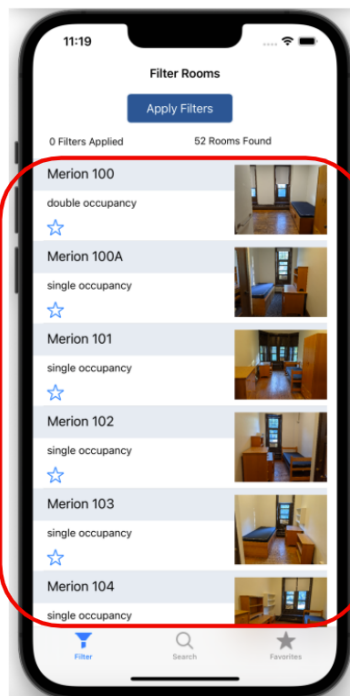
Loading View

In ViewController:

tableView(): Includes tasks of counting the initial object array and sets the values for the two TextViews.

configureLoadingView(): Adds a white UIView as subview of the screen, then calls a UIActivityIndicatorView to start animating. If loading time is reached, it removes all loading subviews from screen to display information.

viewDidLoad(): This function is executed when the screen first loads. It calls above function and set timer of 3 seconds to stop loading.



Rooms fetched

In APIFunctions:

Alamofire (AF) is used to communicate with the server.

fetchRooms(): Sends an AF request to retrieve JSON data from the GET endpoint. It then passes encoded data to **updateArray()** to be used in ViewController.

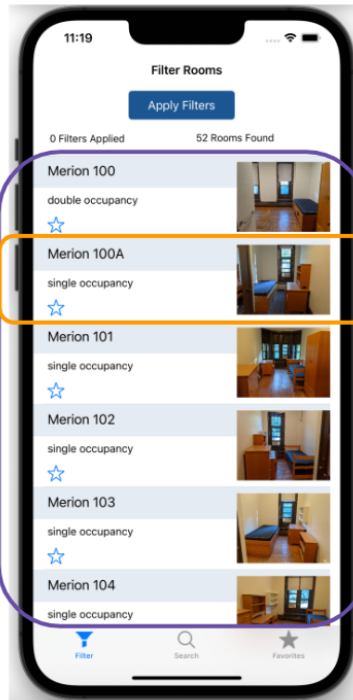
In ViewController:

updateArray(): Belongs to the extension DataDelegate of ViewController class. This function decodes the JSON data into an array of Room objects based on the Codable struct declared, sorts the array alphabetically by dorm names and numbers, and finally calls a reload of data on screen to present all data.

All room data are retrieved and loaded in the background of the animating loading subview. After the subviews are removed, they are presented as shown.

4.2.3 Feature: View All Rooms

The Filter Rooms screen displays a table of all rooms in the database (no filters applied). The screen features custom UITableViewCells called RoomTableViewCell, showing dorm name, room number, a preview photo, and favorite stars of each room. Modified Swift tableView methods aid in making sure the cells contain the data, pictures are loaded, etc...



Rooms Table Cell

Rooms Table

In RoomTableViewCell:

RoomTableViewCell is a custom UITableViewCell in a UITableView that allows us to customize the cell components.

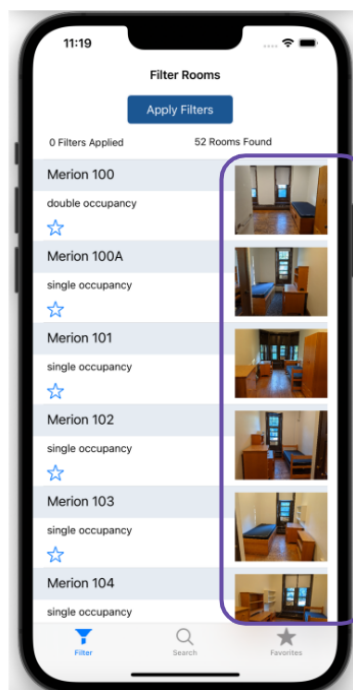
clickFav(): Allows adding room at this cell into the favorite list (*details in 4.7*)

setSelected(): Configures the view for a cell's selected state, triggers a segue to ViewRoomViewController that passes the data for the chosen room

In ViewController:

tableView(cellForRowAt): Populates each RoomTableViewCell with a different room fetched from the backend, displays features dorm, room number, occupancy, preview image, and favorite status.

tableView(numberOfRowsInSection): Sets the number of rows in Rooms UITableView to reflect the *rooms.Array* or *filteredRooms.Array* count if any filters are selected.



Preview images

In ViewController:

Cache class is declared with a static variable of type *NSCache<NSString, UIImage>()*. It allows for caching and quick displaying of images that have been loaded once anywhere in the app.

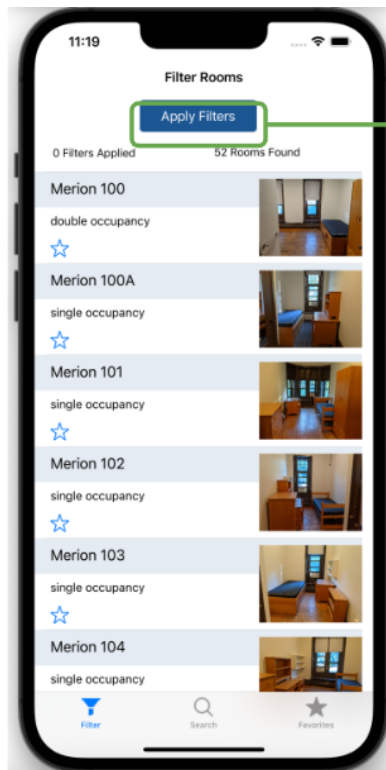
tableView(cellForRowAt): Includes asynchronous check whether an image is cached before loading data directly from the photo URL via **loadFrom()**. This function also prompts the caching of image data via **setObject()**, which takes an UIImage and stores in cache under key of dorm name and number on first encounter.

