

Design Document

Building Management

Team Members:

Alex Lam (<https://github.com/lama449>) (Team Leader),

Eric Zielonka (<https://github.com/zielonka-eric>),

Michael Turner (<https://github.com/mt9u>),

Cameron Thatcher (<https://github.com/cthatchr>),

Kevin Miranda (<https://github.com/mirandak1>)

Monica Mahon (<https://github.com/monicamahon0>)

Dylan Chow(<https://github.com/dchowXII>)

Github: <https://github.com/lama449/SeniorProject>

Slack: penguins-z9p7067.slack.com

Table of Contents

High Level Description	2
Problem Solving Approaches	3
Web Pages	4
Home Page	4
Room Confirmation Page	6
Reservations Page	7
Maintenance Request Pop-Up	7
Facility Creation Page	8
Room Creation Page	10
Room Update Page	11
Management Page	12
Maintenance Page	13
Profile Page	14
User Login Pop-Up	15
User Registration Page	16
Groups Page	16
User Maintenance Request Overview Page	19
Web Page Navigation	21
Flow Diagram	22
Technology Stack	24
Backend Information	25
Authentication and Security	26
Input and Output	27
Database Schema	28
Restful Endpoints	31

High Level Description

The purpose of this application is to provide businesses with a way to manage their facilities by reserving meeting spaces, listing the amenities of their meeting spaces, and allowing users to report any maintenance issues within the building.

The facility owner will create an account and register their facilities. Registration will require them to input the buildings their facility contains, and input the rooms in each building. Owners may also input their own custom room by defining their own amenities that their rooms contain.

Facilities may be public or private, and that is determined by the facility owner at the time of registration. If a facility is public, any user may create an account and make a reservation for any room in that facility. If a facility is private, the user must join by entering an access code that is provided to the facility. If the owner chooses to, users may be sorted into groups to give an additional layer of privatization in the building. Reservations may only be made by users in a group that is authorized to access that room. Reservations will be assigned on a first come, first served basis. If the need arises, facility admins may cancel or change reservations.

The Building Management system will also have the option to submit and manage maintenance requests. If a user sees a problem with a room, they may submit a maintenance request describing the issue to the maintenance team. Admins will review these from a "Maintenance" page and track the progress of each request. Requests will be organized as submitted, in progress, and completed.

Problem Solving Approaches

A challenge that arose while designing the data storage for our app was how to store users in a way that would allow for facilities to add additional restrictions on room reservations internally. Initially, we were just making facilities private or public, which made any room in a private facility able to be reserved by a member of that facility. Upon further consideration, the group decided that we would like to offer facilities greater access control to their rooms in the event that certain areas should be restricted to employees with certain access or if access was only supposed to be granted to a select few departments. We were unsure of how to reorganize our database structure in a way that segments user storage into access control lists per facility. By adding an additional table, Groups, we can assign users to a group, and assign groups to the facility they belong to. When a room is created, we will provide the approved group lists which will segment which users are allowed to make reservations in that room in a facility.

Web Pages

Home Page

The *Home Page* is where the users will be directed when they go to use the web application at the beginning of their use. As seen below, the home page has two different text fields, three when logged in, to enter a facility and a zip code to search for a specific type of facility or known facility. If the Facility is private a user can enter an Access Code to search for the hidden facility. After searching for a facility, results pertaining to that search will be displayed. After the results appear, the user will be able to click on a facility's name to go to the *Facility Page* to view that specific facility's buildings. In addition to searching, the user can also login on this page by clicking the button at the top. Or they can proceed to a different page via the menu bar above the search boxes.

Home Profile Reservations Management Maintenance Submissions Welcome Alex! Log Out

Search for a Facility

Search for a Facility by name

Search for a Facility Zip Code

Submit

OR

Search for a Facility by Access Code

Submit

Team Penguins Building Management Dec 2019

Buildings Page

The *Buildings Page* is where users can select the time slot and date according to what they need for a specific room. To change the date users would click the arrows to move the date forwards or backwards. If they want to find out the features they would hover over the names of the rooms and a pop-up will appear. If they want to select the time slot then they would click on the calendar to choose a range of time and hit the “Submit Time Slot” button. To correctly use the page, the user would need to be logged in. If the user is not logged in, they will see a message requesting the user to login.

The screenshot displays the 'Buildings Page' of a web application. At the top, there is a navigation bar with links: Home, Profile, Reservations, Management, Maintenance, Submissions, and a 'Welcome Alex!' message with a 'Log Out' button. Below the navigation bar is a 'Go Back' button. The main heading is 'Buildings Page:', followed by the instruction: 'Select the timeslot below (by clicking and dragging) and click the submit button to create a reservation for that time slot.'

The form for building details includes the following fields:

- Building Name:
- Building Phone Number:
- Building Description:
- Building Address Line 1:
- Building Address Line 2:
- Building Address City:
- Building Address State:
- Building Country:
- Building Zip Code:

Below the form is a button labeled 'Edit Building Information'. The main content area features a calendar for 'Sunday, Dec 15, 2019'. The calendar has a table with columns for time slots from 6:00 AM to 12:00 AM. The first column is labeled 'Rooms' and the second column is labeled 'Occupancy'. The rest of the columns represent the time slots. A button 'Add New Room' is located at the bottom left of the calendar area, and a button 'Submit Time Slot' is at the bottom right.

At the bottom of the page, there is a footer that reads: 'San Francisco Building Management Dec 2019'.

Room Confirmation Page

The *Room Confirmation Page* will allow the user to see the information for the room they scheduled. The user can cancel the reservation that they just created, go back to the buildings page, or go to their respective reservations page.

Home Profile Reservations Management Maintenance Submissions Welcome Alex! Log Out

Room Confirmation Page

You're all set! Below is the confirmation information for your reservation.
If for any reason you need to cancel your reservation, please do so on the Reservations page.

Address: 123 Test Rd., Test TX 55555

Start Time: Sun Dec 15 2019 12:00:00 GMT-0500 (Eastern Standard Time)

End Time: Sun Dec 15 2019 13:00:00 GMT-0500 (Eastern Standard Time)

Facility: FacilityTest

Building: BuildingTest

Room: TestRoom 456

Features: ["TV"]

[Cancel this Reservation](#)
[Go To Buildings Page](#)
[Go To Reservations Page](#)

Team Penguins Building Management Dec. 2019

Home Profile Reservations Management Maintenance Submissions Welcome Alex! Log Out

[Go Back](#)

Buildings Page:

Select the timeslot below (by clicking and dragging) and click the submit button to create a reservation for that time slot.

Building Name: Building Phone Number: Building Description:

Building Address Line 1: Building Address Line 2: Building Address City:

Building Address State: Building Country: Building Zip Code:

[Edit Building Information](#)

