CS353
Group 15 - Campus Life
Man Yiu Lam - 16458032

Contents

Chapter 0:

Introduction to Agile Development (SCRUM)

SCRUM – An Agile Framework

SCRUM is a framework used in industries to develop products, (most commonly used for Software Products), to bring to market very quickly. SCRUM aims to deliver "the highest business value in the shortest time" [1].

During the late 1990s and early 2000s there was a substantial increase in the amount of research centring on finding a more adaptable software development process. This resulted in many publications and articles being produced on topics such as "on agile software development methodologies, reflecting a growing interest in…new approaches to software development", namely Extreme Programming and SCRUM. The interest was furthered by the formation of the Agile Alliance, "a group of expert consultants and authors on development process". [2, p. 1]

SCRUM uses techniques such as Burndown charts opposed to the traditional static PERT charts. Both charts are used to estimate the rate of development i.e. the critical path. However, with the agile nature of SCRUM, PERT charts can become "obsolete within 24 hours", whereas Burndown charts allow for the estimate to be continuously updated. [2, p. 1]

Dr. Jeff Sutherland recognised that during the software development cycle it was often the case that the requirements of clients would alter from the original specification. This is something that the SCRUM framework already assumes and welcomes. [2, p. 3] "It supports Humphrey's Requirements Uncertainty Principle which states that for a new software system, the requirements will not be completely known until after the users have used it ." [2, p. 2]

The first of SCRUM implementation was in 1993 at Easel Corp where Dr. Jeff Sutherland was the VP of Object Technology. "The first object-orientated design and analysis tool that incorporated round-trip engineering" was designed and built using SCRUM. [2, p. 3] This served as a proof of concept to the world that SCRUM was and is a strong contender to their current software development process.

Ken Schwaber, a software engineer and author, "co-developed the Scrum framework with Jeff Sutherland in the early 1990s to help organizations struggling with complex development projects. One of the signatories to the Agile Manifesto in 2001, he subsequently founded the Agile Alliance and Scrum Alliance." [3] Schwaber SCRUM was also presented the SCRUM framework at the OOPSLA 96 with Sutherland. OOPSLA 96 was the 11th conference held in 1996. OOPSLA, which stands for "Object Orientated Programming, Systems, Languages & Applications, is an annual research conference that focuses "on object-oriented programming systems." [4] No doubt SCRUM was gaining traction.

The results of these strategic campaigns are evident today. SCRUM has been implemented and adapted by many companies to fit their unique work style. These companies include: Microsoft, Google, IBM, Electronic Arts, Lockheed Martin and Philips. The most common use case for SCRUM would be projects that involve Commercial Software, Contract Development, Websites, Mobile Phones, Financial Applications and FDA-Approved Life Critical Systems. [1]

SCRUM teams are self-coordinated under “business set priorities” “to determine the best way to deliver the highest priority features.” SCRUM is comprised of sprints or “leaps” as it were. Each sprint tends to last from 2 weeks – 1 month where at the end something potentially shippable will either be released or further enhanced for another sprint. *[1]*

SCRUM adheres to the Agile Manifesto, which are: [1]

Communication

Interaction: rather than simply following a process, communicating to identify any potential restrictions to the project and allow external feedback to influence the project.

Product owner(s) and/or customer(s): rather than centring the entire project on a fixed negotiated contract before the project commenced, SCRUM teams regularly communicate to ensure the current requirements are up-to-date.

Structure

Working Software: rather than producing extensive documentation at each step of the project, i.e. the waterfall model. SCRUM focuses on producing working software and allows the product to do the talking.

Agile

Respond to Change: ready to change direction when change is necessary and not stuck following a rigid plan.

The SCRUM Framework

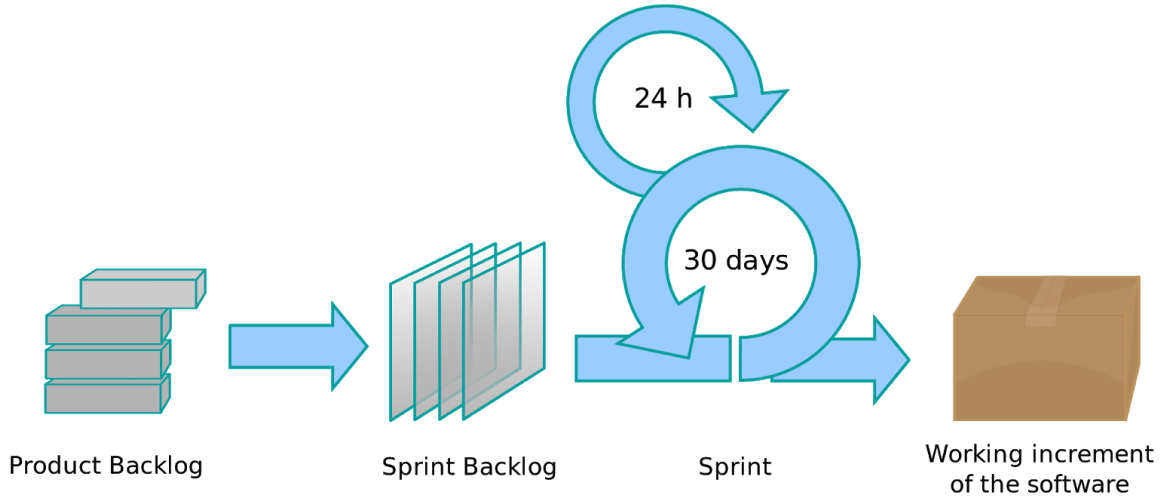


Figure 1. SCRUM Framework from www.adventureswithagile.com

Product Backlog: documentation carefully constructed by the product owner to outline their vision of what the product should be. It will consist of a list of features and requirements.

Sprint Backlog: documentation based off the product backlog that specifies what is to be completed by the end of a sprint.

All sprints in a given project will last the same amount of time. This leads to a better rhythm for the team. **Four** stages are completed in a sprint. Review the **Requirements**. **Design** the product, planning and researching. Write the **Code**. **Test** the product, ensuring it meets the requirements. “Scrum teams do a little of everything all the time.” [5] During the sprint, no changes are made. The SCRUM teams commit to a plan for the duration of the sprint and review the working increment of the software product at the end. Open to feedback and welcome change if necessary. [1]

There are three main roles in the SCRUM frame work: [1]

Product Owner

The person who defines the features of the product and decides the release date and content.
The person who is responsible for the profitability of the product.
They priorities features according to market value and adjust with every iteration where needed.
They can accept or reject work results.

ScrumMaster

Represents management to the project.
Ensures SCRUM values and practices are enacted.
Removes impediments.
Ensures the team is productive.
Enables close co-operation across all roles and functions.
Shields the team form external interferences.