In ViewRoomViewController:

loadFrom(): Checks if a photo URL is valid, then returns its data under the type UIImage.

4.2.4 Feature: Filter

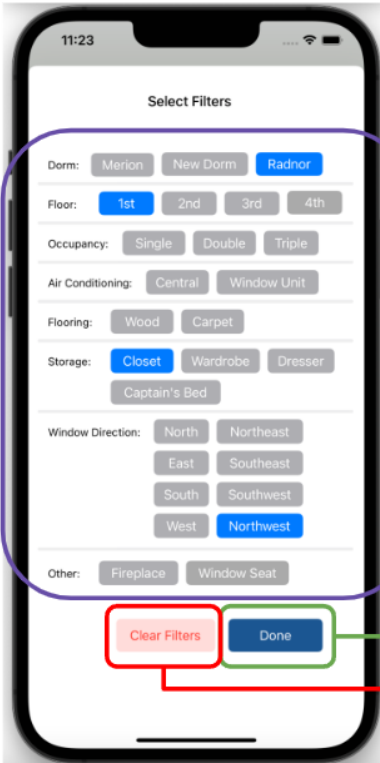
A major feature of our app is the filter-by-features functionality used with the list view of all rooms. Upon clicking the Apply Features button, a Select Feature popover screen with multiple different room feature options is displayed. This screen enables users to narrow down the list of suitable rooms according to their needs. After the user picks the criteria for filtering and clicks Done, they are returned to the Filter Rooms screen which updates to show only rooms that match their criteria.



Apply Filters Button

In ViewController:

prepare(): This function assists with segues between application screens. With this function, Apply Filters UIButton has its segue destination set to the ApplyFiltersViewController. When the user taps this button, the Select Filters screen will be displayed, and the *currentFilters* list object will be passed to the corresponding view controller, ApplyFiltersViewController.



Buttons for Each Filter

Done Button

Clear Button

In `ApplyFiltersViewController`:

`viewDidLoad()`: This function is executed when the screen first loads, setting up the feature filter buttons to reflect the current option selections listed in the *currentFilters* list.

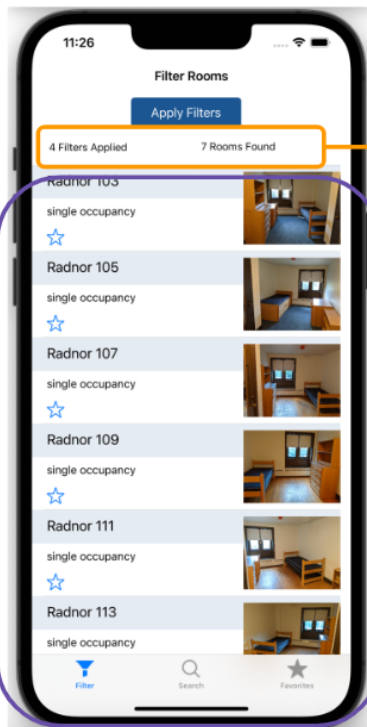
The logic for each filter button is relatively the same, and determined by individual functions following the naming convention of **`tapped<featureType><featureOption>()`**.

`tappedFloor1st()`: This function is one example of how the feature filter buttons are implemented. By tapping the button, the user can select or unselect a feature option, and their selections are appended to a *currentFilters* list object.

`tappedClear()`: When the user taps the Clear Filters UIButton, all filters are cleared from the *currentFilters* list, and all filter buttons are deselected (greyed out).

`prepare()`: This function sets the segue destination as the Filter Rooms screen, and passes the *currentFilters* to ViewController.

`tappedDone()`: The Done UIButton executes the segue.



Filter Count Text View and Rooms Count Text View

Rooms Table

In ViewController:

`unwind(_ seg: UIStoryboardSegue)`: This function formats the segue that returns from the Select Filters screen. It calls on the **`reloadVC()`** function.

`reloadVC()`: This function reloads all data on screen according to the *currentFilters* list passed from the Select Filters screen. It calls the **`filterRooms()`** function, which updates the *filteredRoomsArray* (described more below). It also sets the Filter Count Text View to reflect the number of filter options applied, and setting the Room Count Text Views to reflect the number of rooms that match the filter criteria (displayed in the UITableView.)

`filterRooms()`: This function filters the list of all Room objects to only include rooms that match the feature filter criteria provided by the *currentFilters* list, and saves these rooms to the *filteredRoomsArray* object.

4.2.5 Feature: View One Room

While browsing the listings, students can click on one room to enter an individual room view screen. Each individual room view contains the dorm room number, an image slider of the room's photos loaded via its S3 URL, and a collection of tags describing features of the rooms, all displayed with high contrast colors. As an additional feature, users can long-press to save images from the app to their device. Every feature tag cell within the collection can also be clicked to show filter results if that feature option was used to filter all rooms.



In `TagCollectionViewCell`:

`TagCollectionViewCell` is a custom `UICollectionViewCell` in a `UICollectionView` that allows us to customize the cell components.

In `ViewRoomViewController`:

`viewDidLoad()`: This function is executed when the screen first loads. It applies formatting to the dorm and number label and initial setup, like setting data sources and delegates for the tags and photos.