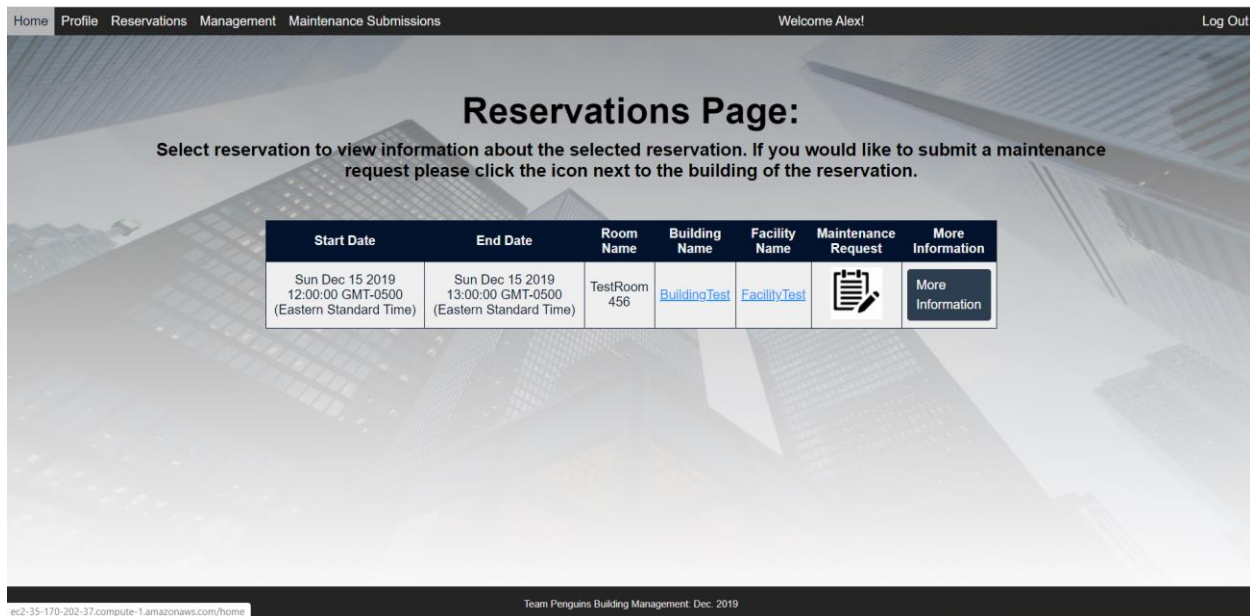
		Sunday, Dec 15, 2019																		
Rooms	Occupancy	6:00 AM	7:00 AM	8:00 AM	9:00 AM	10:00 AM	11:00 AM	12:00 PM	1:00 PM	2:00 PM	3:00 PM	4:00 PM	5:00 PM	6:00 PM	7:00 PM	8:00 PM	9:00 PM	10:00 PM	11:00 PM	12:00 AM
TestRoom 456	5																			

[Add New Room](#) [Submit Time Slot](#)

Team Penguins Building Management Dec. 2019

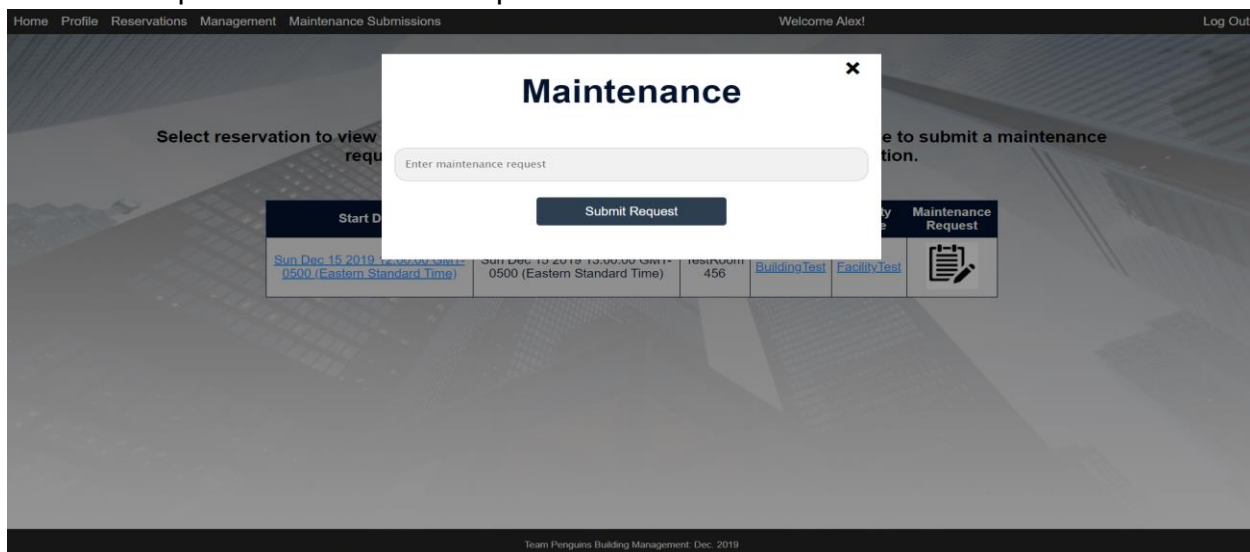
Reservations Page

The *Reservations Page* is where the users will be able to view the reservations that they have made in the past as well as future reservations coming up. When the user clicks on the more information button they will be redirected to the *Room Confirmation Page* where they can view the information for the reservation. Users can also submit a maintenance request by clicking on the maintenance request button corresponding to the reservation that need maintenance.



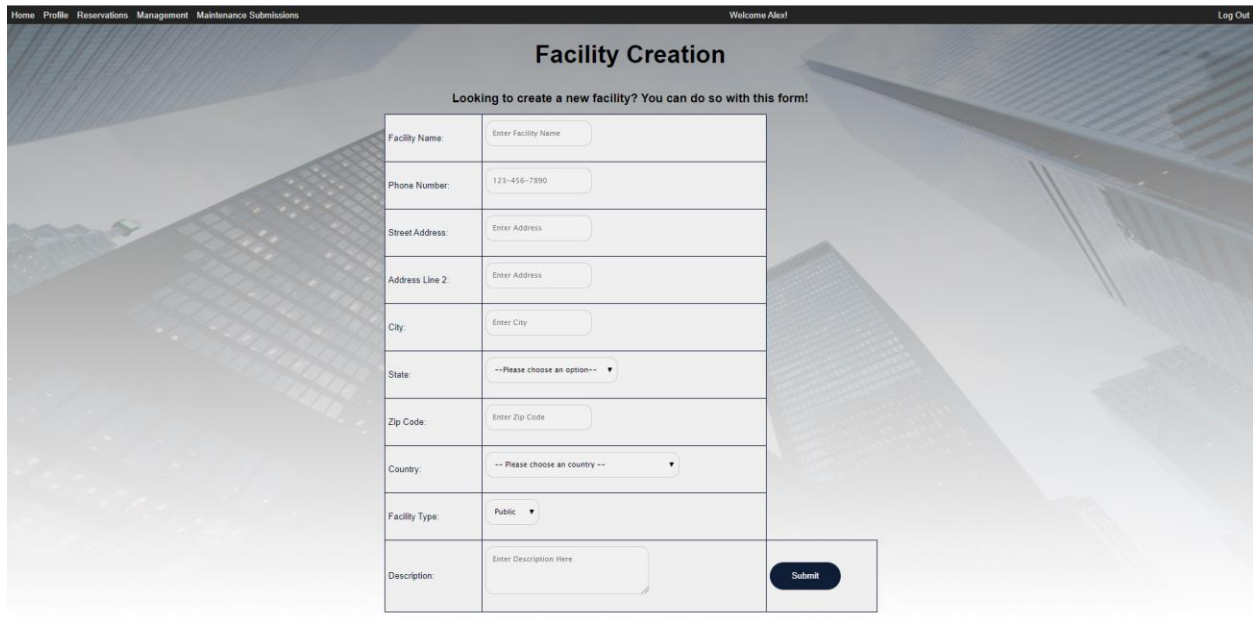
Maintenance Request Pop-Up

The *Maintenance Request Pop-Up* is used on the *Reservations Page* to submit a maintenance request. The user can enter text into the description section and click "Submit Request" to submit the request.



Facility Creation Page

The *Facility Creation Page* is where the users, namely facility owners, will be able to create new facilities. After entering the information about their facility in text boxes, they will click the “Create Facility” button to proceed to the *Facility Page* for that facility. Currently anyone is allowed to create a facility.

A screenshot of a web application's 'Facility Creation' page. The page has a dark header with navigation links: Home, Profile, Reservations, Management, Maintenance, Submissions, and a 'Log Out' button. The main content area has a light gray background with a city skyline image. The form is titled 'Facility Creation' and includes a sub-header: 'Looking to create a new facility? You can do so with this form!'. The form fields are: Facility Name (text box), Phone Number (text box with placeholder '123-456-7890'), Street Address (text box), Address Line 2 (text box), City (text box), State (dropdown menu with '--Please choose an option--'), Zip Code (text box), Country (dropdown menu with '-- Please choose an country --'), Facility Type (dropdown menu with 'Public' selected), and Description (text area). A 'Submit' button is located at the bottom right of the form.