Team

Typically 5-9 people in size.
Varying in skills. i.e. programmers, testers, UX (user experience) designers, UI (user interface) designers, etc.
Self-organising and restructure only between sprints.
Scalable. A large work force can be subdivided into smaller SCRUM teams of SCRUM teams.
Factors that limit scalability:

1. Type of Application
2. Team Size
3. Team Dispersion

4. Project Duration.

Figure 2. SCRUM Planning from www.khatrishashank.wordpress.com

Before the sprint starts a Sprint Planning Meeting is conducted. Here, the team selects from the product backlog what they will commit to completing by the end of the sprint. From this, the sprint backlog is created. Tasks are identified and are given an estimated time till completion. This is done as a team. [1]

As was mentioned earlier, SCRUM relies heavily on constant communication, henceforth, SCRUM meetings are held daily. These meetings are at a max 15 minutes in length and are designed not to be formal. These meetings are open to the public to listen in. However, only the ScrumMaster, Product Owner and team members can talk. [1]

Three main questions are raised to each member during the meeting: [1]

1. What work did they do the previous day?
2. What work do they plan to do today?
3. Was there anything that restricted their progress? If so, what can be done to remove the impediment?

At the end the sprint, a review is carried out. The team presents their accomplishment in the form of a demo featuring the new features in an informal setting. Teams must follow the two-hour prep time rule and no slideshows are allowed. All team members participate. The public are welcome to observe. Sprint retrospectives are conducted at the end of every sprint. In general, these are 15-30 minutes meetings that reflect on what the team is doing right and on what the team

can improve on or change. Each sprint should have a goal. “A short statement of what the work will be focused on during the sprint.” [1]

Focusing on the Product Backlog, this document must list all desired work to be completed on the project. Each item should be associated with a metric that relates to how much the users or customers value the item of the product. This document is “prioritised by the product owner” and “reprioritised at the start of each sprint”. [1]

Sample Product Backlog

ToDo List		
Story	**Estimation**	**Priority**
As a user I want to be able to reset my password	1	1
As a user I want to edit items	3	2
As a user I want to export data	2	3
As an administrator I want to define KPI's for my sales team	4	4
As a user I want to view my data on mobile	5	5
As an administrator I want to send alerts when new leads come in	2	6
As a user I want to create a report of my data	5	7
As a user I want to update my reminder settings when a date is added	3	8
As a user I want filtering enhancements	4	9
As an administrator I want to configure views of data	5	10
Total	**34**	

Figure 3. Sample Product Backlog from www.smartsheet.com

When it comes to the Sprint Backlog, work is never assigned. Team members are free choose what work they wish to do. The estimated work remaining is updated daily. Any team member can edit the sprint backlog and should update work remaining.

Sample Sprint Backlog

User Story	Tasks	Day 1	Day 2	Day 3	Day 4	Day 5	...
As a member, I can read profiles of other members so that I can find someone to date.	Code the ...	8	4	8	0		
	Design the ...	16	12	10	4		
	Meet with Mary about ...	8	16	16	11		
	Design the UI	12	6	0	0		
	Automate tests ...	4	4	1	0		
	Code the other ...	8	8	8	8		
As a member, I can update my billing information.	Update security tests	6	6	4	0		
	Design a solution to ...	12	6	0	0		
	Write test plan	8	8	4	0		
	Automate tests ...	12	12	10	6		
	Code the ...	8	8	8	4		

Figure 4. Sample Sprint Backlog www.i.pinimg.com

Sample Burndown Chart

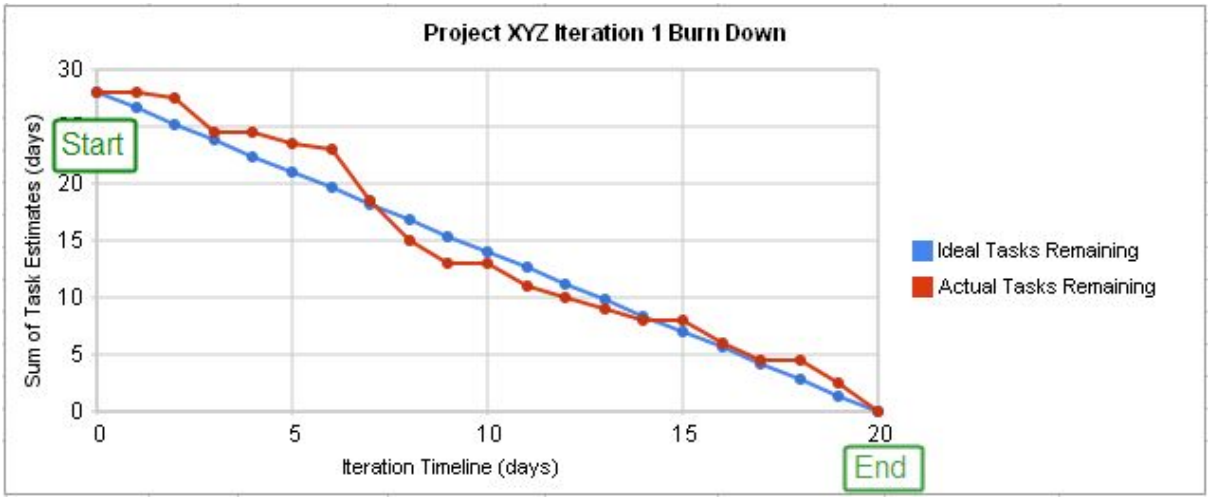


Figure 5. Sample Burndown Chart form www.wikimedia.org

Here is an example Burndown Chart. The work needed to be completed takes up space on the y-axis. This is proportional to the time needed to complete the work. This can be estimated using any metric that the team can understand i.e. hours, days, etc. These estimations can be made using a model-based system such as COCOMO for instance. Which are models are built from observations. Estimations can also be expert based e.g. Wideband Delphi. Planning poker is a variation of Wideband Delphi where a group of experts are given a deck of cards and the moderator reads a description of a user story. The experts ask questions and the product owner answers them. Each expert/estimator selects a card (their estimate value) and places it faced down. When all estimates are in, the cards are flipped. If the estimates vary widely then the owners of the extremities reason their estimates.

They repeat placing cards down and flipping them until the estimates are all within a reasonable range. This process tends to yield more accurate estimates opposed to individual estimates.

SCRUM is a friendly framework that overcomes some of the short falls of other software development processes such as customer demands changing over time and lack thereof regular communication.

Works Cited

[1] K. Casey, "Assistant Lecturer," *An Itroduction to Scrum,* vol. 1, no. 1, 2018.

[2] D. J. Sutherland, "CTO, PatientKeeper, Inc.," *Inventing and Reinventing SCRUM in Five Companies,* 2001.

[3] K. Schwaber, "Bio," *Scrum.org.*

[4] OOPSLA.org, "OOPSLA History," OOPSLA.

