

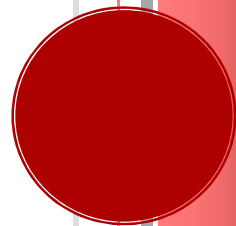
PROJECT 3

<https://wheredoyoushare.herokuapp.com/>

This is a website which allows members a social platform to share their interest in an 'item'. The website was built using the Django framework which encompasses HTML, CSS, JavaScript, jQuery and Python. Data is stored in a ClearDB SQL database. It is hosted on heroku.com and sizes appropriately on phone, tablet or PC.

Maria Hynes

December 2016



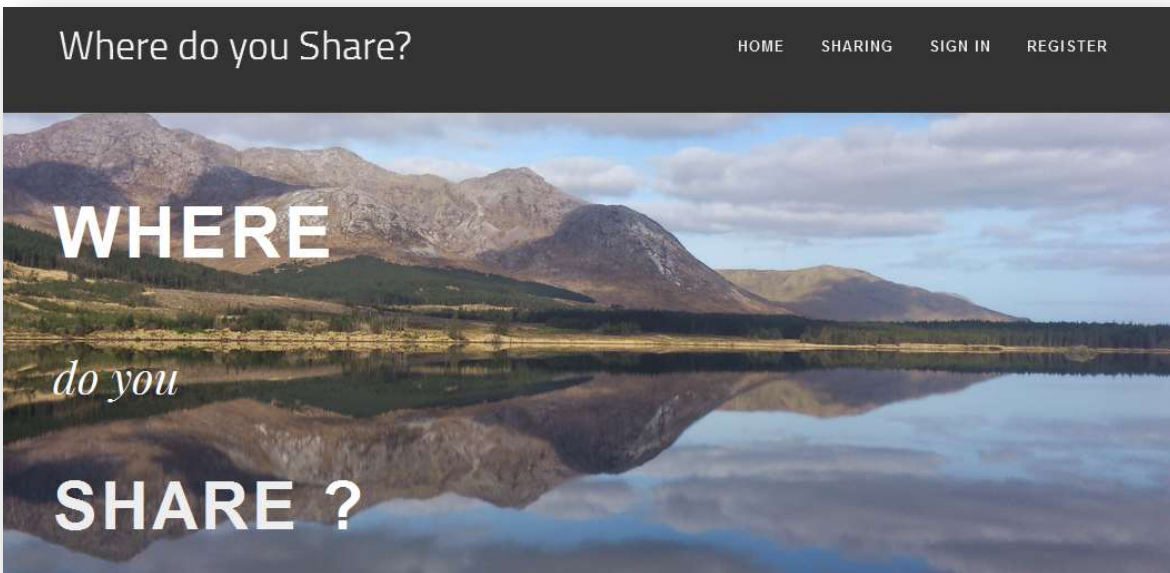
PROJECT 3

Contents

The Brief.....	2
Meeting the Brief.....	2
What is an Asset?.....	2
Who is the Requestor?.....	3
Requestor Benefits	3
How do I link to an Asset?	4
How does the Booking work?	4
Sample Data	5
Weekly Ownership	5
Monthly Ownership	5
Single Ownership	5
Site Functionality per User Type	6
Unregistered Site Visitors	6
Registered Site Visitors.....	7
Models.....	8
Planning	9
Making a Booking	11
The request logic	13
The approval logic	13
The confirmation logic.....	14
The ‘consecutive dates’ issue.....	14
The ‘multiple-owner’ issue	14
Stripe Payment Details.....	18
Testing	18
User Testing	18
Code Testing.....	21
Blackbox Testing.....	21
Next Steps.....	22

THE BRIEF

Arising from a need within my own family – that of finding a way to manage the usage of a shared summer home – my brief was to use *my* functional requirements as starting point to build a website/app that could be used by *other groups of people*, anywhere, who also shared a common ‘resource’. I wanted to create a place that could be used for managing the requests and bookings of that resource and facilitate the social aspect, enabling the users to give opinions and reviews of all things related to it. While most of this could be achieved using an existing app like Facebook, the uniqueness of this site is that it combines the **social aspect** with the **booking** side of things, the fact that not just anyone can join, and the flexibility around what exactly the shared asset can be, i.e. ‘where’ or even ‘what’ ...do you share?!?



MEETING THE BRIEF

Throughout this document I will refer to the ‘common resource’ as an ‘Asset’, an owner of an Asset as an ‘Owner’ and a user of an Asset as a ‘Requestor’ or ‘Member’.

What is an Asset?

An Asset has two features:

- It is something that belongs to a person or persons
- It is something that can be shared

While it is possible on this site to register an Asset which belongs to only one Owner, the real magic of the site becomes apparent when a multi-owned Asset is registered. A ‘multi-owned’ asset is something that is owned by two or more people. As soon as these owners want to share/lend this item outside of the owner circle, then this arrangement becomes more difficult to manage. For example, if someone wanted to borrow this Asset, in a real-world scenario what would they need to do? Would they need to seek permission from all Owners or just a couple of them? What if one Owner says ‘yes’ to a request, without realising that another Owner has already granted permission to someone else for the same dates? Do all Owners have to inform the other Owners when the Asset is booked and unavailable? How would they do that and how long would that take? What if you can’t get in touch with all the Owners? Does one Owner have more authority than another Owner to approve a request?

This site manages the complicated relationship between Assets and their Owners and Requestors. To use this site, Asset Owners need to consider and agree to the concept of a 'virtual' ownership time-slot. This virtual slot is a period of time when you can consider that the Asset is completely under your ownership. After your virtual slot, the ownership moves to the next person for the same period of time and so on and so on until the rotation comes back around. If a Requestor/Member would like to book the use of the Asset, they will only need to seek permission from the person who is the Owner of the time-slot on the requested dates.