`collectionView(cellForItemAt):` Populates each `TagCollectionViewCell` with a different features fetched from the backend, displaying all features for the particular room.

`collectionView(didSelectItemAt):` Triggers the unwind segue back to view controller, setting the selected `TagCollectionViewCell` as a filter.

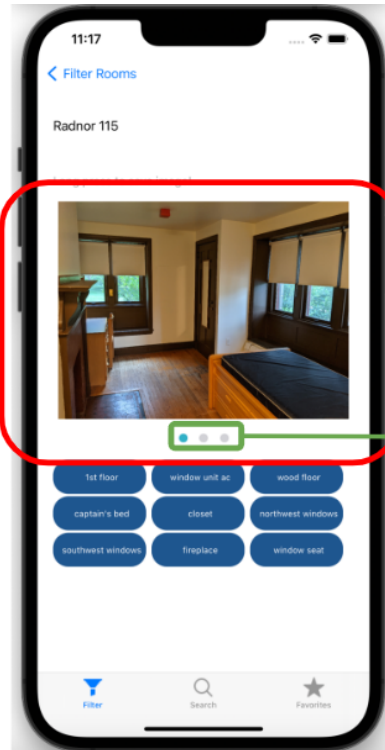


Photo Collection Slider

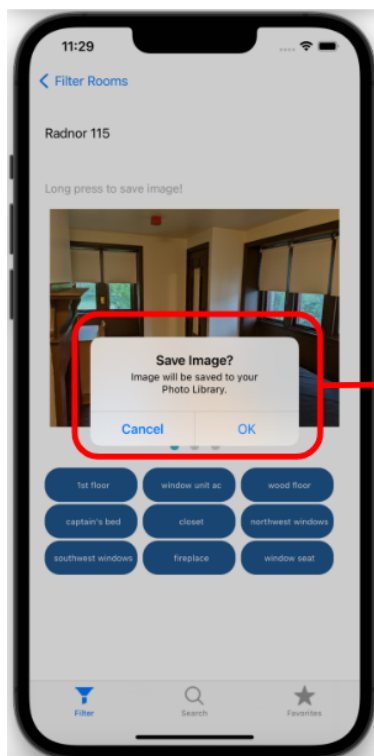
Page Controller

In ViewRoomViewController:

collectionView(cellForItemAt): Populates each PhotoCollectionViewCell with image data from cache or loaded from URL of the specific Room object, displaying all photos available for the particular room.

scrollViewDidEndDecelerating(): Recognizes pauses as a user slides through the photos, changing the teal-colored indicator to the corresponding the current photo page.

viewDidLayoutSubviews(): Scales the page controller to 1.5 times the default sizes.



Save Image Alert

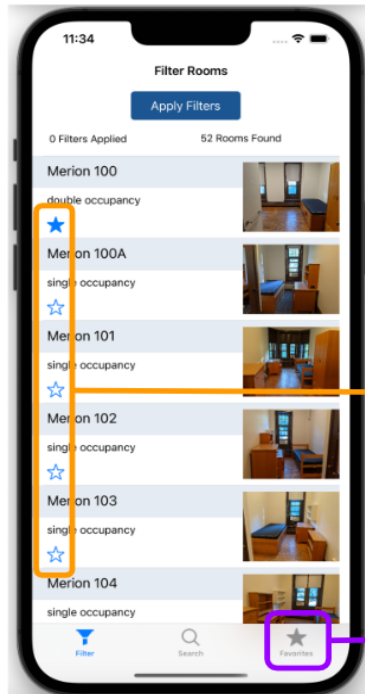
In ViewRoomViewController:

longPressed(): Recognizes the long press gesture of a user within the bounds of the UIImage, and reacts by getting the image data corresponding to the chosen indexPath. This function then displays an alert that prompts the user to choose between "Cancel", which does nothing, or "OK", which calls **UIImageWriteToSavedPhotoAlbum()** to write image data to the device Photo Album with allowed Photo Privacy Settings.

If the settings are turned off, the function calls **image()** to terminate the saving process before displaying another message notifying the user to change their phone settings.

4.2.6 Feature: Favorite

From Filter Room screen, a user can choose to click on a star button to select or deselect rooms as their favorites. Each time the user navigates to the Favorites screen, they can see a newly updated list of their favorite rooms. Additionally, in each of the favorite room cells, there is a personal note option for users to make notes about each room. This data is saved to the device.



Star buttons

Favorite tab

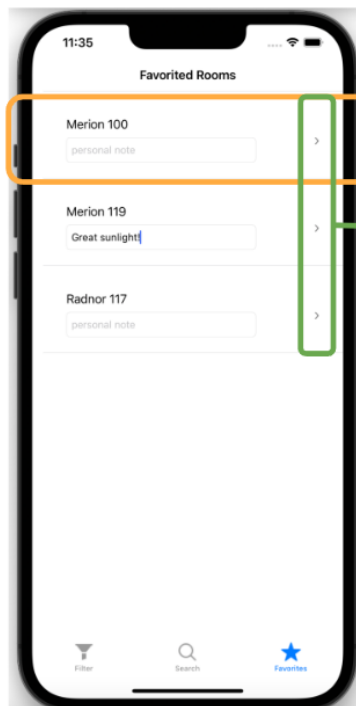
In ViewController:

Array of rooms corresponding to each starred cell is stored in *UserDefaults*, allowing data to persist on the same device even when the app is closed.

favButtonPressed(): Checks if the star button of a cell was previously unselected, adds its information to the *UserDefaults* array if yes, or removes from the array if no. This function changes the button appearance to “fill” and “unfill” correspondingly.