[5] T. &. Nonaka, "The New Product Developement Game," *Harvard Business Review,* 1986.

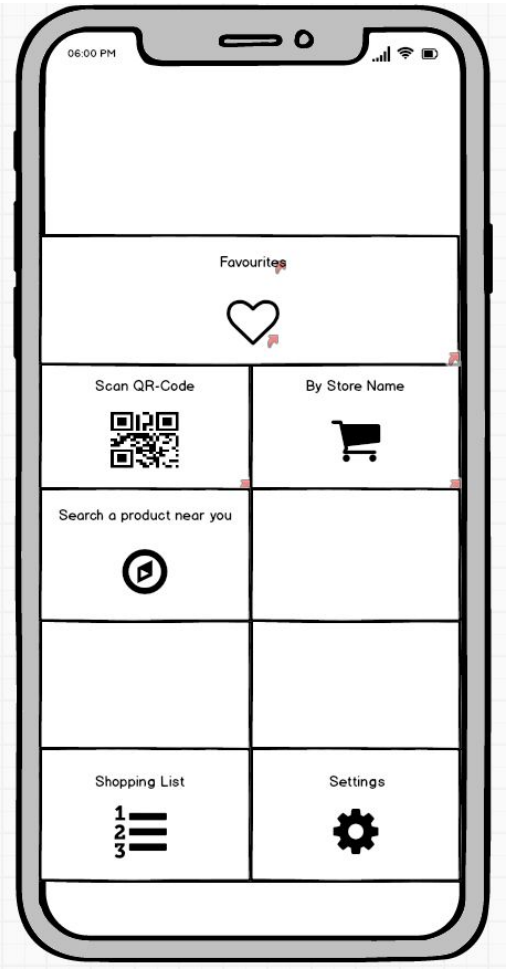
Chapter 1:

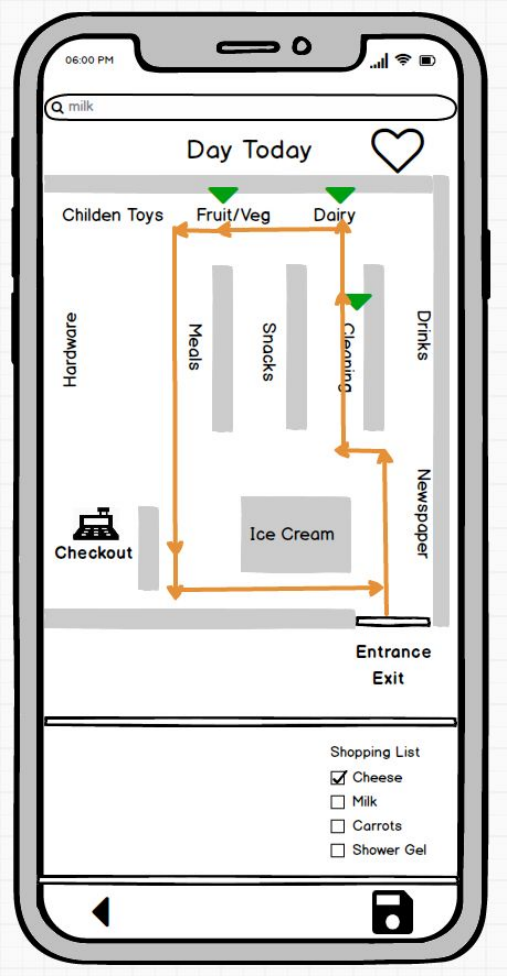
Project Proposals

My first project proposal was originally a shopping application that would take its first step on a web browser and later implemented as a mobile app if it was successful. This was an idea my friend and I came up during the start of Summer, we worked on a mockup of this application and created simple user cases before I paused the project to do my internship. I later discovered that three out of five members wouldn't had been able to code any of this and scraped this idea.

I actually really liked this idea of a shopping app and the demonstrators had said it was an unique, good and expandable idea.
The shopping application would be used for many things, such as discounts within stores (QR code), shopping list route, item locator in a region, easy meal builders and much more.

My apologies that the mockups are mobile based but would be somewhat the same on a browser. Also Meal builder was later added.





The key functionality of this application is to save time and shop more conveniently. I implemented other features so that this application can have more to expand on. A quick guide to this application is when a user is picking items from a shop (sorted in the application into categories and icons) and placing it into a shopping list. The shortest route algorithm would locate all the items in the shopping list within the shop via a map, then draw the shortest path for the user to gather all the items, purchase and then exit the shop. There would also be pricing for every product and to tell the user if a product is out of stock. This algorithm would be similar to a project I have done in 2nd year known as the TSP problem. It would simply locate and price all the items and return the path.

Another cool feature would be the meal builder. The slogan for something like this would be "Eat healthy, Stay healthy". Users that are new to cooking or want to try some new type of food meals can access the meal builder section of the app which would breakdown the meal into ingredients. This allows the user to add the ingredients into the shopping list, if they purchase something out of the meal planner, we may give them future discounts! Later they can follow the rest of the guide to prepare their meal and give reviews etc.

Another important feature is the discounts within a shop. It could be anything from seasonal sells to weekly sells. This will show the users which items are on sale and how many are left within the shop. It could also be used to push close to expiry date items to be sold cheaper.

I've planned to gather information from a smaller shop and include the data into our database as a test case. I've actually achieved a small bit of this over summer. I have talked to a demonstrator about the item locator within a shop, rather it could be done by the manager or a hired professional… or it could be done by users of the app, in return they would get discounts and honour points.