In order for a multi-owned Asset to be suitable for use on this site, with *minimal set-up overhead*, the Owners should agree on the following points in relation to their Asset:

- Ownership constantly rotates between Owners in a set order
- The ownership slots must be divided into multiples of months or weeks
- A sharing start-date must be provided

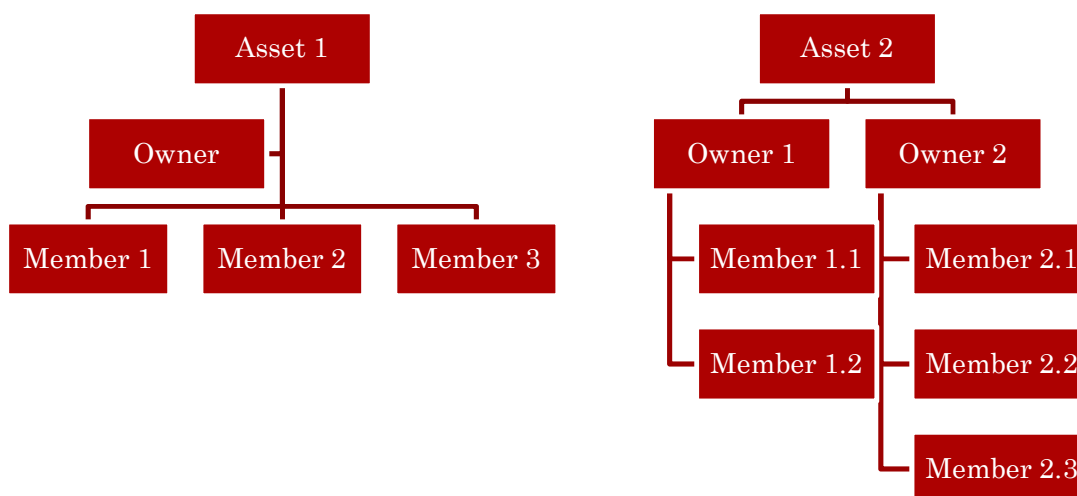
This is the least amount of information necessary in order to set up an efficient booking engine for the Asset. With only these pieces of information, an Owner does not have to upload and maintain large tables of dates and ownership data, yet the site can expertly organise and manage all booking requests.

Who is the Requestor?

A Requestor on this site is someone who has been invited to share an Asset. They have received an invitation from an Asset Owner who is basically saying "hey, I've got this Asset that I'm inviting you to share with me". As soon as the person responds to the invitation and agrees to link to the Asset, they become a Requestor and can use the site to request and book the dates they would like to use the Asset. Every Owner has the ability to invite multiple Requestors and once the Requestor responds, then they are linked to the Asset *through* that Owner.

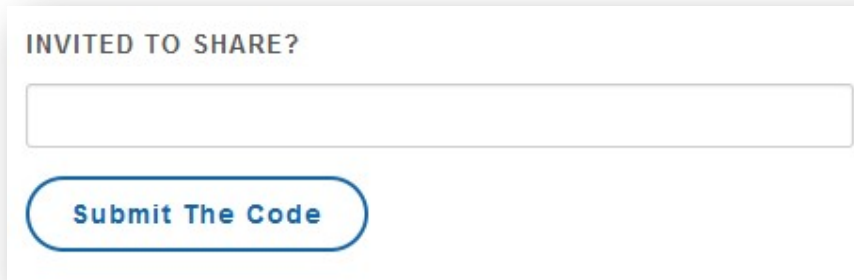
Requestor Benefits

Being linked to an Owner means that Requestors can consider the Owner's time-slot as being their time-slot too. While they still need to request dates in the same way as everyone else, they will get priority if they are booking dates in their own time-slot. They will also get notice if another Requestor wishes to book dates in 'their' time-slot. Although it is only the Owner who has the ability to approve or deny these requests, all the Requestors who are linked to the Owner will know that a request for dates in 'their' time-slot is pending. When this happens, the Requestor can take advantage of this early-warning system and contact the Owner to deny the request or some of the dates in the request, so that they can book instead. This visibility on the pending requests saves the Owner the time and hassle it can take to inform other Requestors about the request. It is convenient for the Owner to approve a request like this as they know that all their Requestors will have seen it and if they haven't disputed it, then it is safe to approve it.



How do I link to an Asset?

There are two ways to link to an Asset. You can enter an Invite Code which has been sent to you, in which case you become a Requestor of the Asset.



INVITED TO SHARE?

Submit The Code

Or you can register a new Asset on the site and in this case, you become the Owner of the Asset and you can send out invitations.

There is no limit to the number of Assets you can share. You can use the site as an Owner of one Asset and as Requestor of another.

How does the Booking work?

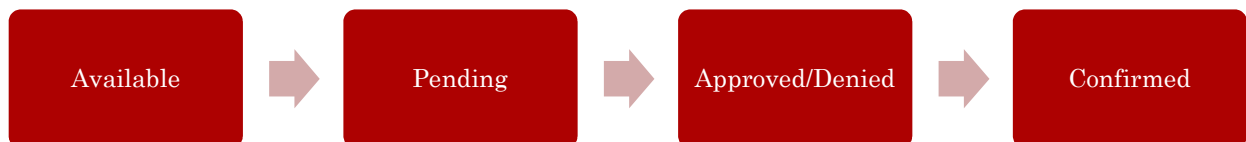
When a Requestor selects the start and end dates they would like. The code checks two things:

- Who is the Owner during the time range requested
- Whether the dates have already been requested by someone else

If all the requested dates are available, the user can click to [Submit the Request]. This will create a Booking reference and will mark each of the requested dates as Pending. If the date range requested spans more than one time-slot then more than one Owner will be informed that they have pending bookings to review. Each Owner is only responsible for approving the dates that fall into their own time-slot. It is only after each individual date has been either approved or denied that the original Requestor can confirm the booking. During this whole process, the dates are unavailable for anyone else to book.