In RoomTableViewCell:

clickFav(): Calls **favButtonPressed()** to set up the initial appearance of star buttons as the app loads the next time by cross-checking with the *UserDefaults* array.



Saved room

Display Info Buttons

In FavoriteTableViewCell:

FavoriteTableViewCell enables customization of the table cells to include corresponding dorm and number information, as well as personal notes.

In FavoriteViewController:

viewWillAppear(): Reads data from *UserDefaults* array and populates the customized cells accordingly with room information. This function allows data to be refreshed every time the Favorite tab is clicked to sync the Favorite list with the *UserDefaults* array.

prepare(): Passes room data of the chosen cell to View One Room screen through a *UIStoryboardSegue*

4.2.7 Feature: Search

The search rooms screen can be accessed simply by clicking the search tab on the bottom bar. Users can scroll through a list of dorms and select the dorm they are looking for. Then, they type in a specific room number and click search. Given that the entered room is in the database, the screen will then load the picture of the room along with the corresponding feature tags.



Room Number
Text Field

Dorm Picker

Search Button

In SearchRoomViewController:

viewDidLoad(): This function is executed when the screen first loads, using `APIFunctions.functions` to the `rooms.Array` of all rooms

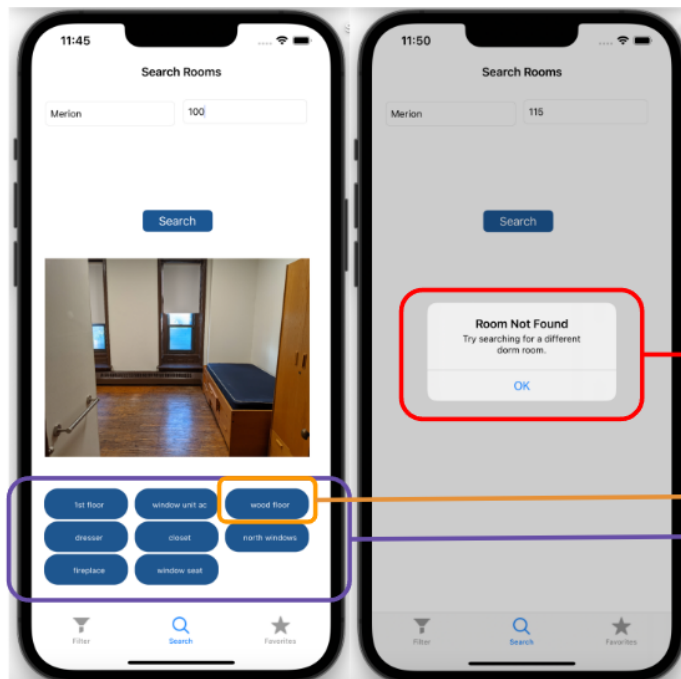
numberOfComponents(in pickerView): Sets picker quantity in Dorm UIPickerView.

pickerView(numberOfRowsInComponent): Sets row quantity in Dorm UIPickerView.

pickerView(titleForRow): Sets title for Dorm UIPickerView rows.

pickerView(didSelectRow): Saves the selected dorm from Dorm UIPickerView.

pickerView(viewForRow): Formats Dorm UIPickerView.



Alert

Feature Tag
Cell

Feature Tags

In SearchRoomViewController:

getRoom(): This function is executed when the Search UIButton is selected. It uses the selected dorm name from Dorm UIPickerView and the inputted room number from Room Number UITextField to look for a specific room. If a room is found, then the room photo and feature tags are displayed. If the room is not found, then an alert is displayed.

This screen uses same collectionView functions to display the feature tags as the singular room view that uses ViewRoomViewController.

5 Evaluation

In order to evaluate our solution, we conducted an in-person user study in which users were asked to perform specific tasks on information gathering before assessing the usability of each app feature. The study served to represent how users might typically interact with the app and provide insight into current app functionality and possible improvements.

5.1 Methodology

The study was carried out with the participation of a total of 9 active Bryn Mawr College students, all of whom were seniors living on campus and had participated in room draw at least three times. Although the target audience for this app consisted of students who would remain active by the 2023 room draw, especially first-years who could not travel to campus prior to the room draw process, our tested users were essential in providing opinions on the current system and necessary improvements. Their experience with and various perspectives on room draw would allow us to thoroughly assess the effectiveness of the app in solving its persistent issues, including the lack of centralized data and unequal accessibility to room information. Additionally, they would also be the most helpful in evaluating the usefulness of possibly integrating this app into the official room draw procedures.

The specific tasks given to our users are listed below:

1. How many rooms are available in the app?
2. Are there any rooms in Radnor with carpet?
3. Does Merion 202 have AC?
4. I want a first-floor room that has hardwood flooring and south or southeast facing windows. How many rooms are there with that?
5. In your opinion, how easy is it to find rooms that fit your criteria, and see the number of rooms that fit your criteria?
6. I will be living in Radnor 104 next year! What directions are the windows facing?
7. (Follow up to 6) What's another room that has windows facing the same direction?
8. In your opinion, how easy is it for you to find all rooms with the same window direction?
9. Check out the photos for Radnor 115. In your opinion, how easy is it to see all the photos of one room?
10. Are you able to save a photo of a room onto the device?