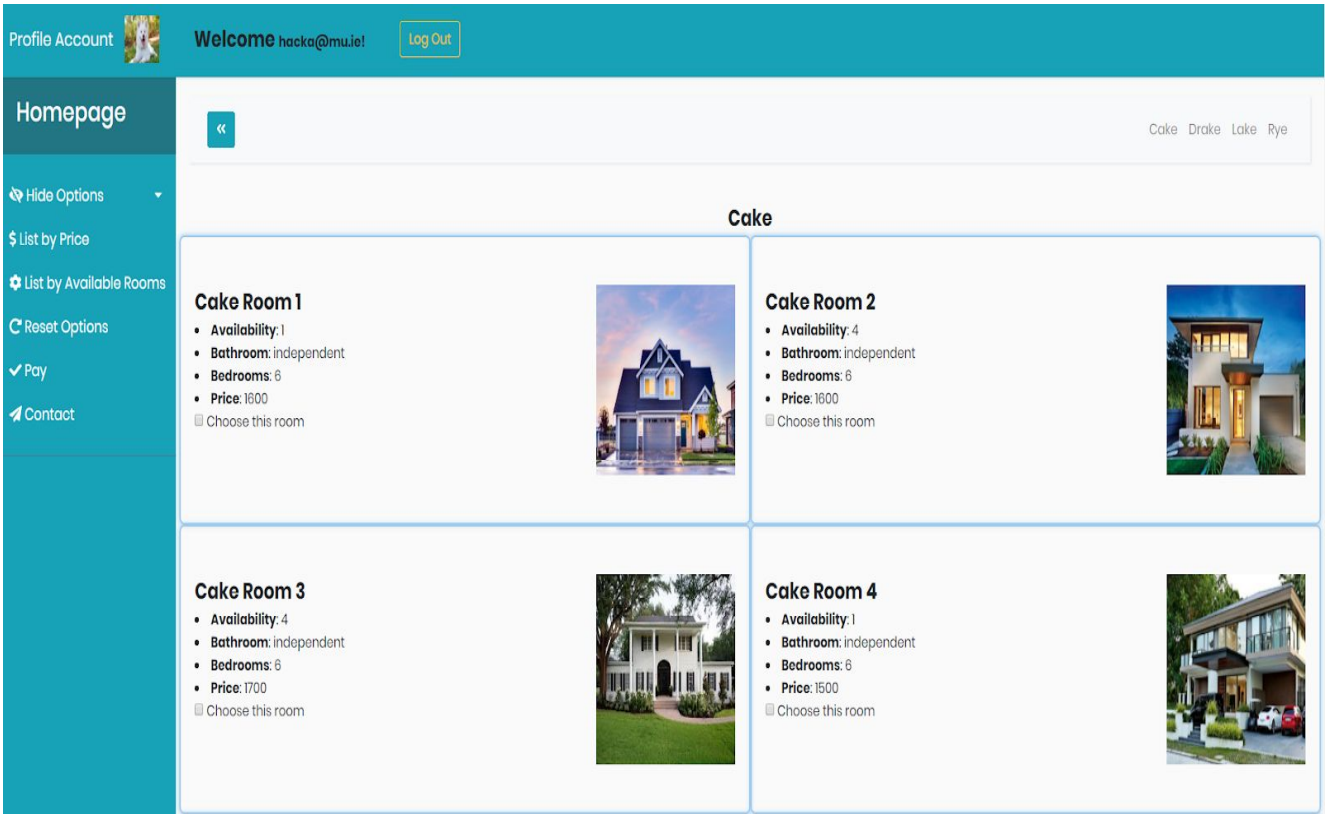
I felt like this project could had been a true success, it would had be an app that I could still work on after college. I regret not pushing this idea out further but I doubt that it would had been achievable by considering my team's effort.

The second idea came from one of my teammates and myself. It was taken from an interview question I done at Hubspot, design your own URL Shortener just like bitly. I thought that this idea would be good and expandable into any sites that required a lot of links. There wasn't a lot to say about this idea, just the fact that it could had been an interesting and a fun one. It would be something unique and complex enough that we would require to think about it.

Project Chosen

The final chosen project is a Student accommodation website which allows users to reserve and pay for their student rooms on campus before a given date. It was planned to allow easy access and updated features that the current website has. The web application would consist of a login page and registration page, ordering page, profile page and an admin page. The purpose of this website is to allow students from different countries and regions to secure a place on campus before arriving to the college.

The core features in this website is student login, registration, payment of room, and storing all the data into firebase. Admins can also modify the rooms by adding more, removing or changing them.



I personally wasn't a big fan of this idea but the votes was towards it. I considered the idea to be too simple, non expandable and a replica of what already exists.

Chapter 2:

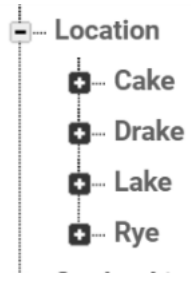
Backend-part:

```
//initialize firebase
const config = {
    apiKey: "AIzaSyBFpk2n34wvReaeKeZ7KKIp2r9f9fCAr7c",
    authDomain: "groupproject-712e1.firebaseapp.com",
    databaseURL: "https://groupproject-712e1.firebaseio.com",
    projectId: "groupproject-712e1",
    storageBucket: "groupproject-712e1.appspot.com",
    messagingSenderId: "785543171942"
};
firebase.initializeApp(config);
```

The first thing is to connect to the firebase and made some data examples in the firebase for test. When it succeeds to download and display the data we started the development of our project.

Main page:

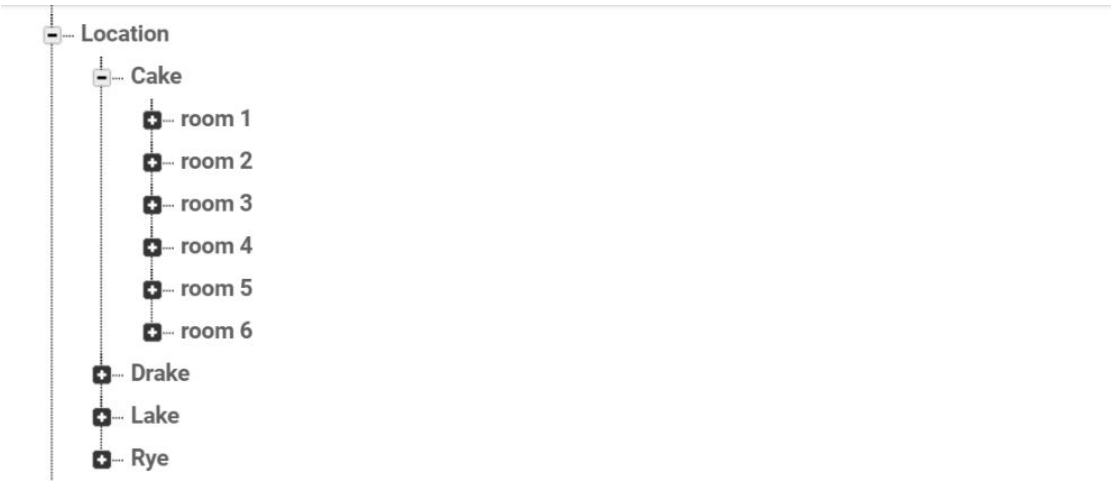
```
(function() {
    const dbRef = firebase.database().ref('All/Location');
    dbRef.on('child_added', snap => {
        const buildingList = document.getElementById('buildingList');
        const div = document.createElement('div');
        const b = document.createElement('b');
        b.innerHTML = (snap.key).fontsize(5);
        div.appendChild(b);
        div.setAttribute("class", "building");
        div.id = snap.key;
        buildingList.appendChild(div);

        const sideBar = document.getElementById('homeSubmenu');
        const sides = document.createElement('li');
        const sides_a = document.createElement('a');
        sides_a.onclick = hideFunction;
        sides.id = snap.key + '_sides';
        sides.setAttribute("name", snap.key);
        sides_a.innerHTML = snap.key;
        sides_a.style.cursor = "pointer";
        sides_a.setAttribute("name", snap.key);
        sides.appendChild(sides_a);
        sideBar.appendChild(sides);
```



