# **GYM PASS**

# **ACCEPTANCE CRITERAIA**

**Summary:** As a user I would like to be able to check the number of people currently at a gym at any given time. I would like the gym doors to open when I log into the web app and command it to open, and the doors to close and lock when I log out, and command it to close.

**Main Feature Highlights:**

* Tracks number of users at a given gym at any given time.
* Open and close door by logging into the web app and flicking the lock switch on app (Gym owners can register the user).
* 3 or 4 User Interface pages include:
  + Login/Register
  + View Number of people in the gym in the main page, along with open/close doors button on the same page (Landing page if logged in, otherwise re-direct to log-in).
  + Dashboard to view/select gyms
* Open door will just display doors opened for 5 seconds when open door is triggered, then return to its normal state of showing doors closed for this app demonstration (Will use JavaScript so the user can watch timer before the door will close, using a real-time progress bar).
* Users registered by gym owners saved on Firebase

*Optional Features:*

* Gym Admin Page for gym owner to grant permission for registered members to open/close doors with app.
* If the user is standing in the way of the door, the 5 seconds countdown will reset and not start going down until user moves (use button standing in way to enable test).
* Use Deep Lens camera to detect when user is outside door (use web camera for demonstration).
* Mobile App version, which will also have a fingerprint authentication option
* Open door only when within 10m (location data needs to be enabled for this – this way it does not allows need to be turned on).
* Ask Alexa to open/close door: “Alexa, open the door”.
* Currently only built for one gym, will consider expanding to registered gyms.
* Best hours based on

# **MOCK-UP**

LOGIN PAGE

Menu Link

LANDING PAGE

Estimated occupancy time icons

IMAGE OF A GYM GOES HERE

GYMPASS

Fingerprint Icon here  
if optional  
proceeds

Progress bar for timer if optional proceeds

Open  
/close door

# of ppl at selected gym

LOGIN

Password

Email

WELCOME

SIGN IN

Forgot password?  
Not registered? Click here to register

A close up of a logo

Description automatically generated  
GYMPASS

50

DOOR : OPENED/CLOSED

USER REGISTRATION PAGE

SELECT GYM PAGE

JOIN NOW!

CURRENT GYM

Gym Name  
Current Number of Clients:   
Average Estimated Occupancy Times:   
Weights-room: Stretching room:

Address: ………………………………………  
 ………………………………………  
 ………………………………………  
Mobile: ………………………………………  
Email: ………………………………………

First Name

Email

Change Gyms

Select gym (select a gym from dropdown)

REGISTER

Register for another gym

Select gym (select a gym from dropdown)

Modal pops up with terms and conditions, disclaimer to check accept. And email will be sent to the gym owner.

OPTIONAL GYM STAFF ADMIN PAGE  
(FOR ACCEPTING USER REQUESTS)

Hi First Name!

NAVIGATION

IMAGE OF A GYM GOES HERE

GYM NAME

UNLOCK MY GYM

SELECT GYM

DASHBOARD

some logout icon

**Notes about mock-up:**

* Pop-up modal questionnaire will be popped-up after unlocking the door.
* There will be a dark background while you are out of a gym, when you are inside it will light up.
* The open/close door lock will only have a text box that shows door open or doors closed below in it initially, will later look-into making shape dynamically interchange between an opened lock icon and a closed lock icon as per door status.
* If we implement facial recognition later, we will show door open or door closed based on result of facial recognition scan results of our web camera. Clicking the lock button will activate the facial recognition scanner. (This way is even better so we can prevent misuse of open-door functionality, being < 10m also being a consideration to enable open door.)
* Opacity on current page goes down when nav button is pressed.
* Change Gyms can only select a gym you are registered to.
* For our presentation only change gym drop down list will have drop down items

(we will make 3 registered users, one registered to two sample gyms who will have both gyms available on the drop-down list. One user with only one gym registered, who can see only that one, and one with none, who will have no gyms to select).

* Page refreshes every 10 mins to refresh data
* Website is being developed to suit iPhone X as it is currently the most used phone
* - Averages for gym numbers are automatically calculated based on monthly data.
* How long will your gym session last?
* 20mins 40mins 1h 1h30mins 2h More than 2h
* What area will you be using? (This is multioption)
* Dumbbell area Weights Cardio area (bicycle, treadmill) Mat area
* Optional reset everything at midnight every day, look into automated postman or alternative automated requests being sent to dedicated controller so we can set everything to false, when this request executes.

# **ALGORITHM**

User tries to check the current gym for it’s current number of visitors, opens the home page.  
IF user is not logged in   
 THEN re-direct to login screen.