In the case that it is an Owner who is requesting their own time-slot dates, the dates, if available, will automatically be set to 'approved' (as no other Owner approval is required in this case).

The requested dates go through each of the following stages:



For a complete Booking process flow, see Planning Section.

SAMPLE DATA

This project is currently set up with three assets in order to demonstrate the functionality and the separation of each asset within the site. I have used two summer homes and one vehicle.

Weekly Ownership

The first Asset is a summer house in Connemara shared by five families. There is one Owner in each family, so there are five Owners. Between these five families there are potentially 50+ family Requestors who will be invited to share this house.

The required data-points are:

- Ownership constantly rotates between five Owners
- The ownership slots are divided into two-week periods, starting on a Sunday
- The sharing start-date is Sunday, 3th January 2016

Monthly Ownership

The second Asset is a second home in France. There are two Owners and potentially 15+ sharing Requestors from three different families.

The required data-points are:

- Ownership constantly rotates between two Owners
- The ownership slots are divided into one-month periods, starting on the first of each month
- The sharing start-date is Sunday, 1st July 2016

Single Ownership

The third Asset is a small van. It owned by one Dad and is shared between his three siblings. They like to book it for emergency IKEA trips.

The required data-points are:

- Ownership does not rotate because there is only one Owner listed.
- There is one continuous ownership slot (all requests will go directly to the one Owner)
- The sharing start-date defaults to the date that the user registered the Asset



Lissaniska

5 Owners | 2 Members

Rotates: Every 2 weeks starting on a Sunday



Le Rouret

2 Owners | 5 Members

Rotates: Every 2nd month starting on the 1st



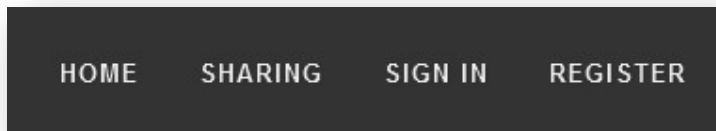
Moving Van

1 Owner | 1 Member

SITE FUNCTIONALITY PER USER TYPE

NEW VISITOR	REQUESTORS	OWNERS
Admin Tasks >>	Admin Tasks >>	Admin Tasks >>
Register Enter an Invite Code Register a New Asset	Log-In Enter an Invite Code Register a New Asset	Log-In Enter an Invite Code Register a New Asset
	Booking Tasks >>	Booking Tasks >>
	Check Availability Request a Booking View My Pending Requests View My Awaiting Confirmation View Asset Bookings/Requests View My Slot Pending Requests	Check Availability Request a Booking View My Pending Requests View My Awaiting Confirmation View Asset Bookings/Requests View My Slot Pending Requests Approve/Deny Pending Bookings Invite New Requestors/Owners
Social >>	Social >>	Social >>
	View Blog View Forum	View Blog View Forum Create a New Forum/Subject

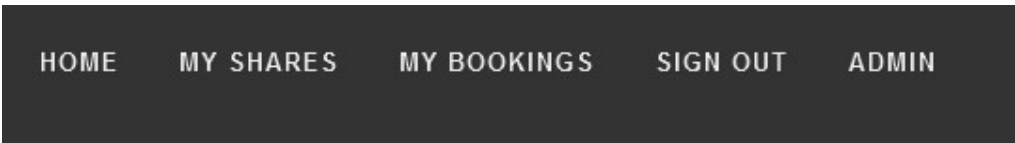
Unregistered Site Visitors



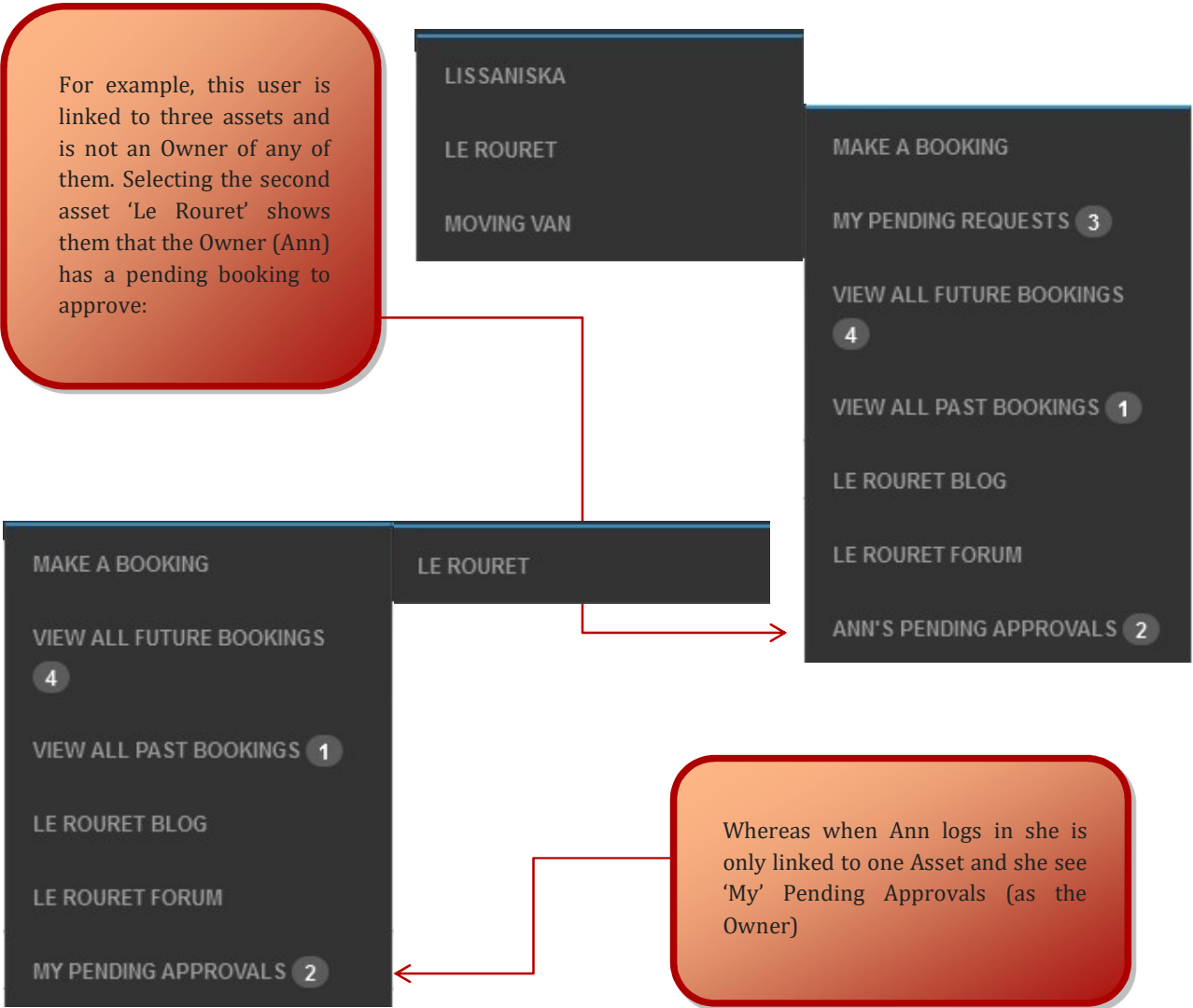
HOME	Opening page (with rotating text)
SHARING	This page gives a quick explanation of the site and what purpose it serves.
SIGN IN	Users enter their email and password in order to log in.
REGISTER	The user supplies their email, name and a password in order to register. The code checks if the username (the email address) is already registered. This is validation on missing fields and unmatched passwords.

