

**Team 11: Product Backlog**

Pedro Del Moral Lopez, Sam Fellers, Ben Huemann, Michael Pike, Kaiwen Wei

# **Problem Statement**

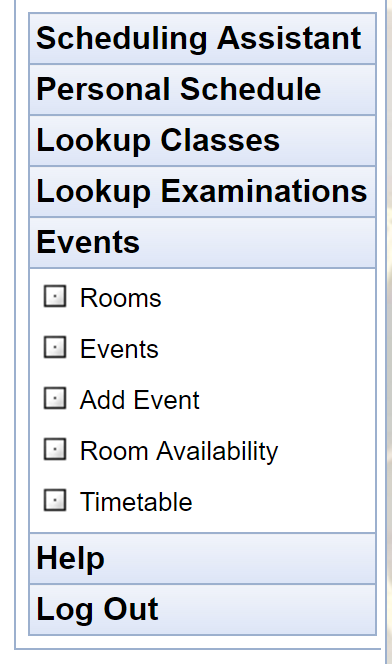
Reserving a room for private study, or sharing rooms for group studying is a key and essential tool for academic success. However students today are confronted with a substandard and clunky tool for reserving rooms. Our project aims to provide a easy-to-use and intuitive tool for checking availability of rooms, reserving rooms, and marking rooms as “shareable”.

# **Background Information**

Immediately upon accessing the current room reservation system, users are presented with the following screen:  
  


As you can tell, this tool is not immediately apparent where you can reserve a room. It provides too many unnecessary functions that could be separated into a standalone dedicated application.

Once a user chooses Events, they are presented with the following options:



This is entirely inefficient. A user cannot tell what rooms are available at a glance, and immediately reserve the room at the same time. They’d have to open multiple windows, one for looking at what rooms are even *capable* of being reserved, then another for room availability, and then one more for reserving time in that room.

The current system also provides no information beyond the room number for understanding *where* in the building the room is located. This could be easily improved as well with an integrated map.

This can be greatly improved, and we aim to redesign the current system with a more functional, single page solution.

# **Environment**

## **Frontend Environment**

The frontend will be written in Javascript, jQuery, nodeQuery, and HTML5, using AngularJS and ImageMapster, a jQuery plugin for interactive maps of the floor. We will use Google’s Material Design and Bootstrap3 to maintain a consistent visual aesthetic.

## **Database Environment**

We will be using the MySQL databases Purdue provides all students. Since we will only be storing very limited information on users and reservation times, with a very normal, structured, data set.

## **Version Control and Document Keeping Environment**

We will be using GitHub as our method of source control as it is what we are all most familiar with. It also provides a good system of metrics for measuring work distribution and contribution.

## **Backend Environment**

The server and backend will run on a Node.js server. Node provides several high-level libraries that will be crucial to communicating with the front end, interfacing with the database, and maintaining various state information necessary to normal application functionality. We will be using Express on top of Node as a lightweight web framework. Passport.js will be used as a middleware for user authentication.

# **Functional Requirements**

|  |  |  |  |
| --- | --- | --- | --- |
| Backlog ID | Functional Requirement | Hours | Status |
| 00 | As a user, I would like to login/logout. | 3 | Sprint 1 |
| 01 | As a user, I would like to register an account. | 6 | Sprint 1 |
| 02 | As a user, I would like to reserve a room. | 12 | Sprint 1 |
| 03 | As a user, I would like an easy way to view all current room statuses. | 50 |  |
| 04 | As a user, I would like to view when a room is reserved. | 4 |  |
| 05 | As a user, I would like a limit on the number of hours one person can reserve a room. | 3 |  |
| 06 | As a user, I would like the option to make a room I have reserved shareable/unmark the room as shareable. | 2 | Sprint 1 |
| 07 | As a user, I would like to be able to cancel reservations I have made. | 4 |  |
| 08 | As a user, I would like to change my password | 4 |  |
| 09 | As a user, I would like to recover my password. | 12 |  |
| 10 | As an administrator, I would like to be able to block a room from being reserved. | 10 |  |
| 11 | As an administrator, I would like to cancel reservations. | 4 |  |
|  | Total: | 98 |  |

# **Non-functional Requirements**

**Security**

Security in our application is crucial as it would be affiliated with the university. Our login and registration systems must protect user information beyond an industry standard level of encryption. All user information stored in our database should be inaccessible to anyone other than the user. All requests to the API will be authenticated before execution.

**Usability**

Our application should not only look good, it should feel good. This is the main issue with the current system for reserving any of the available spaces on campus, it’s too bulky and hard to use. Upon login you shouldn’t need any instruction to know how to reserve a room, it should be built in a way that is intuitive for the average user.

**Scalability**

Our application should be able to be scaled up with ease. It should be able to handle a 50 user load. In the event too many requests are coming in for our server to handle, it should be gracefully dealt with. Not only will it be easily scaled to handle larger traffic loads, our application will be designed with intention of scaling to new buildings and rooms around campus.

# **Use Cases**

## **Case 00: Login**

|  |  |
| --- | --- |
| **Action** | **System Response** |
| 1. Click login button  3. Type in username  4. Type in password  5. Click submit | 2. System opens login modal  6. System verifies log-in information and displays appropriate success or failure screen. |

## **Case 00: Logout**

|  |  |
| --- | --- |
| **Action** | **System Response** |
| 1. Click logout button | 2. System closes session  3. System displays success screen |

## **Case 01: Register an account**

|  |  |
| --- | --- |
| **Action** | **System Response** |
| 1. Click register  3. Type in email  4. Type in password  5. Confirm password  6. Click submit  10. Verify Email | 2. System opens registration modal.  3. System displays success screen.  7. System verifies email. 8. System sends verification email.  9. System adds user to database.  11. System marks account  information as verified |

## **Case 02: Reserve a room**

**Action System Response**

1. Click on a room either on 2. System opens modal with  
   the map or on the timetable next time table information  
   to the map. for the day.