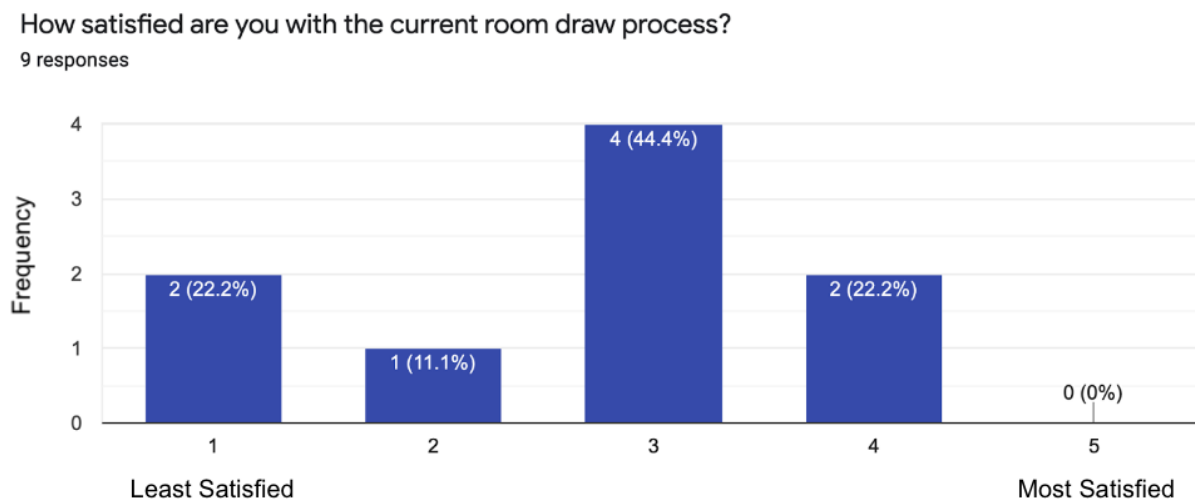
11. Find room info about New Dorm 305. Add New Dorm 305 to your favorites list. In your opinion, how easy is it to save a room to your favorites?

Our tasks were designed with an aim to quantify the usability of the app. Apart from that, we collected qualitative data on whether users think this app would be helpful during the room draw process (i.e. does it solve the existing problem). We also asked users to rate their experience with the app, name their likes, dislikes, and any problems about room draw set out by the app that they felt were or were not addressed.

The participants tested the app using the XCode simulation feature and filled out each question in the survey as they went through the app. While a user was executing the tasks, a team member observed and noted down their performance in order to evaluate the intuitiveness of the app as well as specific user tendencies as users encountered the app for the first time. Observations include details such as how difficult it was for them to complete the task or which approach they took to access a piece of information. Based on this, we were able to evaluate whether the app was effective at disseminating information, if there were any app design flaws that would need further work, or if any additional problems arose out of our solution..

5.2 Results

Participants rated their satisfaction with the current room draw process on a scale of 1 to 5. The results are shown below (*Fig 8*).



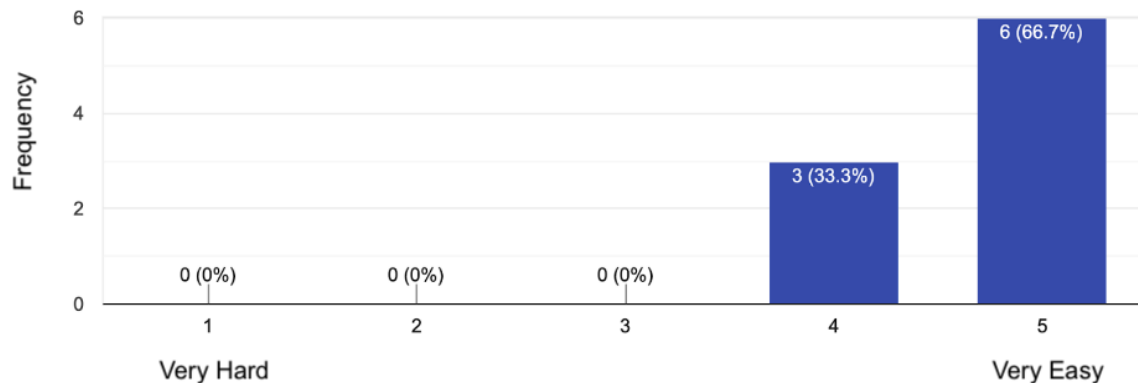
(*Fig 8. Participants' current room draw rating*)

For each of the tasks, the results were as follows:

1. 100% of the participants correctly reported a total of 52 rooms in the app.
2. 100% of the participants correctly answered that there were rooms in Radnor with carpeted floors.
3. 100% of the participants agreed that Merion 202 had air-conditioning, which was the correct answer.
4. 88.9% (8/9) of the participants correctly reported 12 rooms on the first-floor that had hardwood flooring and south or southeast facing windows. 11.1% (1/9) of the participants incorrectly reported 7 rooms.
5. On a scale of 1 to 5 in finding and seeing the number of rooms that fit specific criteria, participants' ratings are shown below (*Fig 9*).

5. In your opinion, how easy is it to find rooms that fit your criteria, and see the number of rooms that fit your criteria?