Registered Site Visitors

Once signed-in, the user is immediately presented with their new Home Page and a new set of navigation links:



HOME	This Page is a Summary Page for the User. If they are linked to one or more Assets, these Assets will be displayed here. If they are not linked to any Assets, this is the page where they can enter any invite codes they received. After entering a valid Invite Code, the new Asset will appear on this page.
MY SHARES	A drop-down Menu of the Assets for this Requestor/Member with sub-menus related to each Asset. This menu is dynamically updated for each user. The number of assets listed depends on what Assets the user is linked to. It also changes depending on where the user is an Owner or a Member of an asset.



MY BOOKINGS	This Page shows the Requestor all their future bookings for all their Assets. While this information is available on a <i>per</i> Asset basis from the MY SHARES link, for convenience, this page shows all their bookings from all their assets in one place, showing in up-coming order.
SIGN OUT	Returns the user to the opening Home page
ADMIN	This link will only appear for the super-user and allows quick access to the Django Admin Panel pages.

MODELS

USER	holds username (email), password and name
ASSET	holds high-level information such as asset name, type of asset, sharing start date, number of owner slots, and duration type of each slot (does not hold Owner names).
ASSET USER MAPPING	links users to assets, having one record per user/per asset (joint primary keys), with an is_owner flag. If the is_owner field is True, then slot_rotation_order must also be populated.
BOOKING REFERENCES	holds the high-level information about the Booking such as requestor id and name of Asset being booked (does not hold any booking dates)
BOOKING DETAILS	holds a record for each date in the Booking Reference and also the Owner ID for each date booked (all bookings contain consecutive dates)
BLOG MODELS	all tables related to the Blog (one Blog per Asset)
FORUMS MODELS	all tables related to the Forum. Subjects, Threads and Posts (per Asset)
STRIPE DETAILS	holds each user's stripe id

PLANNING

I mocked up my vision of the site as though for a phone screen size and spent a lot of time thinking about what would be the least amount of screens/forms that a user needed to fill in in order to register themselves and their Asset. These wireframes are available on my Mocups.com site <https://classic.mocups.com/maria.hynes/o2MWzuUt>

--- Registration ---

John

Smith

john.smith.9999@gmail.com

Submit

Home

I have a code!

Sign in

Register

Registration Step 1

--- Registration ---

Asset Name:

Galway Retreat

Is Asset Shared? ☒ Yes ☐ No

Number of Owners (incl. you):

2

Submit

Home

I have a code!

Sign in

Register

Registration Step 2

--- Registration ---

Jane

Doe

Email Address (Owner 2)

Peter

Smith

Email Address (Owner 3)

Submit

Home

I have a code!

Sign in

Register

Registration Step 3

--- Registration ---

Duration of each ownership: ☐ Days ☒ Weeks ☐ Months

Units of this duration:

2

Sharing from:

04-January-2015

Based on your selections, the ownership changes every two weeks and ownership starts on a Sunday. Click Submit to confirm.

Submit

Home

I have a code!

Sign in

Register

Registration Step 4

--- Registration ---

One last thing!

Enter the Owner order:

Owner	Owner Order
John Smith	3
Jane Doe	1
Peter Smith	2

Submit

Home

I have a code!

Sign in

Register

Registration Step 5

--- Registration ---

That's it!