This snippet is for the functionality to download and load data from firebase and then display on the main page. It will automatically generate many div tags first (the number of div tags according to how many building nodes in firebase) and set some attributes. At the same time it will automatically generate a li tag for sidebar as well, every li tag will invoke an onclick function (it can hide the building if users are not

interested).

```
let index = 1;
let queue = [];
snap.forEach(childSnap => {
    var roomkey = childSnap.key;
    const ul = document.createElement('div');
    div.setAttribute("style", "padding-bottom: 50px");
    div.appendChild(ul);

    ul.innerHTML = (jsUcfirst(snap.key).bold() + " " + jsUcfirst(roomkey)).bold().fontsize(
    ul.setAttribute("building", snap.key);
    ul.setAttribute("class", "col-sm");
    ul.setAttribute("style", "text-align: left;padding-top: 50px;padding-left: 20px;padding
    ul.id = roomkey;

    //div.appendChild(ul);
    if(index % 2 === 0) {
        const row = document.createElement('div');
        row.setAttribute("class", 'row');
        queue.push(ul);
        div.appendChild(row);
        while(queue.length !== 0) {
            row.appendChild(queue.shift());
        }
    } else {
        queue.push(ul);
    }
    index++;
```



This snippet is the part of the functionality before. The last page is just to load building, and this one is to traversal all the rooms in every building and then display on the page. Using a little bit of knowledge of data structure (queue), because we want every two rooms in a row it can reduce the length of html and more reasonable arrangement of rooms. The idea of it is using an index to count the number of this room, if the index is even that push it into the queue array and then poll all the rooms into the row, because index is started from 1 so the first room will be pushed in queue (Here we are using the attribute of bootstrap row).

```
    childSnap.forEach(attributes => {
        if(attributes.key == 'aimage') {
            const img = document.createElement('img');
            img.setAttribute("src", attributes.val());
            img.setAttribute("style", "float:right");
            img.style.height = '200px';
            img.style.width = '200px';
            ul.appendChild(img);
        } else if(attributes.key == 'price' || attributes.key == 'availability') {
            const span = document.createElement('span');
            const li = document.createElement('li');
            li.innerHTML = jsUcfirst(attributes.key).bold() + ': ';
            li.appendChild(span);
            li.setAttribute("name", attributes.key);
            ul.appendChild(li);
            span.innerHTML = attributes.val();
        } else {
            const li = document.createElement('li');
            ul.appendChild(li);
            li.setAttribute("name", attributes.key);
            li.innerHTML = jsUcfirst(attributes.key).bold() + ': ' + attributes.val();
        }
    });

    checkBoxDiv = document.createElement('div');
    const selected = document.createElement('input');
    selected.setAttribute("type", "checkbox");
    selected.setAttribute("id", roomkey);
    selected.setAttribute("name", snap.key);
    checkBoxDiv.appendChild(selected);
    checkBoxDiv.innerHTML += '  Choose this room';
    ul.appendChild(checkBoxDiv);
});
);
```

This snippet is to load all the attributes of the room and set some html and css attributes, and we want every room has a unique image so it's easier to recognize. So we store the link of image in firebase and create an image tag and set a href attribute. Every room will generate automatically a checkbox to allow user to choose this room. Because only checkbox can be canceled and there is a functionality to guarantee the user only can choose one room.

```
    //change included(replace and change)
    dbRef.on('child_changed', snap => {
        const building = document.getElementById(snap.key);
        building.innerHTML = '';
        const b = document.createElement('b');
        b.innerHTML = (snap.key).fontsize(5);
        building.appendChild(b);

        let index = 0;
        let queue = [];
        snap.forEach(childSnap => {
            var roomkey = childSnap.key;
            const ul = document.createElement('div');
            building.setAttribute("style", "padding-bottom: 50px");
            building.appendChild(ul);

            ul.innerHTML = (jsUcfirst(snap.key).bold() + " " + jsUcfirst(roomkey)).bold().fontsize(
            ul.setAttribute("building", snap.key);
            ul.setAttribute("class", "col-sm");
            ul.setAttribute("style", "text-align: left;padding-top: 50px;padding-left: 20px;padding
            ul.id = roomkey;

            //div.appendChild(ul);
            if(index % 2 != 0) {
                const row = document.createElement('div');
                row.setAttribute("class", 'row');
                queue.push(ul);
                building.appendChild(row);
                while(queue.length != 0) {
                    row.appendChild(queue.shift());
                }
            } else {
                queue.push(ul);
            }
```

This functionality almost is the same as the one before. The difference between these two functions is the last function only the first time to load data, but this one is to monitor the firebase, if the number of room changed or the attributes of the room changed, it will load again. The child node of that reference (delete, add and change), but there are four building nodes and many rooms so that we are unable to write functionalities for every one so the idea is to reload all the data again to avoid it.

```
//remove function(if building layer is not a object)
dbRef.on('child_removed', snap => {
    const removedElement = document.getElementById(snap.key);
    removedElement.remove();
});
```

And this functionality is to monitor the building list, if remove a building node at firebase, it will cause a display error on the page.

```
const rooms = document.getElementsByTagName("input");
function checkCheckboxs () {
    //check room only one can be choose
    let count = 0;
    let selectedNumArray = [];
    for(let i = 0; i < rooms.length; i++) {
        if(rooms[i].checked) {
            selectedNumArray[count] = i;
            count++;
        }
    }
    if(selectedNumArray.length > 1) {
        alert("Only one room you can choose!");
        return false;
    } else if(count == 0) {
        alert("You have not choose any room yet!");
    } else {
        //console.log(selectedNum.length);
        const selectedNum = selectedNumArray[0];
        bookRoom(selectedNum, count);
    }
}
```

This functionality is to guarantee the user only can book one room. There is a for-loop to check every checkbox, if it was checked and count one. If count is more than one, it will alert user only one room you can choose and finish this function. If count is 0 that is represented the user have not choose any room yet then to alert user. Only if count is one so it will invoke booking functionality and pass the parameter to it

```
function bookRoom (selectedNum) {
    //inspect the parameter which one is choose#
    if(rooms[selectedNum].checked && confirm("Are you sure choose this room?")) {
        const buildingName = rooms[selectedNum].name;
        const roomKey = rooms[selectedNum].id;
        firebase.auth().onAuthStateChanged(firebaseUser => {
            if(firebaseUser) {
                //email cannot as the key in firebase
                const email = firebaseUser.email;
                queryStudentId(email, buildingName, roomKey);
            } else {
                //
                alert("You are not logged in");
            }
        });
    }
}
```