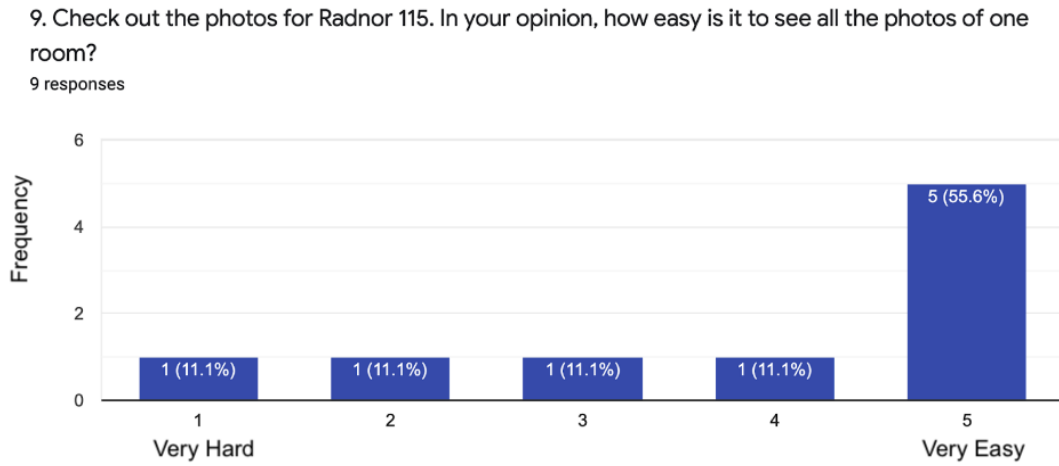
9 responses



(*Fig 9. Ease of finding filtered rooms and the number of rooms, ranked by participants*)

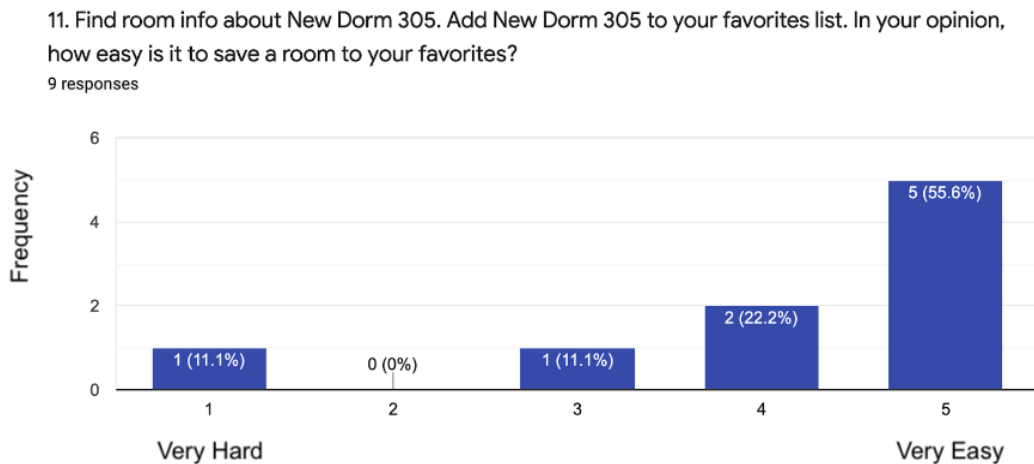
6. 88.9% (8/9) users correctly found that Radnor 104 had southeast facing windows. 11.1% (1/9) of the participants incorrectly reported north facing windows.
7. 44.4% (4/9) participants answered Merion 202 as a room that has windows facing in the same direction as Radnor 104, 22.2% (2/9) found New Dorm 305, 22.2% (2/9) of the participants reported Radnor 106, and 11.1% (1/9) of the participants answered 12. All of these answers, except 12, were correct.
8. On a scale of 1 (very difficult) to 5 (very easy) in finding all the rooms with the same window direction, 100% of the participants rated 5.

9. On a scale of 1 to 5 in seeing all the photos of one room, participants' ratings are shown below (Fig 10).



(Fig 10. Ease of viewings all of one room's photos, ranked by participants)

10. 88.9% (8/9) users were able to save a photo of a room onto the device while 11.1% (1/9) of the participants were unable to save a photo.
11. On a scale of 1 to 5 in saving a room to favorites, participants' ratings are shown below (Fig 11).



(Fig 11. Ease of favoriting a room, ranked by participants)

For the user experience survey, the results were as follows:

1. On a scale of 1 to 5 in their experience with the app, participants rankings are shown below (*Fig 12*).

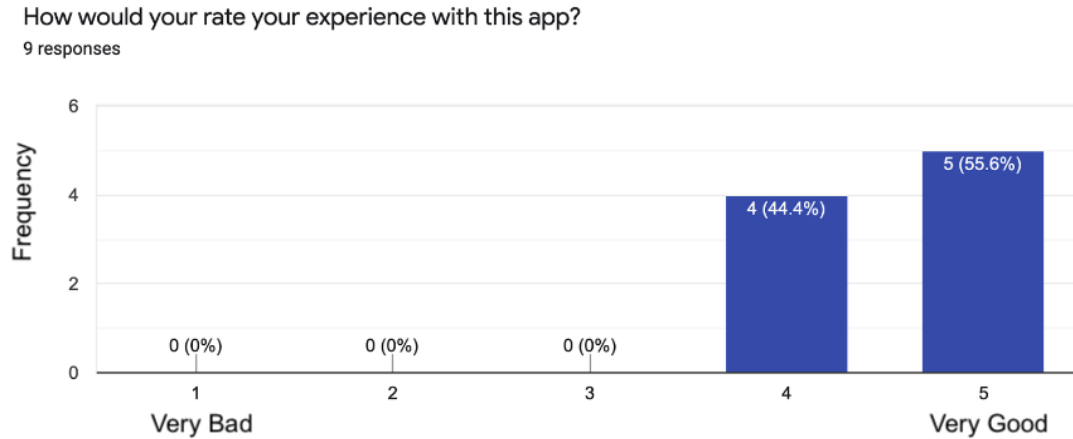
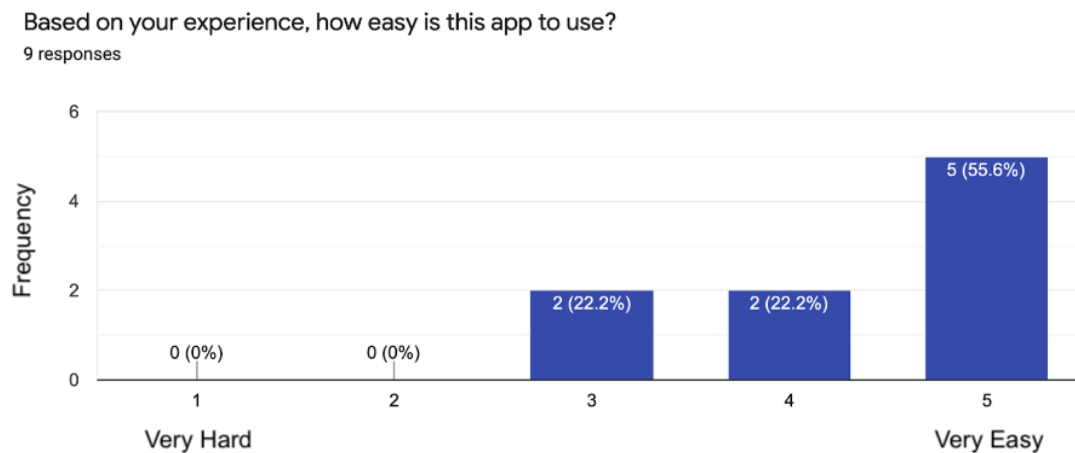