Go to Member Area

Home

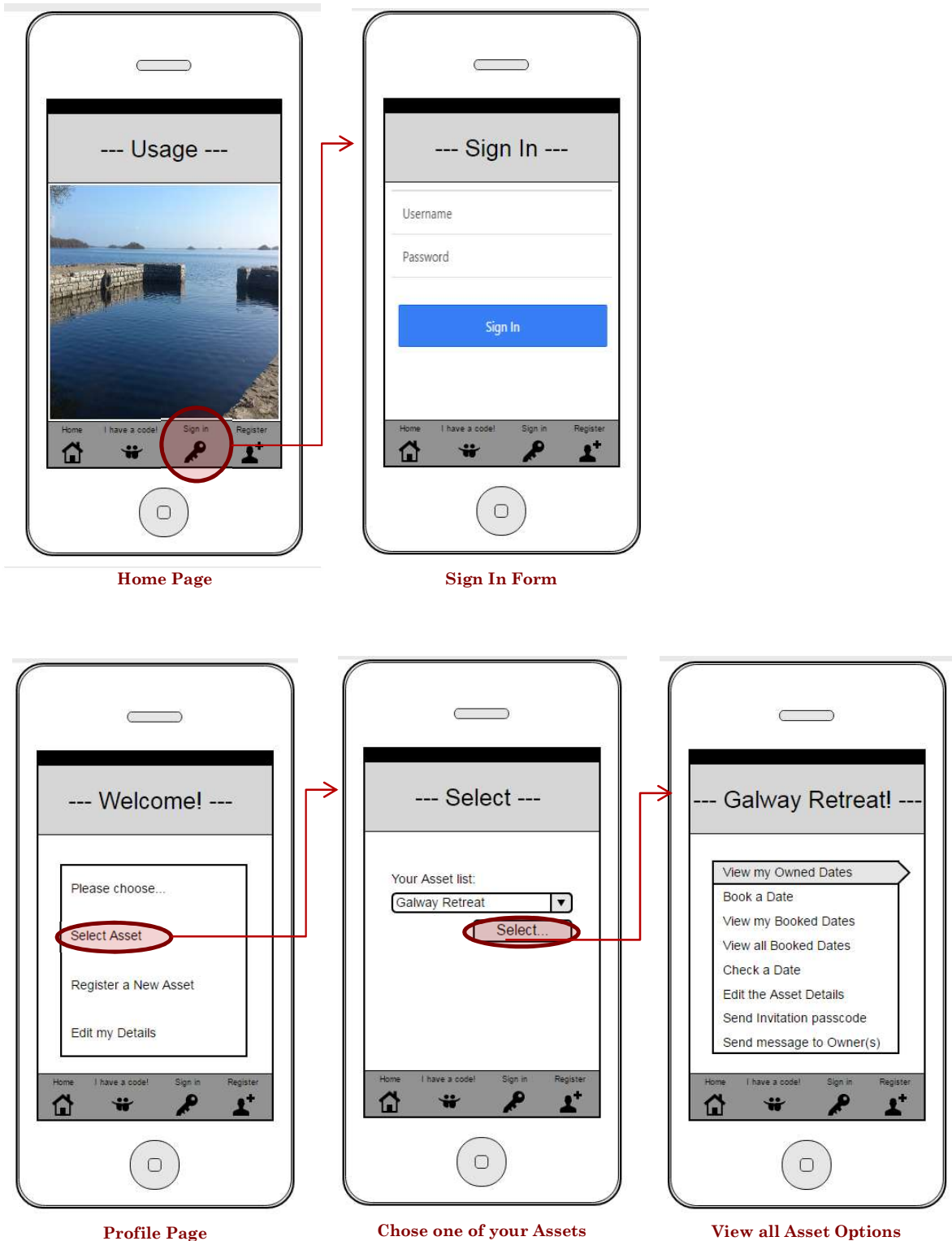
I have a code!

Sign in

Register

Registration Step 6

In the end, for this project, I decided to focus on the logic behind what would happen *after* the Asset was registered. The functionality for a user to register an Asset will be in Phase 2. The following screens show my plans to what the process would look like after logging in. I wanted the user to see their pre-registered assets as soon as they signed in and to be presented with a list of options for the Asset:



My intention was that no matter how many Assets someone is linked to, these Assets are available for selection from the user's Home screen.

Making a Booking

The planning for the booking process logic was extensive. It was important to me that the booking process for the Requestor and the subsequent approval process for the Owner, would be simple and intuitive. I discovered that in trying to making it simple for the user means that the code can easily get over-complicated when it tries to cover all eventualities. Throughout development, issues arose with my process logic which I had to overcome. I noticed also that I was often repeating similar pieces of code that I had already written for other views. So part of my development time was spent revisiting existing code to make it more generic. As a result there are many re-usable functions and reusable views in this project and many sections of code that could do with being rewritten in hindsight.

The *booking_detail.html* page is a good example where the one page caters for multiple scenarios. This one View is used in all these cases:

- 1) If any Requestor/Owner wants to view the dates in the booking
- 2) If the Owner is required to Approve/Deny the dates in the booking
- 3) If the Requestor is required to Confirm the dates in the booking
- 4) If the Requestor wants to delete dates in the booking.
- 5) If the booking is Pending, Confirmed or Awaiting Confirmation

When a booking detail page is requested, the code checks whether the user is an Owner or if they are the Requestor of the Booking. Depending on which, there are separate forms that will display the booking with radio sliders to allow the user to edit the booking in the appropriate way. For example, if it is an Owner and the request is Pending, then they will see 'Approved' and 'Denied' options on each date.

See example below with Booking Reference 84 (from Owner's perspective):

Booking | Le Rouret

Booking Reference: 84
Total days: 11
Requested By: Maria Hynes on 12 Dec 2016
Confirmed: Not Yet

Date	Status	Owner	Update To
Mon, 13 Mar 2017	Pending	Ann Hynes	Approved <input type="radio"/> No <input checked="" type="radio"/> Denied <input type="radio"/> No <input checked="" type="radio"/>
Tue, 14 Mar 2017	Pending	Ann Hynes	Approved <input type="radio"/> No <input checked="" type="radio"/> Denied <input type="radio"/> No <input checked="" type="radio"/>
Wed, 15 Mar 2017	Pending	Ann Hynes	Approved <input type="radio"/> No <input checked="" type="radio"/> Denied <input type="radio"/> No <input checked="" type="radio"/>
Thu, 16 Mar 2017	Pending	Ann Hynes	Approved <input type="radio"/> No <input checked="" type="radio"/> Denied <input type="radio"/> No <input checked="" type="radio"/>