This functionality receive a parameter passed by the function before, it will check again which room the user selected and let user confirm. Before passing the details of this room to the next functionality, to query user are logged or not, if not will alert user you are not logged in to guarantee the safety.

```
function queryStudentId(email, buildingName, roomKey) {
    console.log(2);
    const testRef = firebase.database().ref("/All/StudentList");
    testRef.orderByChild("email").equalTo(email).on("value", snap => {
        snap.forEach(childSnap => {
            const studentId = childSnap.child("studentId").val();
            const name = childSnap.child("name").val();
            if(studentId != null && studentId != "") {
                //console.log(studentId);
                console.log(0);
                confirmRepeatSelect(studentId, name, buildingName, roomKey);
            } else {
                console.log("Not found");
            }
        });
    }, error => {
        if(error) {
            console.log(error);
        } else {
            console.log("successful");
        }
    });
}
```

This snippet is to query id of the user and then pass the details of the user to the next functionality. Because authentication part the firebase provided is only using email and password, and cannot store any other information. Every node in firebase the key cannot include any character out of letters, so we decide to use student id as the key to store. As it ask the user when they signing up they have to type in their student id.

```
function confirmRepeatSelect (studentId, name, buildingName, roomKey) {
    console.log(studentId);
    const ref = firebase.database().ref('All/BookedList');
    ref.once("value").then(snap => {
        if(!snap.child(studentId).exists()) {
            checkAvailability(studentId, name, buildingName, roomKey);
        } else {
            alert("Sorry, you have booked room!");
        }
    });
}
```

This functionality is to query student is booked room or not, ours idea is every student only can book one room so to check the record, if there is no record of that student that can book room.

```
function checkAvailability(studentId, name, buildingName, roomKey) {
    const ref = firebase.database().ref('All/Location'+'/'+buildingName+'/'+roomKey);
    ref.once('value').then(snap => {
        const availability = snap.val().availability;
        //var price = snap.val().price;
        if(availability > 0) {
            updateAvilability(buildingName, roomKey, availability);
            recordTheRoom(studentId, name, buildingName, roomKey);
        } else if(availability < 1) {

            $('input[type="checkbox"]:checked').prop('checked',false);

            alert("This room was full!");
        }
    });
}
```

This functionality is to check availability of the room user choice, if availability is more than 0 that can be booked, and pass the parameters to two functionalities. One to record the details of room and user to booked list, another one is to update the availability of that room. Otherwise will alert user this room was full. Because firebase is real-time database, so we make better use of this feature that will greatly improve the user experience.

```
function updateAvilability(buildingName, roomKey, availability) {
    const after = availability - 1;
    let updateData = {
        availability: after
    };
    let path = {};
    path['All/Location'+'/'+buildingName+'/'+roomKey+'/availability'] = after;
    firebase.database().ref().update(path);
    //callback function here
    window.location = 'order.html';
}

function recordTheRoom(studentId, name, buildingName, roomKey) {
    firebase.database().ref('All/BookedList/' + studentId).set({
        name: name,
        buildingName: buildingName,
        roomKey: roomKey,
        semester: 2
    });
}
```

```
function hideFunction() {
    let name = this.name;
    const rooms = document.getElementsByClassName('col-sm');
    for(let i of rooms) {
        if(name == i.getAttribute('building')) {
            if(i.style.display != "none") {
                i.style.display = "none";
            } else {
                i.style.display = "block"
            }
        }
    }
}
```

The idea before is to hide the whole div tag which name is equal to the user choose, but at the end we add two sorting functions there, if sorting then to hide it will cause the problem that cannot works this function, so changed to hide all the rooms belongs to that building.

```
function listByPrice() {
    const list = document.getElementsByClassName('col-sm');
    let newList = document.getElementById('buildingList');
    const result = [...list];
    result.sort((a,b) => {
        return a.getElementsByTagName('span')[1].innerHTML - b.getElementsByTagName('span')[1].inne
    })
    newList.innerHTML = "";
    let queue = [];
    for(let i = 0; i < result.length; i++) {
        if(i % 2 != 0) {
            const row = document.createElement('div');
            row.setAttribute("class", 'row');
            queue.push(result[i]);
            newList.appendChild(row);
            while(queue.length != 0) {
                row.appendChild(queue.shift());
            }
        } else {
            queue.push(result[i]);
        }
    }
}
```

```
function listByAvailability() {
    const list = document.getElementsByClassName('col-sm');
    let newList = document.getElementById('buildingList');
    const result = [...list];
    result.sort((a,b) => {
        return b.getElementsByTagName('span')[0].innerHTML - a.getElementsByTagName('span')[0].inne
    })
    newList.innerHTML = "";
    let queue = [];
    for(let i = 0; i < result.length; i++) {
        if(i % 2 != 0) {
            const row = document.createElement('div');
            row.setAttribute("class", 'row');
            queue.push(result[i]);
            newList.appendChild(row);
            while(queue.length != 0) {
                row.appendChild(queue.shift());
            }
        } else {
            queue.push(result[i]);
        }
    }
}
```

These are two sorting functionalities, sorting by price or availability. When we doing this encountered a very important problem is that we want to collect nodes to an array, but it is not array type and it hasn't iterator so we cannot sort them directly. It's html collection. So we spent some time to research what is the difference between html-collection, node-collection and array. Then we using ES6 grammar to transfer it to array and rewrite the sorting function (according to the price or availability of that node) to sort them.

Before we rewrite this sorting function, we wrote a merge sort, because we just know sorting function is for array type, but the array we collected is html tags, and the parameter we wanted is inside in tag. So we decided to use extra sorting function. But we found we can rewrite sorting function ES6 provided that we changed it. It saves amount of work.

```
function mergeSort(array, number) {
    if(array.length < 2) {
        return array;
    }
    let mid = parseInt(array.length / 2);
    let left = array.slice(0, mid);
    let right = array.slice(mid);
    return merge(mergeSort(left, number), mergeSort(right, number), number);
}

function merge(left, right, number) {
    let result = [];
    let low = 0, high = 0;
    while(low < left.length && high < right.length) {
        if(number == 1) {
            if(left[low].val().price > right[high].val().price) {
                result.push(right[high++]);
            } else {
                result.push(left[low++]);
            }
        } else {
            if(left[low].val().availability < right[high].val().availability) {
                result.push(right[high++]);
            } else {
                result.push(left[low++]);
            }
        }
    }
    while(low < left.length) {
        result.push(left[low++]);
    }
    while(high < right.length) {
        result.push(right[high++]);
    }
    return result;
```