Facility Creation	
Looking to create a new facility? You can do so with this form!	
Facility Name:	<input type="text" value="Enter Facility Name"/>
Phone Number:	<input type="text" value="123-456-7890"/>
Street Address:	<input type="text" value="Enter Address"/>
Address Line 2:	<input type="text" value="Enter Address"/>
City:	<input type="text" value="Enter City"/>
State:	--Please choose an option--
Zip Code:	<input type="text" value="Enter Zip Code"/>
Country:	-- Please choose an country --
Facility Type:	Public
Description:	<input type="text" value="Enter Description Here"/>
<input type="button" value="Submit"/>	

Building Creation Page

The *Building Creation Page* is where users, namely facility owners, can create buildings to link to their facility. Users complete the relevant required information of the form and then submit the information. If a user wanted to go back to the *Facility Creation Page* they would click the button to direct them back. Like the *Facility Creation Page*, this page is only visible to building/facility owners while they are logged into their specific account.

Building Creation	
Building Name:	<input type="text" value="Enter Building Name"/>
Phone Number:	<input type="text" value="123-456-7890"/>
Street Address:	<input type="text" value="Enter Address"/>
Address Line 2 (Optional):	<input type="text"/>
City:	<input type="text" value="Enter City"/>
State:	<input type="text" value="--Please choose an option--"/>
Zip Code:	<input type="text" value="Enter Zip"/>
Country:	<input type="text" value="--Please choose an option--"/>
Description:	<input type="text"/>
<input type="button" value="Submit"/>	

Team Penguin Building Management Dec. 2019

Room Creation Page

The *Room Creation Page* is where users, namely building/facility owners, will be able to create rooms for the buildings and facilities. By entering the relevant information for the room below into the text boxes the user can create the room according to how they need to. After clicking the “Submit” button, the user will be redirected to the previous page, the *Buildings Page*, where they can edit other rooms or choose to create new rooms.

Home Profile Reservations Management Maintenance Submissions Welcome Admin! Log Out

Room Creation

Room Name:	<input type="text" value="Enter Room Name"/>
Room Number:	<input type="text" value="Enter Room Number"/>
Room Capacity:	<input type="text" value="Enter Room Capacity"/>

Click the button to add an option at the end of the list.
Control+click (command on mac) to select multiple features.

TV Whiteboard Telephone Microphone	<input type="text" value="TV"/>	<input type="button" value="Add Feature"/>
Which groups have access to this room?		<input type="button" value="Submit"/>
admin default		

Team Penguins Building Management, Dec. 2019

Room Update Page

The *Room Update Page* is where users, namely building/facility owners, will be able to edit the rooms that they have created from the *Room Creation Page*. After clicking on a specific room on the *Buildings Page*, the user will be directed to this page wherein they can modify the room.

Home Profile Reservations Management Maintenance Submissions Welcome Admin! Log Out

Update Rooms

Room Name (Optional):	Room Demo
Room Number:	101
Room Capacity:	20

Click the button to add an option at the end of the list.
Control+click (command on mac) to select multiple features.

Which attributes does this room have:

TV
Whiteboard
Telephone
Microphone

Add Attribute

Which groups have access to this room?

admin
default

Submit

Team Pangaris Building Management Dec. 2018

Management Page

The *Management Page* is where users, namely building/facility owners, will be able to navigate to different areas to manage their facilities. As seen below, a user can see a list of facilities that they are an admin of, add a new facility, delete a facility, or go to the *Facility Page* for one of their facilities.

Home Profile Reservations Management Maintenance Submissions Welcome Alex! Log Out

Management Page

Select the facility you would like to view more information about below or click the add button to add a new facility.

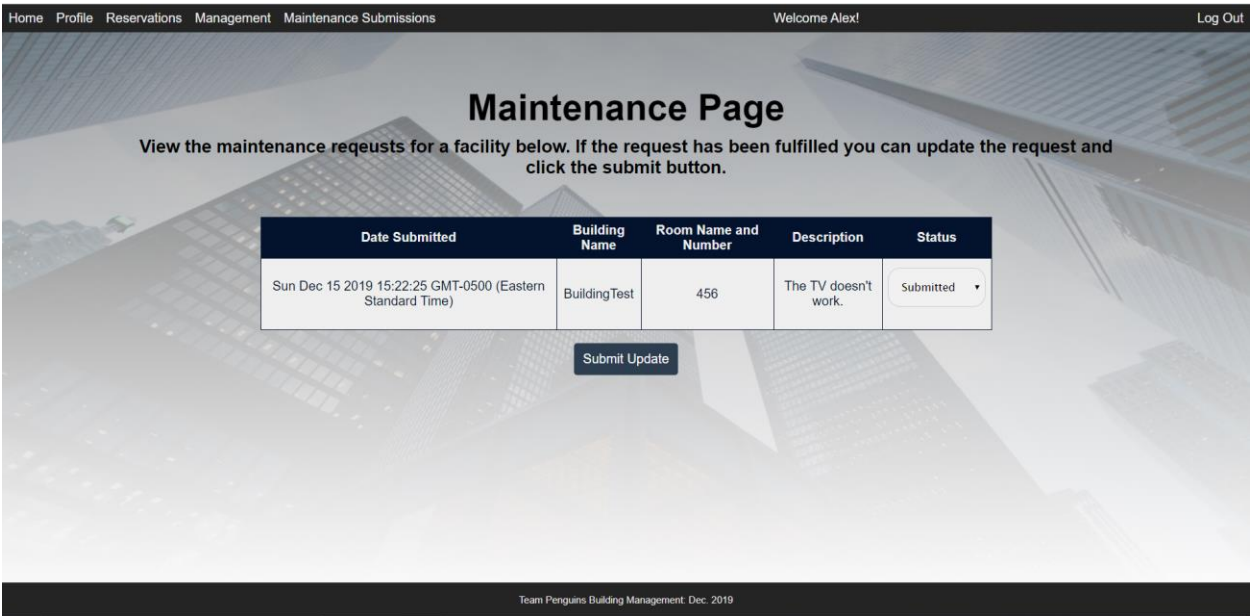
Facility	Location	Description	Phone Number	Access Code	Building Privacy	Delete?
FacilityTest	123 Test Rd., Test, TX, 55555	This facility is a test.	555-555-5555	IDnT4EmYp4	Public	

[Add A New Facility](#)