From the Requestor's perspective, this same page looks like this:

Booking | Le Rouret

Booking Reference: 84
Total days: 11
Requested By: Maria Hynes on 12 Dec 2016
Confirmed: Not Yet

Date	Status	Owner	Remove?
Mon, 13 Mar 2017	Pending	Ann Hynes	<input type="checkbox"/> No
Tue, 14 Mar 2017	Pending	Ann Hynes	<input type="checkbox"/> No
Wed, 15 Mar 2017	Pending	Ann Hynes	<input type="checkbox"/> No
Thu, 16 Mar 2017	Pending	Ann Hynes	<input type="checkbox"/> No

If it is the Booking Requestor viewing the booking, then they will see 'Remove' on each date in case they want to remove some of the dates. They will also be given the option to delete the entire booking.



For a person viewing this booking who is *not* the Owner and *not* the Requestor, the page looks like this:

Booking | Le Rouret

Booking Reference: 84
Total days: 11
Requested By: Maria Hynes on 12 Dec 2016
Confirmed: Not Yet

Date	Booking Status	Slot Owner
Mon, 13 Mar 2017	Pending	Ann Hynes
Tue, 14 Mar 2017	Pending	Ann Hynes
Wed, 15 Mar 2017	Pending	Ann Hynes
Thu, 16 Mar 2017	Pending	Ann Hynes

Each booking is a collection of records (one row per date in the booking), so the forms on these pages take advantage of Django's Model Formset functionality. Only the Owner or the Booking Requestor sees these forms. If a normal user views the page, they will simply see the list of dates with the current status of each date, and no forms, so nothing can be edited.

Another useful and widely used function is the *get_booking* function in the Bookings views.py. This has been written to take a varying number of ****kwargs** and will dynamically formulate a query filter to suit the circumstances. At the basic level, it will always return a list of booking id and booking dates but it can be filtered by any combination of the following:

- 1) Booking requested by a given User
- 2) Bookings where a given owner is the Owner
- 3) By future or past bookings
- 4) By pending or confirmed
- 5) By Asset

This function saved writing separate functions for each type of filter. For this function I used Django's Q Object code (<https://docs.djangoproject.com/en/1.10/topics/db/queries/>)

After first thinking that for each date I only need two states (approved or pending), the following is the logic that I have applied to cover all stages from requesting to confirming a booking:

The request logic

The booking process differs only in step 4 (below) for Owners and Requestors

REQUESTOR STEPS

1. Requestor selects date range required
2. If dates are consecutive and available, Requestor can request them
3. Booking ref created and each date stored
4. All dates are set to 'Pending'
5. If the booking is viewed, all dates show the current status for each date (which will be 'Pending' at the start)
6. No one else will be able to request these dates

OWNER STEPS

1. Owner selects date range required
2. If dates are consecutive and available, Owner can request them
3. Booking ref created and each date stored
4. All dates owned by this Owner are set to 'Approved', all other dates are set to 'Pending'
5. If the booking is viewed, all dates show the current status for each date (which will be 'Approved' and/or 'Pending' at the start)
7. No one else will be able to request these dates

The approval logic

APPROVAL STEPS (only applicable for Owners)

1. Each Owner sees that they have pending requests
2. Owner opens the pending booking page and can see the list of dates that require their approval

3. Owner selects either 'Approve' or 'Deny' *per date*
4. Owner Saves the booking update
5. If the booking was entirely in this owner's time-slot, then the booking will show up for the Requestor to approve. If there are dates that other Owners still need to approve, it will remain in 'Pending' for those dates (until all Owners have followed steps 1-4)
6. If the booking is viewed, each date shows the updated status (denied/approved)
7. No one else will be able to request these dates

The confirmation logic

CONFIRMATION STEPS (the same for Owners and Requestors)

1. Each Requestor sees they have a booking to confirm (i.e. each date is either approved or denied and there are NO pending dates in the booking)
2. Requestor opens their booking and sees one of three scenarios:
 - a. all dates are approved
 - b. all dates are denied
 - c. some dates are approved and some dates are denied
3. If all dates are approved, Requestor can either confirm the booking or delete the booking
4. If all dates are denied, Requestor can delete the booking
5. If some dates are approved and some are denied, the Requestor can confirm or remove the approved dates and they must delete the denied dates
6. If the dates confirmed are not consecutive dates, then the code will automatically group each consecutive group of dates into separate bookings, creating a new booking ref for each group.
7. If any dates are denied, these dates will be deleted from the table and the dates will return to an 'available' status

The 'consecutive dates' issue

Based on the above booking steps, it is entirely possible for an Owner to deny some dates in a booking and approve others, and it is also possible for a Requestor to remove some dates in a booking and leave other dates remaining in the booking. Because this is possible (and must be allowed to be possible!) it messed with my original assumption that it should always be possible to define a Booking by a start and end date. I wanted to be able to say "this is the booking Ref and it starts on this date and ends on that date" with the understanding that all dates in between were part of the booking! But by allowing Requestors and Owners to edit the bookings, it soon became clear that I would potentially end up with a series of unconnected dates all with the one booking reference.

The solution

In order to be able to refer to a booking as a date-span I resolved the issue by creating a function that runs after a booking is edited. This function is called *make_the_split* and it's in `bookings.views.py`. It loops through every date in the booking and if it finds a date that is not consecutive, it creates a new booking (with all the same parameters as the old booking) and updates the remaining dates in the booking with the new booking ref. It continues to loop through each date in the booking and repeats the splitting as many times as it finds non-consecutive dates. So, if a user deletes three random dates from their booking, this means that the original booking is split into three booking references.