Fig 12. User experience, ranked by participants)

2. On a scale of 1 to 5 in their experience in how easy it is to use the app, participants' rankings are shown below (*Fig 13*).



(Fig 13. Application ease of use, ranked by participants)

5.3 Discussion

The study results allowed us to return to the problem statement and carefully assess how well the initial goals were accomplished, whether we have solved the problems on the user side as well as on our side as developers.

One of our main goals for developing this app was to provide equal and easy access to the room database and all available room information. This was tested through various tasks of reporting the total number of rooms available, number of rooms that matched specific criteria, and whether or not a particular room had a wanted feature. Overall, all participants were able to browse the full database, and most reported the correct answers to given tasks. This proved that the app succeeded in ensuring equal footing for students in terms of information dissemination. There was one noticeable outlier for each of the three tasks, tasks 4, 6, and 7, which we strongly speculated to have been caused by a misunderstanding of the evaluation question, such as choosing only one filter instead of selecting all the criteria mentioned.

Secondly, we aimed to create a centralized database that would allow students to filter and search through rooms in order to make informed decisions on what rooms they would like to live in. All participants in the user study agreed that the app enabled easy filtering by specific needs, as well as a fairly convenient favorite feature to record suitable rooms as “favorites”. However, expectations for the search function fell rather short, as it was not equipped with all the available features offered in individual room views. This was reflected well in the general reported usability and intuitiveness: participants who mostly used filters to navigate through the survey tended to record a better experience than those who approached tasks by searching for one room. The comprehensive filter feature and sorted list view were mentioned by several participants as being very easy to use and able to replace search, prompting the question of whether search should be swapped with another feature.

Lastly, we intended for the app to bridge the gap of misinformation about room details caused by scattered sources of information and not-to-scale floor plans. Through the system of tags in individual room views, users were able to instantly extract key information about a room such as air-conditioning system, window directions, or flooring. Additionally, the photo slider in each room view provided multiple photos at different angles that aided participants in visualizing the rooms fairly effectively. Most users were also able to save images to their device's Photo Library.

Comments on the app were very positive, with many participants agreeing that the design was intuitive, clear, and easy to follow. The app provided different ways to complete the same task, which made it quite accessible. Among all the features, filtering was inarguably the most warmly welcomed, as it catered for specific needs and enabled searching for all available rooms based on one's preferences. The Favorite screen was also mentioned a few times as a great extra

feature. On the other hand, the search bar and tags were functional but not extremely efficient. Nevertheless, many agreed that they would be very useful after some minor improvements.

We have also considered potential threats to the validity of our evaluation. The most difficult factor for us to evaluate was the convenience and accessibility of our solution. Since we decided not to put our iOS app on the App Store, we tested the app with participants using the XCode simulation feature, allowing us to run the local version of the app on an iPhone device. Thus, we were unable to assess whether people would choose to download this app onto their phone, and actually use the app as if it were on their phone.

On a whole, the experience with the app was recorded to be very pleasant, with a high score of usability on the first use and significant applicability. To assess the effectiveness of the app in enhancing the room draw experience for Bryn Mawr students, we asked users to rate their satisfaction with the current room draw process and the hypothetical room draw system which would incorporate this app, on a scale of 1 to 5, with 5 being “very satisfied”. The former averaged at 2.7 out of 5, while the latter reached an average of 4.2 out of 5. This significant difference in satisfaction level clearly highlighted the benefits of *Room-o-Pedia*!

6 Related Work

One major attempt to solve the issue of a stressful room draw at Bryn Mawr College is the Google Drive folder of dorm room pictures. The database of photos is definitely very useful and helpful; however, it is not very accessible, as students need information and links to the folder, most likely through friends or word of mouth. This is difficult for Bryn Mawr College freshmen in particular, who do not have vast connections throughout the school. The Google Drive folder still does not include every dorm room on campus and can be difficult to navigate, as there is no way to search through all the rooms by feature. Even if a student has an idea about what dorm they would like to live in, they must go through every room and dig for information themselves, i.e. see if the room has wood or carpet flooring, if there are many windows, etc. Before they open each individual room in the Google Drive folder, there is no real way to know what the room will look like.