Team Penguins Building Management, Dec. 2019

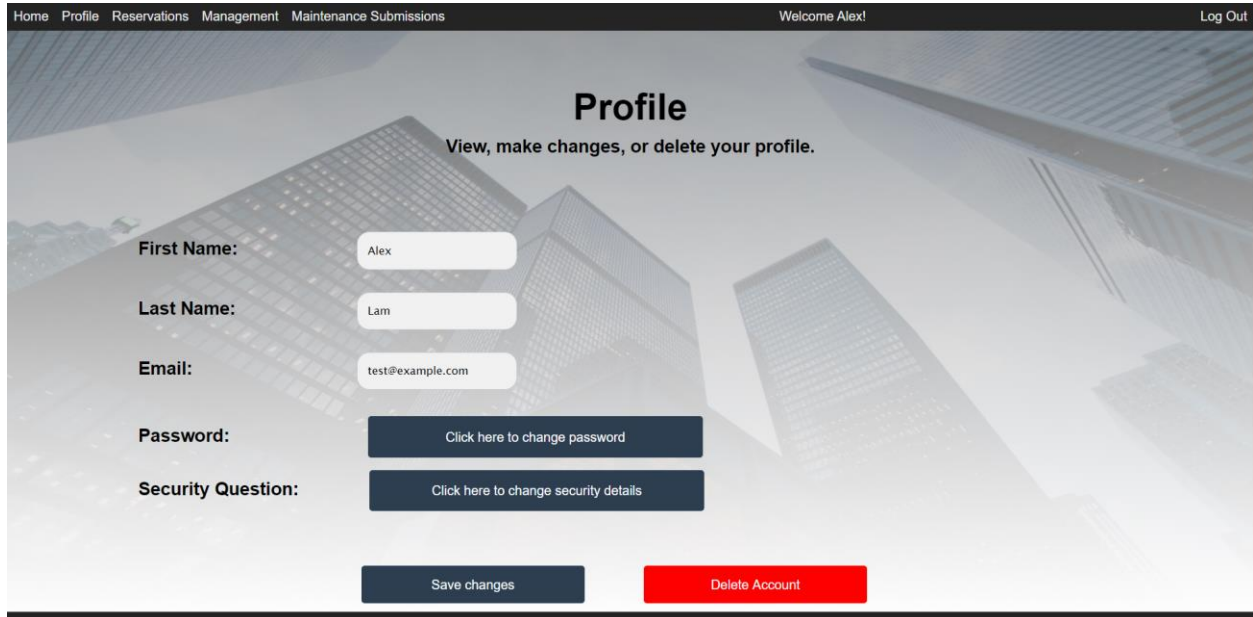
Maintenance Page

The *Maintenance Page* is where users, namely building/facility admins, will be able to view the requests for a facility. They will be able to see the date submitted, building name, room number, short description of what is wrong, and the status of the maintenance request. By using the dropdown menu for the status, they will be able to modify and update the maintenance request.



Profile Page

The *Profile Page* is where users will be able to edit their information that they put in at the time they registered.



The screenshot shows a web application's profile page. At the top is a dark navigation bar with links: Home, Profile, Reservations, Management, Maintenance Submissions, and a 'Welcome Alex!' message on the right. Below the navigation bar, the page has a light gray background with a faint city skyline. The main heading is 'Profile' in bold, followed by the subtitle 'View, make changes, or delete your profile.' The form contains five sections: 'First Name:' with a text input containing 'Alex'; 'Last Name:' with a text input containing 'Lam'; 'Email:' with a text input containing 'test@example.com'; 'Password:' with a dark button labeled 'Click here to change password'; and 'Security Question:' with a dark button labeled 'Click here to change security details'. At the bottom of the form are two buttons: a dark 'Save changes' button and a red 'Delete Account' button.

Home Profile Reservations Management Maintenance Submissions Welcome Alex! Log Out

Profile

View, make changes, or delete your profile.

First Name:

Last Name:

Email:

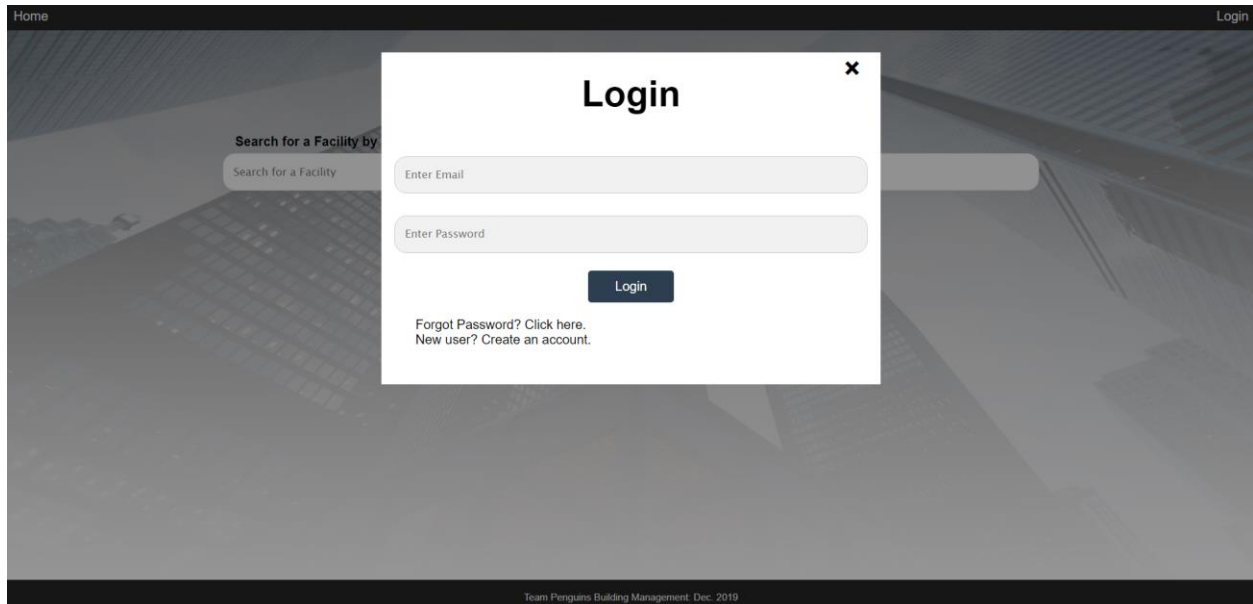
Password: [Click here to change password](#)

Security Question: [Click here to change security details](#)

[Save changes](#) [Delete Account](#)

User Login Pop-Up

The *User Login Pop-Up* will pop-up if they click the login button in the navigation bar. They are required to enter their email and password. If they are a new user they can click the link to create a new account. If they forgot their password, they can click the link “Forgot Password?”



The screenshot shows a web application interface with a dark header bar. On the left, there is a 'Home' link. On the right, there is a 'Login' link. Below the header, there is a search bar with the placeholder text 'Search for a Facility by'. A modal window titled 'Login' is centered on the screen. The modal has a close button (an 'x' icon) in the top right corner. Inside the modal, there are two input fields: 'Enter Email' and 'Enter Password'. Below these fields is a dark blue 'Login' button. At the bottom of the modal, there are two links: 'Forgot Password? Click here.' and 'New user? Create an account.' The background of the page is a grayscale image of a city skyline.

Home Login

Search for a Facility by

Search for a Facility

Enter Email

Enter Password

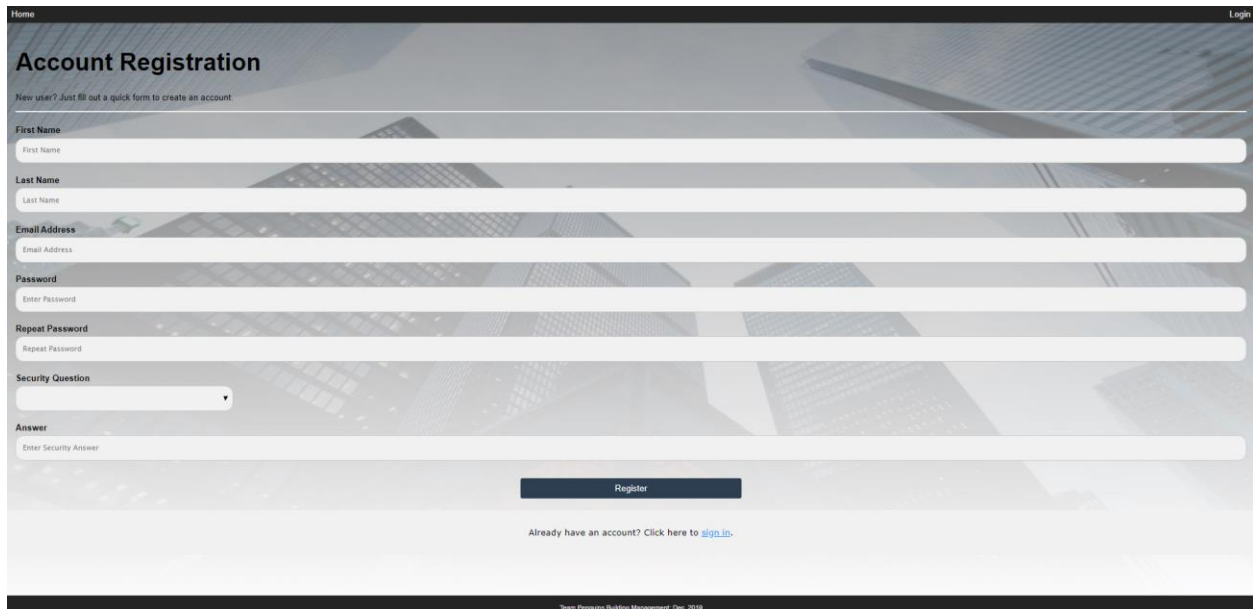
Login

Forgot Password? Click here.
New user? Create an account.