Management Page:

```
(function () {
    //bookedlist loading part
    const bookedList = document.getElementById('roomList');
    const bookedListRef = firebase.database().ref('All/BookedList');
    bookedListRef.on('child_added', snap => {
        const div = document.createElement('div');
        const ul = document.createElement('ul');
        const button = document.createElement('button');
        ul.innerHTML = snap.key;
        button.innerHTML = 'Delete';
        button.name = snap.key;
        div.id = snap.key + 'div';
        button.onclick = deleteFunction;
        div.appendChild(ul);
        div.appendChild(button);
        bookedList.appendChild(div);
        //details of every student information
        snap.forEach(childsnap => {
            const li = document.createElement('li');
            li.innerHTML = childsnap.key + ": " + childsnap.val();
            ul.appendChild(li);
        });
    });
```

```
    bookedListRef.on('child_changed', snap => {
        const div = document.createElement('div');
        const ul = document.createElement('ul');
        const button = document.createElement('button');
        ul.innerHTML = snap.key;
        button.innerHTML = 'Delete';
        button.name = snap.key;
        div.id = snap.key + 'div';
        button.onclick = deleteFunction;
        div.appendChild(ul);
        div.appendChild(button);
        bookedList.appendChild(div);
        //details of every student information
        snap.forEach(childsnap => {
            const li = document.createElement('li');
            li.innerHTML = childsnap.key + ": " + childsnap.val();
            ul.appendChild(li);
        });
    })
```

```
    //bookedlist remove part
    bookedListRef.on('child_removed', snap => {
        const removedElement = document.getElementById(snap.key + 'div');
        removedElement.remove();
    });
}());
```

These three snippets are the same functionalities as before to load data from firebase, this part is to load the booked information, like which room which building who booked and his/ her details, to help administrator to easier to manage.

```
function deleteFunction() {
    name = this.name;
    if(confirm('Are you sure to cancel this appointment?')) {
        const ref = firebase.database().ref('All/BookedList/' + name);
        let buildingName = '';
        let roomKey = '';
        ref.once('value').then(snap => {
            snap.forEach(childsnap => {
                if(childsnap.key == 'buildingName') {
                    buildingName = childsnap.val();
                } else if(childsnap.key == 'roomKey') {
                    roomKey = childsnap.val();
                }
            });
        }).then( () => {
            const newRef = firebase.database().ref('All/Location/' + buildingName + '/' + roomKey);
            newRef.once('value').then(snap => {
                snap.forEach(childsnap => {
                    if(childsnap.key == 'availability') {
                        let availability = childsnap.val();
                        const after = availability + 1;
                        let updateData = {
                            availability: after
                        };
                        let path = {};
                        path['All/Location' + '/' + buildingName +'/'+ roomKey +'/availability'] = after;
                        firebase.database().ref().update(path);
                        ref.remove();
                    }
                });
            });
        });
    }
}
```

This delete functionality is for administrator to manage the data in booked list. It can delete the record of the room. Before you cancel this record will alert you “are you

sure to delete?” to protect errors. And this part using callback function (.then()). Ours idea is when delete function works and then the availability of that room will plus one to guarantee this room can recovery.

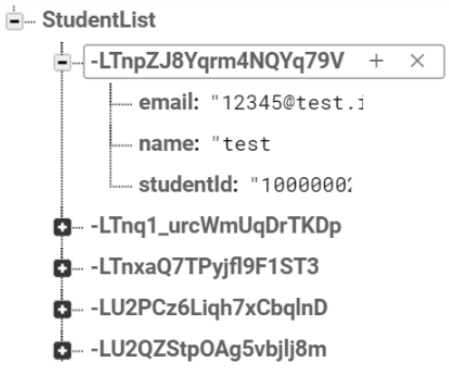
Login Page:

```
const txtEmail = document.getElementById('txtEmail');
const txtPassword = document.getElementById('txtPassword');
const btnLogin = document.getElementById('btnLogin');
const btnSignUp = document.getElementById('btnSignUp');

//Login event
btnLogin.addEventListener('click', e => {
    var email = txtEmail.value;
    var pass = txtPassword.value;
    var auth = firebase.auth();

    //Sign in
    const promise = auth.signInWithEmailAndPassword(email, pass);
    promise.catch(e => console.log(e.massage));
})

//SignUp event
```

```
btnSignUp.addEventListener('click', e => {
    var auth = firebase.auth();
    var emailVal = txtEmail.value;
    var passVal = txtPassword.value;

    var studentId = txtStudentId.value;
    var nameVal = txtName.value;
    //signUp(emailVal, studentId, nameVal);

    const checkRef = firebase.database().ref("/All/StudentList");
    checkRef.orderByChild("studentId").equalTo(studentId).once("value", snap => {
        snap.forEach(childSnap => {
            const email = childSnap.child("email").val();
            if(emailVal == email) {
                alert("You have signed up!");
                return;
            } else {
                const promise = auth.createUserWithEmailAndPassword(emailVal, passVal);
                promise.catch(e => console.log(e.massage));
                const dbRef = firebase.database().ref('All/StudentList');
                const value = {
                    studentId : {
                        email : emailVal,
                        name : nameVal,
                    }
                }
                dbRef.push().set(value, error => {
                    if(error) {
                        console.log('Sign up failed, please try again');
                    } else {
                        console.log("Sign up successful!");
                        window.location = "index.html";
                    }
                });
```

These two snippets are to allow the user login and sign up. The signing up functionality is to check email was already used or no, if not then to write in the firebase. We are using push function to record the information of user, because push function will generate a unique value and it as the key to represent this user

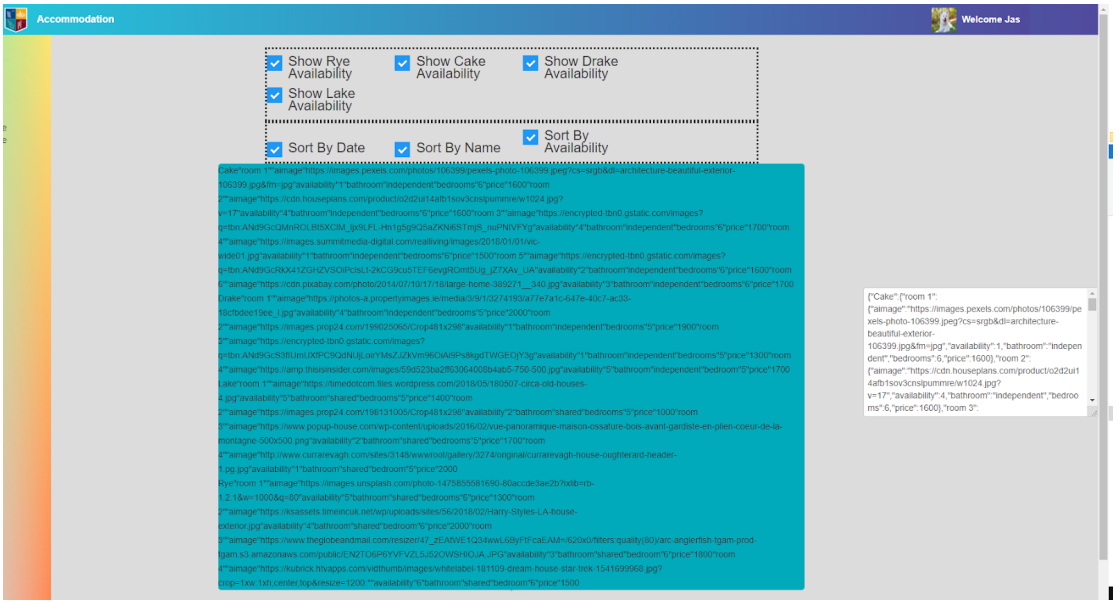
.