Other attempts to make the room draw process easier for students include various Google spreadsheets and forms that rely on students to input information themselves about the rooms they have lived in. Since all the data is self-inputted, it tends to be very scattered and varied, as students are able to be as detailed or as vague as they desire. These resources only consist of the rooms of people who the spreadsheet has reached and who are willing to enter their information. Over the course of our college careers, there were about five separate attempts using this method of collecting data. All of these attempts were started by students, unaffiliated with Residential Life, who just wanted to help and make information more accessible for underclassmen. However, in each database, there had never really been above 30 entries, and the resources failed to be widespread enough for it to make a big difference, only reaching certain friend groups and friends of friends.

Our project combines various resources like the Google Drive folder and spreadsheets to solve the major issue and to address the downfalls of each individual attempt. Instead of needing

a specific link to a resource, a student can just download our app and gain access to a database full of information about dorm rooms at Bryn Mawr. This is more accessible and easier for students, saving them the hassle of searching online and asking as many people as they can just to gain a better understanding of the room they want to live in the following year. Our application does not only save them this time in searching, but it also displays the data in an easy-to-use format that allows students to make informed decisions, cutting down much of the stress related to room draw.

In brainstorming app features and user interface, we consulted a couple of apps and websites that we use in our daily lives, such as Yelp, Zillow, and Apartments.com. Yelp relies heavily on user reviews to help inform other users of the restaurant/establishment experience. We thought this was very helpful in gaining a better understanding and making a more knowledgeable and reassuring choice. Our app includes student reviews of dorm rooms so that students can hear multiple perspectives and learn information that cannot be seen just through pictures or tags. We also looked to Zillow and Apartments.com to understand what people look for when they are searching for housing, such as what features are important and what format is best to represent housing-related data. Of course, these apps do not try to solve our particular problem, but we were inspired by the overall user experience and the ways in which these apps attempt to solve their own problems. Bringing together our personal experiences of room draw and using these apps, our app effectively addresses our problem statement and the stress-inducing room draw process.

7 Conclusion

Our iOS application, *Room-o-Pedia*, assists students in making more informed decisions during the room draw process by centralizing dorm room data on a convenient platform. Students can view detailed information and photos of Bryn Mawr College dorm rooms. The app's search and filter features make sifting through the countless dorm rooms much quicker and easier. Our solution is unique as it is the first time this much information has been documented, organized, searchable/filterable, and all centralized in one location.

Through our user study to evaluate the app, it was clear that *Room-o-Pedia* is a beneficial solution. On average users rated their satisfaction with the current room draw process as 2.7 out of 5, with 5 being very satisfied. After completing tasks that replicate how a typical student might use our app, we asked the users to again rate their satisfaction with the room draw process, but this time consider a hypothetical revised room draw process where Bryn Mawr College has implemented *Room-o-Pedia* as part of their process. This time, users rated their satisfaction with this hypothetical process as 4.2 out of 5. The difference in satisfaction is over 30%. While only a hypothetical question, it is clear that our participants believe that this app would greatly improve their experience with the room draw process. All users thought that the app would be helpful for Bryn Mawr College students going through the room draw process. When asked for comments, one student wrote: "Res Life should pay the students for this app. Would save a lot of headaches on both sides of the process (admin and students) - from a student who has assisted in facilitating room draw." Users were also able to complete the evaluation tasks with a high degree of accuracy. The evaluation findings indicate that our application does succeed in assisting students

through the room draw process. Students would find the application helpful and our implementation provides dorm information in an accessible and centralized way.

7.1 Future Work

While we have succeeded in creating a solution in *Room-o-Pedia*, there is still potential future work that could improve our solution. Based on the user study, we have found a few areas of our implementation that could benefit from clarification. For instance, our evaluations showed that some users were confused with the Search Room screen. While the dorm information resulting from a search looked similar to the individual room views, the functionality was different, leading to some confusion. By expanding on the functionality of the Search Room screen, possibly by linking search results to the existing individual room views, we could improve both the clarity and usability of our app. A few more potential changes that could increase usability would be to allow favoriting of rooms from the individual room view and to redesign the clickable room feature tags to make them more intuitive.

In addition to these improvements, there are new features that could enhance the current state of our app. A future direction for *Room-o-Pedia* could be expanding the interactivity and creating a platform for students to share their own knowledge of the dorm rooms. We would implement additional features that would allow users to upload new images of dorm rooms, add public comments and feedback on a room view, or post questions on a room view to be answered by other students. These features would also require further implementation of user accounts for ease of use, as well as admin accounts for verification purposes. By requiring students to create an account with their college email, the app can be maintained as a safe online environment.

In conclusion, we can confidently say that our solution addressed the lack of accessible and centralized information on dorm rooms available to Bryn Mawr College students. Through user evaluation testing of our iOS application, *Room-o-Pedia*, we can see that our solution improves upon Bryn Mawr College's existing room draw, providing a resource that assists students in making more informed decisions throughout all stages of the room draw process. As one user answered when asked what they liked about the app, "what's not to?"

8 References

Apartments.com Rental Finder. CoStar Realty Information Inc., Version 12.0.2, 2022. *Apple App Store*, <https://apps.apple.com/us/app/apartments-com-rental-finder/id319836632>

Yelp: Food, Delivery & Reviews. Yelp, Inc., Version 22.12.0, 2022. *Apple App Store*, <https://apps.apple.com/us/app/yelp-food-delivery-reviews/id284910350>

Zillow Real Estate & Rentals. Zillow, Inc., Version 15.19.1, 2022. *Apple App Store*, <https://apps.apple.com/us/app/zillow-real-estate-rentals/id310738695>