Team Penguins Building Management, Dec. 2019

User Registration Page

The *User Registration Page* allows users to create their account by entering their first name, last name, email, password, security question, and security question answer. Once they hit the “Register” button then they can access their account that was created on the site. They cannot create an account if their email already has an account associated with it.



The screenshot displays the 'Account Registration' page of a web application. The page has a dark header with 'Home' on the left and 'Login' on the right. The main title 'Account Registration' is prominently displayed, followed by a subtitle: 'New user? Just fill out a quick form to create an account.' The registration form consists of several input fields: 'First Name', 'Last Name', 'Email Address', 'Password', 'Repeat Password', 'Security Question' (a dropdown menu), and 'Answer'. Each field has a small placeholder text. Below the form is a dark 'Register' button. At the bottom, there is a link for users who already have an account: 'Already have an account? Click here to [sign in](#).' The footer contains the text 'Team Pangina Building Management Dec. 2018'.

Home Login

Account Registration

New user? Just fill out a quick form to create an account.

First Name
First Name

Last Name
Last Name

Email Address
Email Address

Password
Enter Password

Repeat Password
Repeat Password

Security Question
▼

Answer
Enter Security Answer

Register

Already have an account? Click here to [sign in](#).

Team Pangina Building Management Dec. 2018

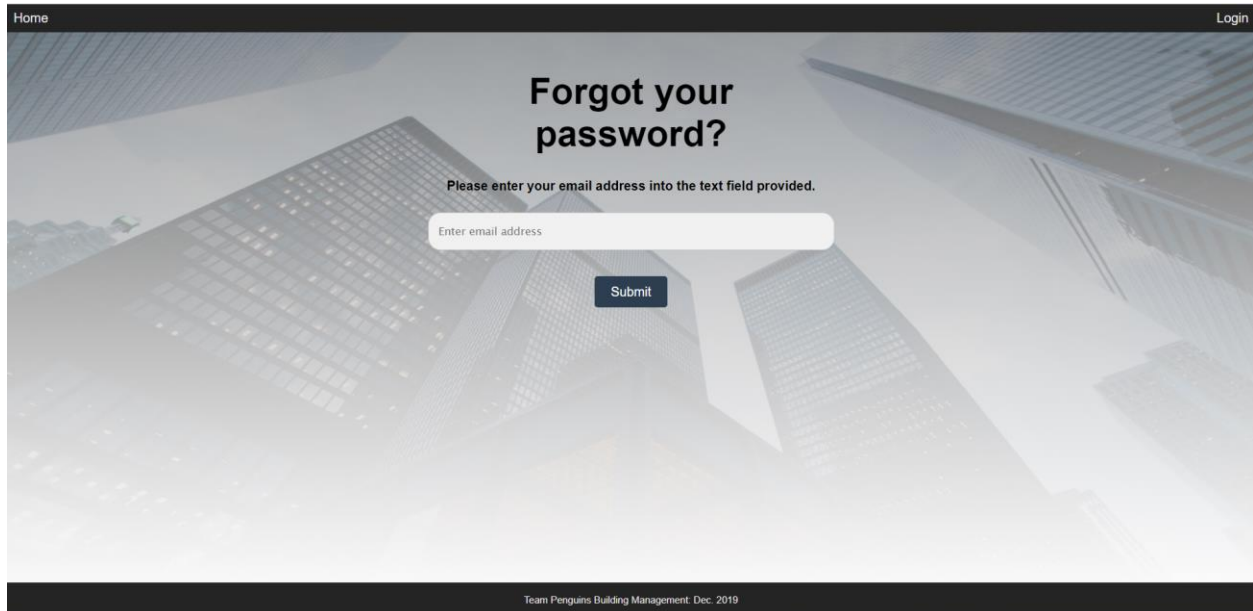
Groups Page

The groups page allows admins to place existing users into different groups for a facility. That way, facility owners are able to know who is allowed to make reservations in their facilities.

The screenshot shows a web application interface for managing user groups. At the top, a navigation bar includes links for Home, Profile, Reservations, Management, and Maintenance Submissions. The user is logged in as 'Alex' and can click 'Log Out'. The main heading is 'Update Groups for Facility'. On the left, there are two dropdown menus: 'Select a group:' with options 'admin' and 'default', and 'Users:' with the option 'Alex Lam (test@example.com)'. Below these are two empty input fields. At the bottom left, there are four buttons: 'Add Group', 'Add User', 'Remove Group', and 'Remove User'. The background features a faint image of skyscrapers. The footer text reads 'Team Penguins Building Management, Dec. 2019'.

Forgot Password

The forgot password page is to allow users who have forgotten their passwords to recover their accounts. They will be first redirected to a page that takes in their email address.



Home Login

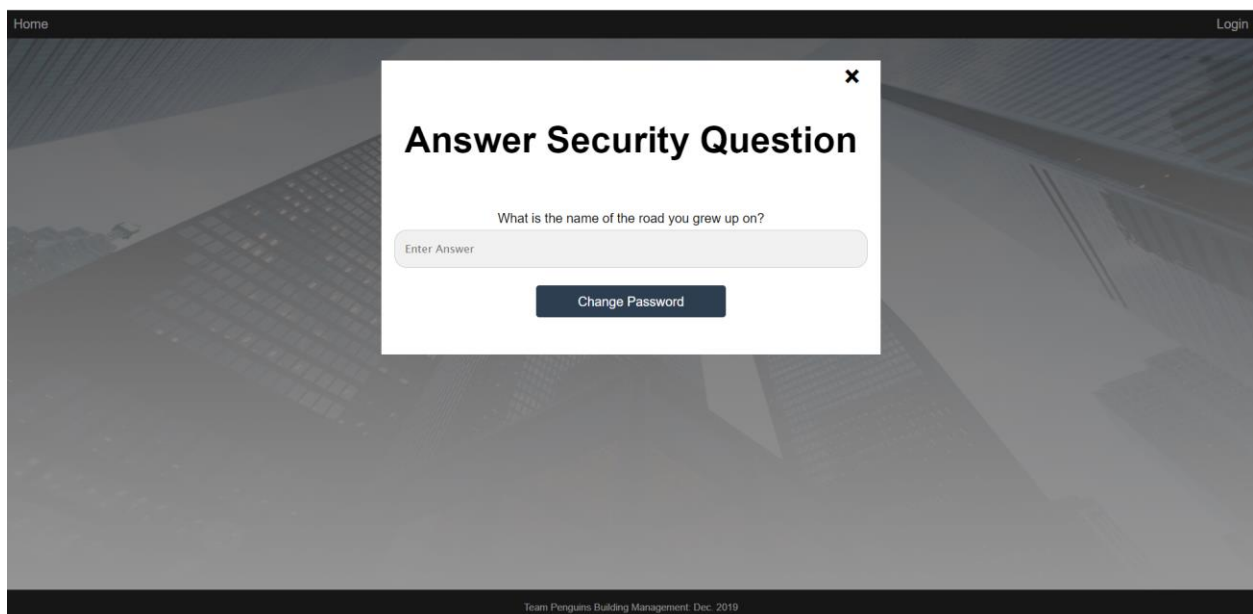
Forgot your password?