The 'multiple-owner' issue

Owners need only approve the dates that are requested in their time-slot, even if the entire booking spans *more* than their time-slot. This is the handiness of the site, that the approval process for a booking does not have to

reach every Owner, but only the owner of the time-slot affected. But, of course, there's always the situation that a Requestor can book a date range that spans the time-slots of multiple owners.

The solution

In order to work out correctly which Owner should receive the Pending request, I created a function that calculates the owner and dates in a given date range. This function, called *get_owners_and_dates* is under the Booking app in myFunctions.py. I wrote this out first in VBA, as I am very familiar with this language, in order to test the logic and then I re-wrote it (very slowly!) into Python. To test that it output the correct owner for the given dates, I created a reference Excel spreadsheet of dates and Owners for a two year span so that I could check that the output was the result I wanted to see. The function takes the asset id, a start date and an end date and returns an object of objects – all the information needed to inform the user of the Owners in the requested date span and whether each date is booked or not and by who. Here's an example of the data returned from that object. This is a requested date span where not all the dates are available:

You are checking 58 dates

They fall in 5 owner-slots

19 of these dates are unavailable

Slot Owner	Requested From	Requested Until	Available	Unavailable
Darragh Hynes	Wednesday, 21 Dec 16	Sunday, 01 Jan 17	2	9
Brendan Hynes	Sunday, 01 Jan 17	Sunday, 15 Jan 17	10	4
David Hynes	Sunday, 15 Jan 17	Sunday, 29 Jan 17	8	6
Eoghan Hynes	Sunday, 29 Jan 17	Sunday, 12 Feb 17	14	
Eddie Hynes	Sunday, 12 Feb 17	Friday, 17 Feb 17	5	

In this example, the date span covers 5 time-slots. Hovering/Clicking on the blue 'Slot Owner' will show the Requestor the start an end dates of that Owner's time-slot (this information comes from the *owners_and_dates* object):

Slot Owner	Slot Dates
Darragh Hynes	From Sun 18 Dec 16
Brendan Hynes	Until Sun 01 Jan 17

Clicking or hovering over the Red 'Unavailable' buttons will display which dates are unavailable, who has booked them and their current status (could be Pending, or Confirmed):

Unavailable Dates			Unavailable
Fri	06 Jan 17	Pending for Maria Hynes	
Sat	07 Jan 17	Pending for Maria Hynes	
Fri	13 Jan 17	Pending for Maria Hynes	
Sat	14 Jan 17	Pending for Maria Hynes	

Clicking or hovering on the green buttons will display what dates are available:

Available Dates		Available
Thursday	19 Jan 17	
Friday	20 Jan 17	
Saturday	21 Jan 17	
Tuesday	24 Jan 17	
Wednesday	25 Jan 17	
Thursday	26 Jan 17	
Friday	27 Jan 17	
Saturday	28 Jan 17	

You will see from the above screen hover that although 8 days are available, they are not consecutive so this is a way to inform the Requestor *which* dates are available/not available, without having to create a completely new page view. Again, all this information was returned from the owners_and_dates object.

On a smaller screen this table has a responsive layout, so that a scroll-bar becomes available to see all the detail:

You are checking 58 dates

They fall in 5 owner-slots

19 of these dates are unavailable

Slot Owner	Requested From	Requested Until
Darragh Hynes	Wednesday, 21 Dec 16	Sunday, 27 Dec 16
Brendan Hynes	Sunday, 01 Jan 17	Sunday, 08 Jan 17
David Hynes	Sunday, 15 Jan 17	Sunday, 22 Jan 17
Eoghan Hynes	Sunday, 29 Jan 17	Sunday, 05 Feb 17
Eddie Hynes	Sunday, 12 Feb 17	Friday, 17 Feb 17

It is only when the user revises the start and end dates to a range where all dates *are* available, that they see the [Request the Dates] button:

You are checking 7 dates

They fall in 2 owner-slots

The dates are available

Slot Owner	Requested From	Requested Until	Available	Unavailable
David Hynes	Tuesday, 24 Jan 17	Sunday, 29 Jan 17	5	
Eoghan Hynes	Sunday, 29 Jan 17	Tuesday, 31 Jan 17	2	

Request The Dates

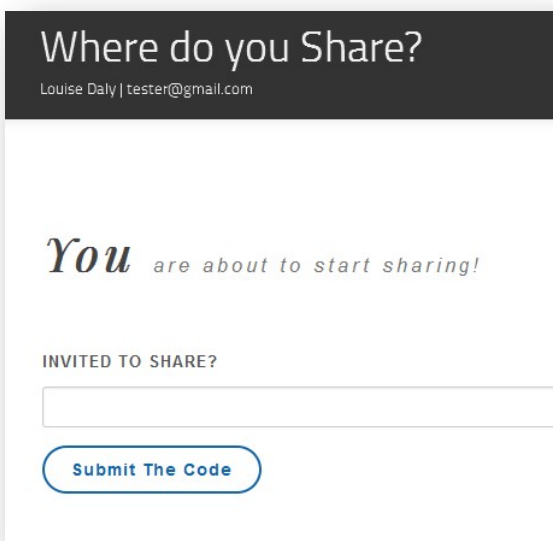
Stripe Payment Details

At the point when the user clicks to 'Request Dates', the code checks if there are stripe details on file for this Requestor. I decided not to require the user to enter these details when they first Register on the site because I wanted the Registration process to be quick. Adding Credit Card details that early on could potentially put them off signing up. Instead the user will only be asked at the point they make the booking (if they haven't already entered them on a previous occasion). The thinking behind this is that if they are invited to share something, and there is a cost associated with it, then they will already know from the Owner, that they will need to pay something (perhaps a daily rate) and will not be surprised when asked to enter their details at this point. Once the details are saved, they will not be asked to enter them again.