IF user is logged in  
 THEN send to landing page  
IF User Clicks Enter GYM  
 THEN IsOpenDoorRequested is true, DoorOpened is true  
 DoorCloseTimer starts  
 IsOpenDoorRequested turns false  
 WHILE DoorCloseTimer is not 0, the DoorCloseTimer ticks down  
 IF IsOpenDoorRequested true while counting down  
 DoorCloseTimer resets  
 E

# **FLOWCHART**

TO DO

# **SOFTWARE**

* AWS AppSync
* AWS Dynamo DB
* AWS swf simple workflows
* AWS step function

**Currently Using**

* AWS Bucket

# **TESTS**

* Verify two users can affect the lock timer reset (Optional)
* Use three browsers to conduct tests
* Test if user has been authenticated and authorized by input box with default true value, unhide the input box to show both access granted and access denied.
* Based on above test, but user real facial recognition with webcam to determine if authentication succeeded.
* Get three users to enter the gym, show database reflect number of people in the gym to reach 3, when one person clicks the button to leave the gym
* Check that we only show access scanning and access granted/denied when entering and not when leaving
* Access denied, will show for several seconds after denied, once we refresh it will remain, until time has passed.
* Test if we can log another workout after logging one today, show we cannot, however manually update db again with yesterdays date, show we can now do it.
* Test the estimated number in the gym at any given time, test that if we select a check training time past the current users estimated training time. Check if we log into another user and use a estimated time ahead of all users training logged estimated training time (1hr) that we find 0 estimated in 2hours time, so long you are in the gym even if all users are in gym, you will
* Gym Occupancy Panel Tests:
  + Click on total in gym to swap between, estimated total and actual total.
* Test location check, use google map to identify something within 10m of current location, change source code to reflect >20m, and see it get declined.
* Click on map icon to reveal real map, which can show real map location
* Use Edit Image Page to edit the image to someone else, show authentication failed, then compare against me to show validation based on confidence level >90% OR just take a image or point camera for 3 seconds somewhere else
* Take picture when I’m not looking at the camera properly and when I am
* Show that an image is uploaded to an S3 bucket when I enter, and removed when I leave

# **FEATURES DISCOVERY**

Troydon:

* Crowd sensing to track real time data of users currently at the gym
* - Everything else, as the remaining users have left

Benjamin Sweetnam:

* DeepLens Scan camera scan to gain entry

Marcos Esgueva:

* Refined idea of logging workout logs

# **TECHNICAL SPECIFICATIONS**

1. When user first logs in, website attempts to send him to the home page, this is the default routes Index page. Default route is specified in the Startup.cs class where the services and middleware are handled.

// services

public void ConfigureServices(IServiceCollection services)

{

// tells services to add controllers

services.AddControllersWithViews();

// adds the databases connection string here

services.AddDbContext<FacilityContext>(options =>

options.UseSqlServer(Configuration.GetConnectionString("FacilityContext")));

}  
 // This method is called at runtime, this configs the HTTP pipeline.  
 public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

endpoints.MapRazorPages();

});

}

1. For the logins to work, login view and register view were scaffolded. In registration I had added two additional customised properties, as below.

// by using input models to save user data, instead of the properties from the ApplicationUser class where the actual properties lie, we can safely update user details with added security, as in the event of an attack only these will change.

[BindProperty]

public InputModel Input { get; set; }

public class InputModel

{

[Required]

[DataType(DataType.Text)]

[Display(Name = "First Name")]

public string FirstName { get; set; }

[Required]

[DataType(DataType.Text)]

[Display(Name = "Select Default Gym")]

public int SelectDefaultGym { get; set; }

}

// This is where the HTTP Post service sends the request to transfer data // about creating users

public async Task<IActionResult> OnPostAsync(string returnUrl = null)

{

var user = new ApplicationUser {

UserName = userName,

Email = Input.Email,

FirstName = Input.FirstName,

// default value for default gym for now, later I will implement default gym selection during sign up and at dashboard.

DefaultGym = 1

};

1. In the login post action, I added a method that allows you to sign in with username:

// allow you to sign in with username

var userName = Input.Email;

if (IsValidEmail(Input.Email))

{

var user = await \_userManager.FindByEmailAsync(Input.Email);

if (user != null)

{

userName = user.UserName;

}

}

1. Every page is designed to use one universal Layout page which assists with avoiding to have to rewrite code for default elements that should appear on all pages, such as the background colour, and the back navigation. In \_Layout.cshtml

<!DOCTYPE html>

<html lang="en">

<head>

Metas, script and style imports here will appear on every page

</head>

<body>

@\*<a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index">GYMPass</a>\*@

<main role="main" class="pb-3">

<header>

// Main Navigation system goes here 🡪

@ViewData["Title"] // this automatically invokes the view data = title of page for each page

</header>

// this calls every other pages html and renders it

@RenderBody()

</main>

<footer class="border-top footer text-muted">

// footer

</footer>

// any scripts that should be called after load is imported here

</body>

</html>

1. After we login we are sent to the home page with our default gym. The code explains the current functions