Please enter your email address into the text field provided.

Submit

Team Penguins Building Management Dec. 2019

When they enter a valid address into the text field, they will be given a prompt to enter their security question that the user created at the creation of their account.



Home Login

Answer Security Question

What is the name of the road you grew up on?

Change Password

Team Penguins Building Management Dec. 2019

When the user answers the security question correctly, they will be given a page

to change their password.

The screenshot shows a web interface for changing a password. At the top, there is a dark navigation bar with 'Home' on the left and 'Login' on the right. The main heading is 'Change password'. Below it, there are two input fields: 'Create new password' with the placeholder 'Enter new password', and 'Confirm password' with the placeholder 'Re-enter password'. A dark blue button labeled 'Change Password' is positioned below the second field. The background is a faded image of a city skyline. At the bottom, a dark footer bar contains the text 'Team Penguins Building Management. Dec. 2019'.

Once the user enters a new password, then the account will be ready to use again.

The screenshot shows a web interface for searching for a facility. At the top, there is a dark navigation bar with links 'Home', 'Profile', 'Reservations', 'Management', and 'Maintenance Submissions' on the left, 'Welcome Alex!' in the center, and 'Log Out' on the right. The main heading is 'Search for a Facility'. Below it, there are two search options. The first is 'Search for a Facility by name' with a text input field containing the placeholder 'Search for a Facility' and a dark blue 'Submit' button. The second is 'Zip Code' with a text input field containing the placeholder 'Zip Code' and a dark blue 'Submit' button. Between these two sections is the word 'OR'. Below the 'OR' is the option 'Search for a Facility by Access Code' with a text input field containing the placeholder '#####' and a dark blue 'Submit' button. The background is a faded image of a city skyline. At the bottom, a dark footer bar contains the text 'Team Penguins Building Management. Dec. 2019'.

User Maintenance Request Overview Page

The user maintenance request page allows users to see all of the requests they have made for rooms. Users will be able to see when they submitted the request, the facility and building the room belongs to, the room number, and the description of what they have submitted. The user will also be able to see the status of their maintenance request.

HomeProfileReservationsManagementMaintenance Submissions

Welcome Alex!Log Out

User's Maintenance Requests Overview

View the maintenance requests for a facility below. If the request has been fulfilled you can update the request and click the submit button.

Date Submitted	Facility and Building Name	Room Number	Description	Status
Sun Dec 15 2019 15:22:25 GMT-0500 (Eastern Standard Time)	FacilityTest: BuildingTest	456	The TV doesn't work.	Submitted

Team Penguins Building Management. Dec. 2019

Web Page Navigation

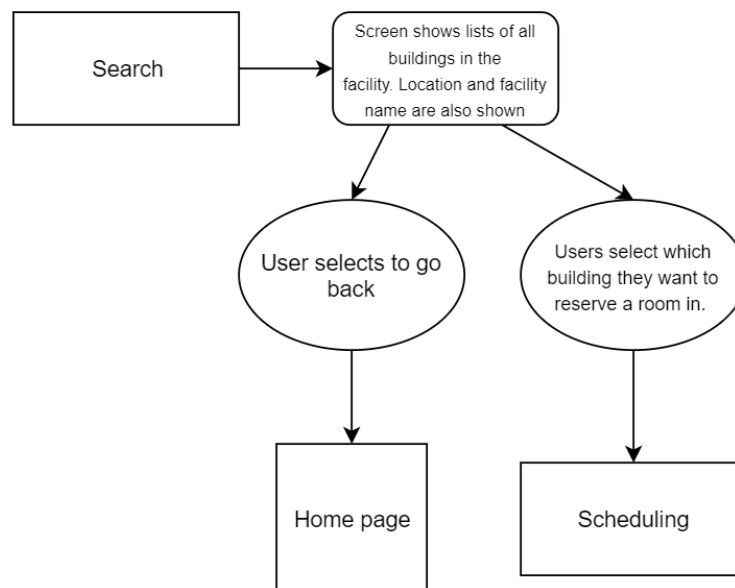
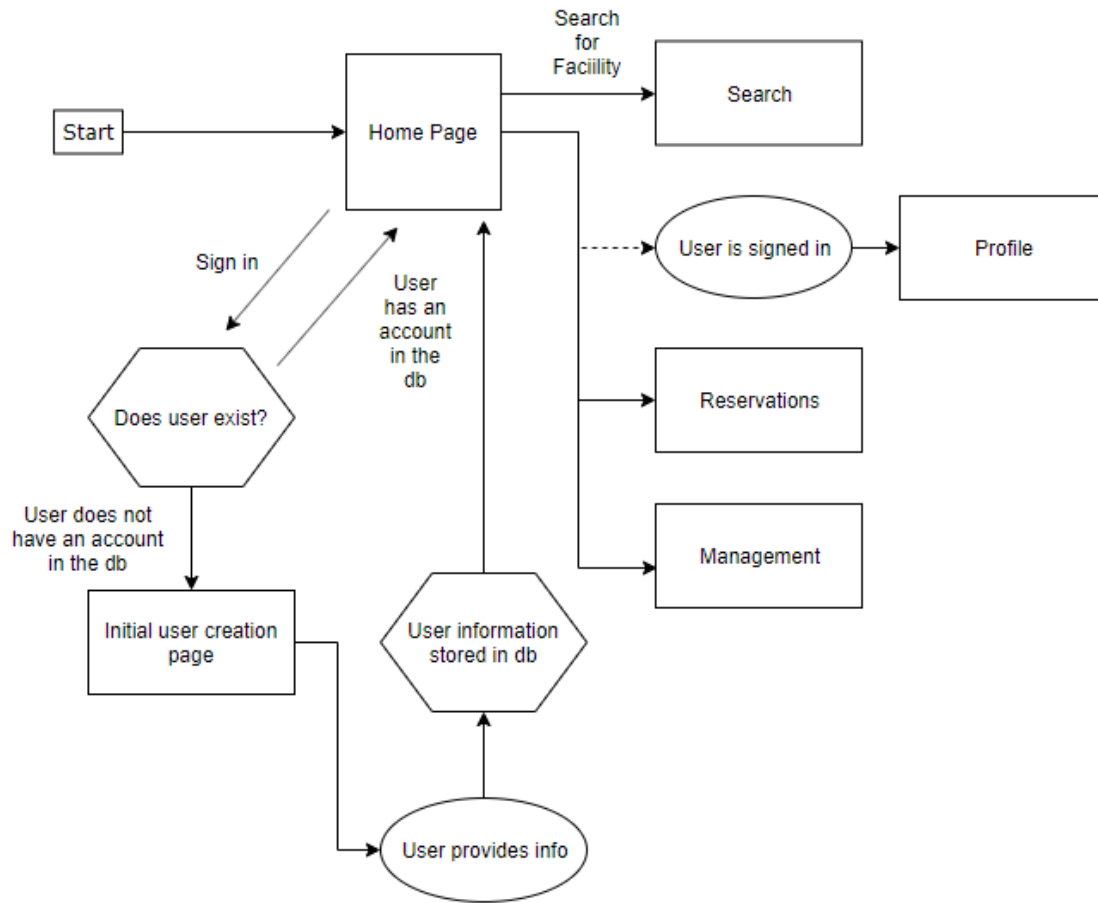
When the user first loads the website they will be presented with the home screen. The user will be able to enter a facility name and zip code to search for a facility. Once the user searches for a facility, results with the facility's name and location will appear. Users will be able to click on a facility's name to be redirected to the *Facility Page* for more information for the selected facility. On the *Facility Page*, users are presented a list of buildings within the facility. After selecting a building they will be sent to the *Buildings Page*.