TESTING

User Testing

To join and test this demonstration site, please take the time to register perhaps with a couple of different email addresses. This site does not *send* emails yet so for the purposes of this demo site, the email addresses you enter do not have to be real. Once registered, you will be presented with an 'empty' Home page:



Where do you Share?

Louise Daly | tester@gmail.com

You are about to start sharing!

INVITED TO SHARE?

Submit The Code

Enter the code '11111' to link to the sample Weekly Ownership Asset and your Home Page will update:


Where do you Share?
Louise Daly | tester@gmail.com

HOME
MY SHARES
MY BOOKINGS
SIGN OUT

You are sharing...

INVITED TO SHARE?

Submit The Code



Lissaniska
5 Owners | 3 Members
Rotates: Every 2 weeks starting on a Sunday
You're linked to Owner: David Hynes
David's next slot starts: 15th Jan 2017

Enter the code '22222' to be connected with the Monthly Ownership Asset:


Where do you Share?
Louise Daly | tester@gmail.com

HOME
MY SHARES
MY BOOKINGS
SIGN OUT


You are sharing...

INVITED TO SHARE?

Submit The Code



Lissaniska
5 Owners | 3 Members
Rotates: Every 2 weeks starting on a Sunday
You're linked to Owner: David Hynes
David's next slot starts: 15th Jan 2017



Le Rouret
2 Owners | 6 Members
Rotates: Every 2nd month starting on the 1st
You're linked to Owner: Ann Hynes
Ann's next slot starts: 1st Jan 2017

Enter the code '33333' to be connected to the Single-Ownership Asset:

Where do you Share?


Louise Daly | tester@gmail.com

HOME MY SHARES MY BOOKINGS SIGN OUT


You are sharing...

INVITED TO SHARE?


Submit The Code



Lissaniska
5 Owners | 3 Members
Rotates: Every 2 weeks starting on a Sunday
You're linked to Owner: David Hynes
David's next slot starts: 15th Jan 2017



Le Rouret
2 Owners | 6 Members
Rotates: Every 2nd month starting on the 1st
You're linked to Owner: Ann Hynes
Ann's next slot starts: 1st Jan 2017



Moving Van
1 Owner | 2 Member

The text under each photo is dynamically generated (see Assets/template_tags/asset_extras.py) and comes from the Asset table.

Rotates: Every 2 weeks starting on a Sunday
You're linked to Owner: David Hynes
David's next slot starts: 15th Jan 2017

Rotates: Every 2nd month starting on the 1st
You're linked to Owner: Ann Hynes
Ann's next slot starts: 1st Jan 2017

This information is useful for all Members to know how the item is divided and more importantly what is the next time-slot for their 'Owner'. Remember, although they can request to book *any* dates, it is likely that if they book dates in their own time-slot that they will get quicker approval!

To view the site from an Owner's perspective, log in as david.hynes@gmail.com with password "123". But before you do, you should book some dates in his time-slot so that they are available for approving/denying when you/he logs in.

To view the site as the Superuser log in as maria.m.hynes@gmail.com with password "databasis". The superuser will see additional buttons on various screens allowing deletion of anyone's bookings and blog posts. This was useful during development.

Code Testing

For this project I prepared UnitTests to check that each URL resolves to the correct view. Once a user has logged in, each view thereafter can require up to four parameters from the database. The first parameter in each view is the ID of the Asset and each parameter after that needs to be *related* to that asset id. So to test this, I created json file fixtures. In the test, an asset id is pulled from the fixture, and then this asset id is used to pull a related field from the next table. For example, for a Forum page, the URL might be "forum/forum/2/5/1/18" and for the view to resolve correctly each parameter must be linked. This means that post '18' must be part of thread '1', which must be in subject '5' for the forum in asset '2'. Likewise, to resolve a Blog post correctly, when the URL is "blog/blog/2/14", the blog post '14' *must* be linked to asset id '2'. Using the fixtures files, this is correctly tested.

Blackbox Testing

The basic validation is in place before each view, to ensure that certain views are only available to someone who is logged in. However, because the site caters for more than one asset, it was not enough to check whether the user is or is not logged in. I realised very early on in development that unfortunately *any* logged-in user could amend the URLs to view the blogs and forums for Assets even if they weren't linked to those Assets. So I wrote additional functions that are used in the views when each URL is accessed. These are stored in the Home App under MyAuth.py. The forum and blog apps have been updated to include the asset id in each request. On this site now, no one can view data for an asset if they are not 'linked' to that asset. So even if user alters the URLs or creates URLs that might show them another asset's data, the site validates every request to ensure that the user is entitled to see the page. The user will see a "You are not authorised..." message if they try to view a page they shouldn't be viewing.

The 'Sign In' and 'Register' form fields are tested for missing fields and duplicate passwords. When first-time registering, the email address (username) is also tested to see if it has already been used in order to avoid a duplicate primary key error.

I used the Python Console to test each query prior to using it in the code as a quick way to ensure if the correct data would be pulled. Many print statements in the code also helped when testing to see that the code stepped into each section correctly.

NEXT STEPS

For the purposes of this Project, in order to demonstrate these three sample Assets, I manually populated the Asset tables with the initial Asset data using the Django Admin Panel. When this site goes live, the intention is that the initial setting-up of Assets will be carried out by the Owners themselves so that the entire process is automated and anyone anywhere will be able to use this site to manage their Assets. Also for this Project I have pre-linked the existing set of users to particular owners. Any test users who use the Invite Code mentioned above, are also linked to one particular owner. In the future, this will be performed by an emailed invitation link whereby an Owner of an asset will be given functionality to send an 'invite code' to a User.

REFERENCE

Github	https://github.com/mariahynes/HouseShare
Moqups	https://classic.moqups.com/maria.hynes/o2MWzuUt
Reference Dates File	Date_Sample_for_testing.xlsx (in GitHub repo)
This documentation	Project_3_FINAL_DOCUMENTATION.pdf (in GitHub repo)