3. Click on an open time slot . 4. System verifies user is   
 logged in and requests   
 user to log-in if   
 necessary

5. System opens reservation   
 dialog.

6. Select time slot.

7. Click submit. 8. System verifies time   
 slot selection does not   
 conflict with   
 pre-existing   
 reservations, and does   
 not over-reserve for   
 user budget.

9. System stores the registration information in the database.

10. System edits user budget in database.

10. System closes modal and   
 shows success screen.

11. System sends   
 confirmation email.

## **Case 03: Check availability of all rooms**

**Action System Response**

1. View summarized room availability  
   On scrolling container next to map.

## **Case 04: Check availability of single room**

**Action System Response**

1. Select room on map. 2. System opens modal with   
    time table information   
    for the day

3. Press off modal/exit button. 4. Modal disappears.

## **Case 05: Limit reservations to avoid abuse**

**Action System Response**

1. User attempts to reserve room 2. System checks how many   
    hours the user currently   
    has reserved.

3. If the user has 6 hours   
 of reservation across   
 any number of rooms, the   
 system will deny the   
 request until already   
 reserved time is   
 cancelled or used.

## **Case 06: Mark rooms as shareable**

### **Method 1:**

**Action System Response**

1. User follows steps to reserve room 2. System follows steps to   
    reserve room.

3. User checks “Shareable” checkbox 4. System marks the  
 before submittal reservation as   
 shareable and displays   
 the room as shareable to   
 other users.

### **Method 2:**

**Action System Response**

1. User clicks “My Reservations” button 2. System changes web page   
    to page displaying   
    active and future   
    reservations associated   
    with user’s account.

3. User selects reservation from list. 4. System pulls up modal   
 with summary of   
 reservation, and options   
 for editing reservation.

5. User checks “Shareable” checkbox

6. User clicks “Submit Changes” 7. System marks the   
 reservation as shareable   
 and displays the room as   
 shareable to other   
 Users.

## **Case 06: Unmark room as shareable**

**Action System Response**

1. User clicks “My Reservations” button 2. System changes web page   
    to page displaying   
    active and future   
    reservations associated   
    with user’s account.

3. User selects reservation from list. 4. System pulls up modal   
 with summary of   
 reservation, and options   
 for editing reservation.

5. User unchecks “Shareable” checkbox

6. User clicks “Submit Changes” 7. System unmarks the   
 reservation as shareable   
 and displays the room as   
 shareable to other   
 Users.

## **Case 07: Cancel reservation**

**Action System Response**

1. User clicks on “My Reservations” 2. System changes web page  
   Button. to page displaying  
    active and future  
    reservations

3. User clicks on a reservation from 4. System brings up modal  
 The list of available reservations. with summary of   
 reservation information  
 and options for editing  
 the reservation.

5. User clicks on the   
 “Cancel Reservation” button.

6. User clicks the “submit changes” 7. System removes  
 Button. reservation and marks   
 room as available for   
 other users.

## **Case 08: Change password**

**Action System Response**

1. User clicks “Change password” button 2. System will navigate to   
 change password page.

3. User inputs current password.

4. User inputs new password.

5. User confirms new password. 6. System verifies that

New password meets

Password requirements.

7. System verifies current

Password.

8. System changes user’s

Password in database.

9. System displays

Confirmation page.

## **Case 09: Recover password**

**Action System Response**

1. User clicks “Recover password” 2. System will navigate to   
    password recovery page

3. User types in email to confirm  
 Identity

4. User hits submit. 5. System verifies email   
 and sends password   
 recovery email with   
 verification pin

6. User correctly inputs 7. System logs in user and   
 verification text. prompts user to change   
 password

## 

## **Case 10: Block a room from being reserved**

|  |  |
| --- | --- |
| **Action** | **System Response** |
| 1. Administrator selects a room on map.  3. Administrator clicks the “Block Room” button.  4. Administrator clicks the “Submit Changes” button. | 2. System opens modal with time table information for the day.  5. System verifies that the user is an administrator.  6. System emails anyone with the room reserved that it has been blocked.  7. System displays screen that the room has been successfully blocked.  8. System updates the room as blocked in the database. |

## **Case 10: Unblock a room**

|  |  |
| --- | --- |
| **Action** | **System Response** |
| 1. Administrator selects a blocked room on map.  3. Administrator clicks the “Unblock Room” button.  4. Administrator clicks the “Submit Changes” button. | 2. System opens modal with time table greyed information for the day.  5. System verifies that the user is an administrator.  6. System emails anyone with the room reserved that it has been blocked.  7. System displays screen that the room has been successfully blocked.  8. System updates the room as blocked in the database. |

## 

## **Case 11: Cancel a user’s registration**

|  |  |
| --- | --- |
| **Action** | **System Response** |
| 1. Administrator clicks on “Reservations” button.  3. Administrator clicks on a reservation from the list of current reservations.  5. Administrator clicks on the “Cancel Reservation” button.  6. Administrator clicks the “Submit Changes” button. | 2. System changes web page to page displaying all active and future reservations.  4. System brings up modal with summary of reservation information and options for editing the reservation.  7. System verifies that the user is an administrator.  8. System removes reservation from the database and marks the room as available for other users.  9. System emails the user with the reservation, notifying them it has been canceled. |

## 