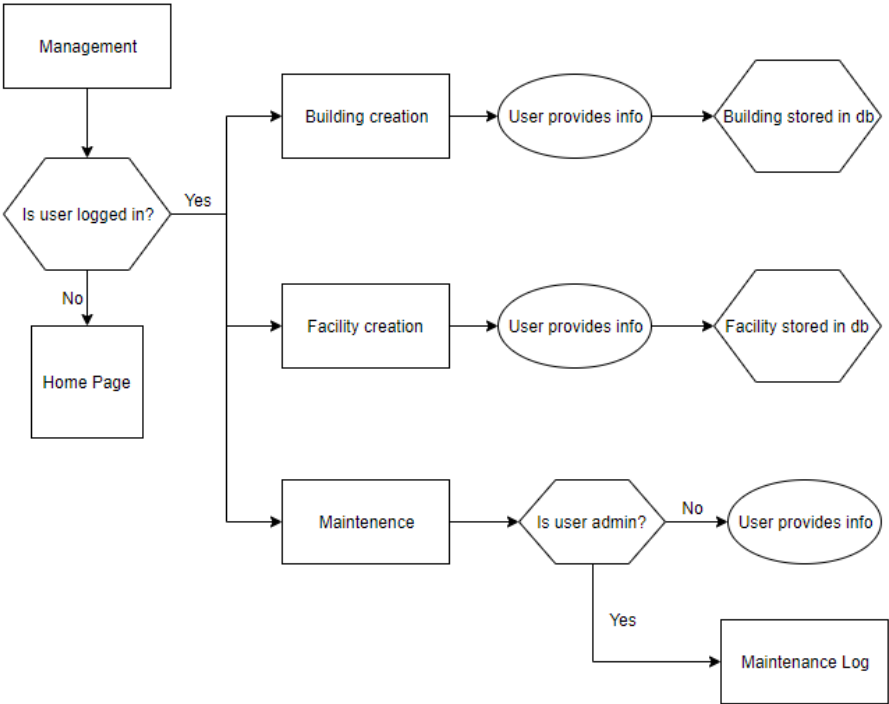
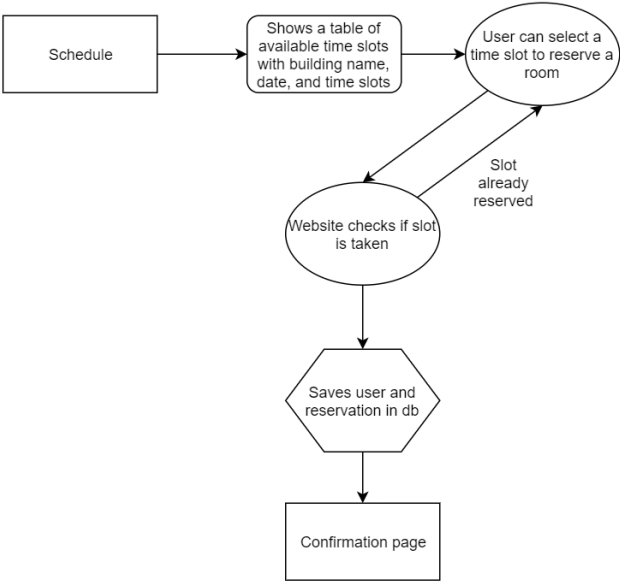
The user will schedule a room within the facility from the *Buildings Page*. They will select a room, date and time for when they want to reserve the room and then submit their request. If the user has not logged in, they will not be able to reserve a room until they do log in. If successful then they will be sent to the *Room Confirmation Page* which confirms and provides information on their request.

All users will have to login to access all available features provided. When prompted to login, if the user does not already have an account, or use an email that is already associated with an account, they can create one using the *User Registration Page*. When logged in, the user will be able to access the *Management Page*, through the management tab to manage facilities they are an admin of. The *Management Page* provides the user with the ability to create, or delete, a facility, buildings and submit and view all current maintenance request. Users can then go to the *User Maintenance Request Overview Page* to see the status of all maintenance requests they have made.

Other options available from the Navigation Bar include *Profile*, *Home* and *Reservations*. If the user clicks on any of these options they will be prompted to login if they have not already done so. The *Profile Page* lets users edit their information, including but not limited to their: name, email, password and phone number. The *Home Page*, as described above, list buildings that are within a facility. Lastly, the *Reservations Page* shows users all of their current, and past, reservations.

Flow Diagram





Technology Stack

The application uses the Javascript framework Vue.js. We are also using Axios for consuming API endpoints. We are using FullCalendar as our calendar component for the scheduling pages.

The app uses the Python microframework Flask for the backend service. It handles the RESTful API and serving the web pages. MongoDB was chosen for the database for its NoSQL structure. It connects to Flask using PyMongo.

Our website, backend, and database are hosted on an AWS EC2 instance, which is also running the Apache HTTP server.

Backend Information

The backend is made in Flask for Python, and it will connect with the MongoDB database, providing data for the frontend web pages. The database holds all of the information about the facilities, buildings, rooms, maintenance requests, and users.

All of the business logic code for scheduling will be performed on the backend through the use of API endpoints. This includes checking whether a user has the required permissions to create a new reservation, and ensuring any one room does not have two or more reservations for the same time.

Authentication and Security

To make any reservations using our application, users must create an account first. Accounts will be created by providing an email address and creating a password. Users will also provide their first and last name. Passwords will have a salt value added to them, then hashed. We will not store plain text passwords. All database injection attacks are considered and accounted for. To log in, the user will enter the email address they registered with and their password.

Facilities registered as private will not be visible upon search to a user who is not registered with that facility. A user may register with a private facility by providing an access code that will be given to the facility's owner.

Input and Output

Input: User input will be a mixture of buttons and text fields. In most cases the text fields will require you to enter information, which will be used to search, filter, edit and add to the database. These include, but are not limited to, facilities, buildings, groups, rooms and maintenance requests. Buttons will be primarily utilized to navigate through web pages, they will also be used to submit data in the text fields, select/set fields, and to select objects from our lists. For our program we will utilize a calendar to display the reservations a room currently has, in this instance we would require the user click a spot on the calendar in order to select it, bring up more information, or submit a reservation for that time slot. Our web pages will almost always have a mixture of these two input types.

Output: The output to the user is dependent on which page the user is currently on. In most cases you will search one of the lists (like facilities) and a list of facilities would be returned that match any filters input by the user. All the information may not be shown in the list, however if you were to click on it, the information for that specific facility will be shown in more detail. This extends to many of our database objects. When a user wants to create or edit an instance of an object, the object in question will be returned and output. Searching list will return the list of said objects, while specifying the object ID will return information of the object specified.

Database Schema

The database will utilize MongoDB's structure in expanding columns. If a user would like to add/remove rooms, buildings, and facilities, the database will maintain its integrity without extraneous maintenance. Since all the scheduling is self-contained to each different facility, they will each have their own reservation system. A facility will have many features and it will be the host of the scheduling process.

Each facility will have a name, address, and the collection of groups created in the facility. The data structure also has an access code which will be used if the facility is marked as private. The facility also holds all the maintenance reports. Maintenance reports will have the building the report belongs to, where the room is located, and what amenity needs to be addressed.

Facility Document

```

_id: ObjectId("5dd81b02c3fe0ffb0e9e2950")
name: "Rowan University"
private: false
access_code: "950phcQLWI"
address: Object
  address_L1: "123 Mullica Hill Road"
  address_L2: null
  city: "Glassboro"
  state: "NJ"
  zip: "08028"
  country: "USA"
phone: "123-555-7887"
description: "another description"
attributes: Array
  0: "whiteboard"
  1: "TV"
  2: "computers"
  3: "projector"
maintenance: Array
  0: Object
    _id: ObjectId("5df402aa62d9a0b16ba6352d")
    buildingName: "Robinson"
    roomID: ObjectId("5ddd997363777aaba9717840")
    roomNum: "101"
    description: "bad"
    userID: ObjectId("5dd99013e0ca294864d98b8b")
    date: 2019-12-13T21:29:14.387+00:00
    status: "Submitted"
groups: Array
  0: Object
    _id: ObjectId("5dd81b02c3fe0ffb0e9e294f")
    name: "admin"
  1: Object
    _id: ObjectId("5dea9bf20516b0c9e4d25a6d")
    name: "new group"
  2: Object
    _id: ObjectId("5deaa05177ae9b4f5b0a5f64")
    name: "another new group"

```

Buildings are held in a separate collection. Every building is an individual document, linked back to the facility it belongs to by storing its facility ID. The buildings hold the name of it, its address, and a description of the building.

Building Document

```

    _id: ObjectId("5dd9993738535a92b3b6f12d")
    name: "Test1"
  > address: Object
    phone: "1231231234"
    description: "test"
    facilityID: ObjectId("5dd998f138535a92b3b6f12c")

```

Rooms are held in a separate collection. Every room is an individual document, linked back to the building it belongs to by storing its building ID. Rooms may have different attributes listed in them, such as a whiteboard or TV. They will also have a room number to associate where the room is located within the building.

Reservations for each room are also stored in the Room document. This will hold all the reservations that are created for that room with its scheduling information. The user ID of whoever made the reservation is also stored.

Room Document

```

    _id: ObjectId("5de5b6c0981d115774185a09")
  < attributes: Array
    0: "whiteboard"
    1: "tv"
    buildingID: ObjectId("5de5b69c981d115774185a08")
    capacity: "35"
  < groupID: Array
    0: "5de5b66c981d115774185a06"
    name: "Classroom"
    number: "300"
  < reservations: Array
    < 0: Object
      _id: ObjectId("5de5badbbe2411aad604c3a1")
      start_time: 2019-12-03T10:00:00.000+00:00
      end_time: 2019-12-03T12:00:00.000+00:00
      userID: ObjectId("5dd99013e0ca294864d98b8b")

```

Users will have a username, password, and email associated with them at the creation of the account. They will also have their first and last name associated with their account to identify the identity of the account holder. Each user will also have an array of groups that the user belongs in. This is to decide which users have access to certain buildings. For security, passwords and security answer questions are hashed.

User Document

```
_id: ObjectId("5de5d0225a799d7eb479a8a5")
password: Binary('JDJiJDEyJGlyR0JCYUJ2blJ1RVNxdjVWNGJFYWUuZnFSMlFOWjB5U29TTnEuLnp1ckExeTB2a1kwVzFX')
email: "monicasemail@gmail.com"
first_name: "Monica"
last_name: "Mahon"
question: "What is your favorite book?"
answer: Binary('JDJiJDEyJE5yZkFSLlFwSWN0QTY4a0ozM1JhbHVUCe1UVGw3STdMQ1BPS0ZKZTgxVEZW3c5Z0d4SkNP')
groupID: Array
  0: ObjectId("5df108ef0de27b3507c316ea")
  1: ObjectId("5df108ef0de27b3507c316eb")
  2: ObjectId("5df117c3d6c52d45e5bc1b26")
```

Restful Endpoints

The endpoints are labeled in the form of REQUEST TYPE /api/url/. Inputs to an endpoint are denoted by brackets. Additional parameters/information needed will be passed using JSON. Assume that JSON with the appropriate data is passed for all POST and PUT requests, and if any errors occur a corresponding error message is returned.

Facility

GET /api/facilities Returns facilities that match with the passed data; ZIP and name, or access code for a facility

GET /api/facilities/<facility_ID> Returns the facility that corresponds to the facility_ID

POST /api/facilities Create a new facility and return the facility_ID

PUT /api/facilities/<facility_ID> Updates the facility of the given facility_ID

DELETE /api/facilities/<facility_ID> Deletes the facility that corresponds to the facility_ID

Building

GET /api/facilities/<facility_ID>/buildings Returns the buildings of a given facility_ID

GET /api/facilities/<facility_ID>/buildings/<building_ID> Return the building of the corresponding building_ID

POST /api/facilities/<facility_ID>/buildings Creates a new building and returns the building_ID

PUT /api/facilities/<facility_ID>/buildings/<building_ID> Updates the facility of the given building_ID

DELETE /api/facilities/<facility_ID>/buildings/<building_ID> Deletes the building that corresponds to the building_ID

Room

GET /api/facilities/<facility_ID>/buildings/<building_ID>/rooms Returns the rooms of a building given the facility_ID and building_ID

GET /api/facilities/<facility_ID>/buildings/<building_ID>/rooms/<room_ID> Returns a room that corresponds with the given facility_ID, building_ID and room_ID

POST /api/facilities/<facility_ID>/buildings/<building_ID>/rooms Creates a new room and returns the room_ID

PUT /api/facilities/<facility_ID>/buildings/<building_ID>/rooms/<room_ID> Updates the room that corresponds with the given facility_ID, building_ID and room_ID

DELETE /api/facilities/<facility_ID>/buildings/<building_ID>/rooms/<room_ID> Deletes the room that corresponds with the given facility_ID, building_ID and room_ID

Reservation

GET /api/facilities/<facility_ID>/buildings/<building_ID>/rooms/<room_ID>/reservations Return all reservations in the room that corresponds with the given facility_ID, building_ID and room_ID

GET /api/facilities/<facility_ID>/buildings/<building_ID>/rooms/<room_ID>/reservations/<reservation_ID> Return the reservation that corresponds with the given facility_ID, building_ID, room_ID, and reservation_ID

POST /api/facilities/<facility_ID>/buildings/<building_ID>/rooms/<room_ID>/reservations Creates a new reservation and returns the reservation_ID

PUT /api/facilities/<facility_ID>/buildings/<building_ID>/rooms/<room_ID>/reservations/<reservation_ID> Update the reservation that corresponds with the given facility_ID, building_ID, room_ID, and reservation_ID

DELETE /api/facilities/<facility_ID>/buildings/<building_ID>/rooms/<room_ID>/reservations/<reservation_ID> Delete the reservation that corresponds with the given facility_ID, building_ID, room_ID, and reservation_ID

Users

GET /api/users Returns the user that is currently logged in

POST /api/users Create a new user

PUT /api/users Update the user that is currently logged in

DELETE /account Delete the user that is currently logged in

Maintenance

GET /api/facilities/<facility_ID>/maintenance/ Return all maintenance requests

GET /api/facilities/<facility_ID>/maintenance/room/<room_ID>

GET /api/facilities/<facility_ID>/maintenance/<maintenance_ID> Return request given a maintenance_id

POST /api/facilities/<facility_ID>/maintenance/ Create a new maintenance request

PUT /api/facilities/<facility_ID>/maintenance/<maintenance_ID> Update a maintenance request given the facility_ID and maintenance_ID

DELETE /api/facilities/<facility_ID>/maintenance/<maintenance_ID> Delete a maintenance request given the facility_ID and maintenance_ID

Group

GET /api/facilities/<facility_ID>/groups Return the groups of a facility given the facility_ID

GET /api/facilities/<facility_ID>/groups/<group_ID> Return the group that corresponds with the given facility_ID and group_ID

POST /api/facilities/<facility_ID>/groups Create a new group

PUT /api/facilities/<facility_ID>/groups/<group_ID> Update a group given the facility_ID and group_ID

DELETE /api/facilities/<facility_ID>/groups/<group_ID> Delete a group given the facility_ID and the group_ID

UserReservations

GET /api/user/reservations Return the reservations for the user thats logged in

UserGroup

GET /api/facilities/<facility_ID>/users/groups/<group_id> Returns the users that correspond with the given facility_ID and group_ID

POST /api/facilities/<facility_ID>/users/groups/<group_id> Add a user to the group

DELETE /api/facilities/<facility_id>/users/<user_id>/groups/<group_ID> Delete the user with the given user_ID from the group with the corresponding facility_ID and group_ID

UserFacility

GET /api/users/facilities Returns the facilities the logged in user is an admin of

UserMaintenance

GET /api/maintenance/users Returns the maintenance requests the logged in user has created