The Front End of the project:

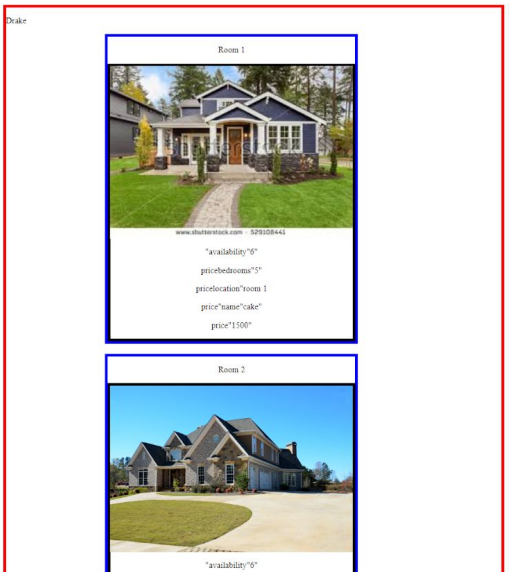
The Student accomodation website had a very basic template at the start, this was then updated heavily by the use of bootstrap.

Below shows the first version of our project, as data is pulled from our firebase and displayed on screen.



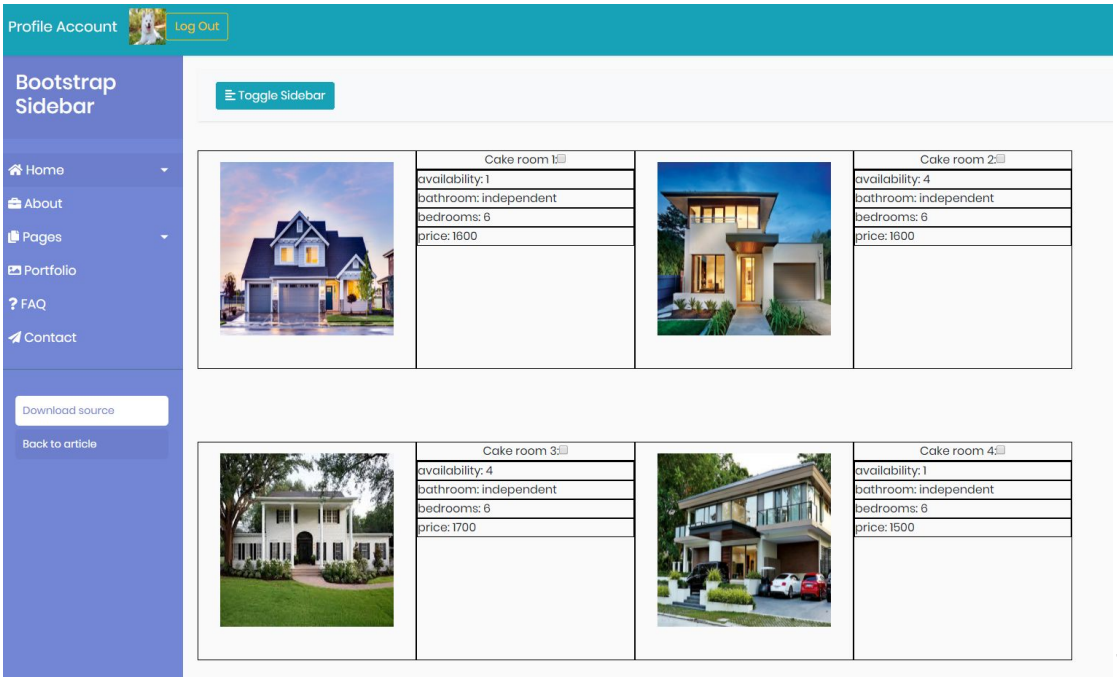
Later we produced better templates and included detailed boxes to contain every detail on each room that could be booked.

Below shows the second version of our product, it is more clear to the users about every detail about the room being booked.



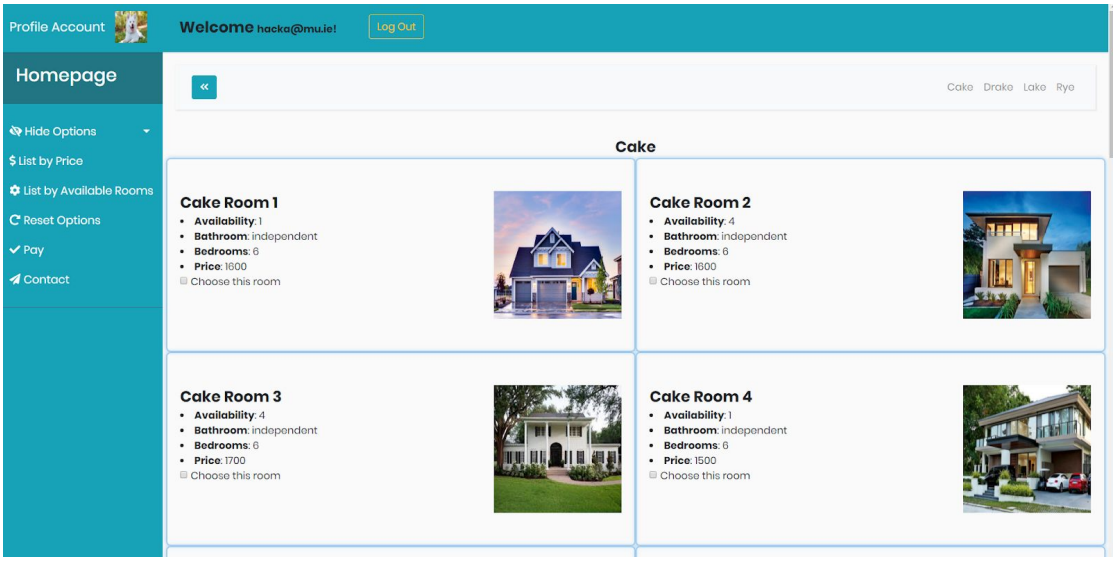
This was all done by simple altering a div tag to include a image slot on the top and a few lines of static text, then the detailed is pulled through. Allowing the image slot to be occupied by the src link and the text appended at the end of the static text.
A better version of this is seen in our third and fourth versions.

3rd Version



This version was a prototype towards the end product, everything had worked perfectly by now. Every bug found was fixed.

4th Version

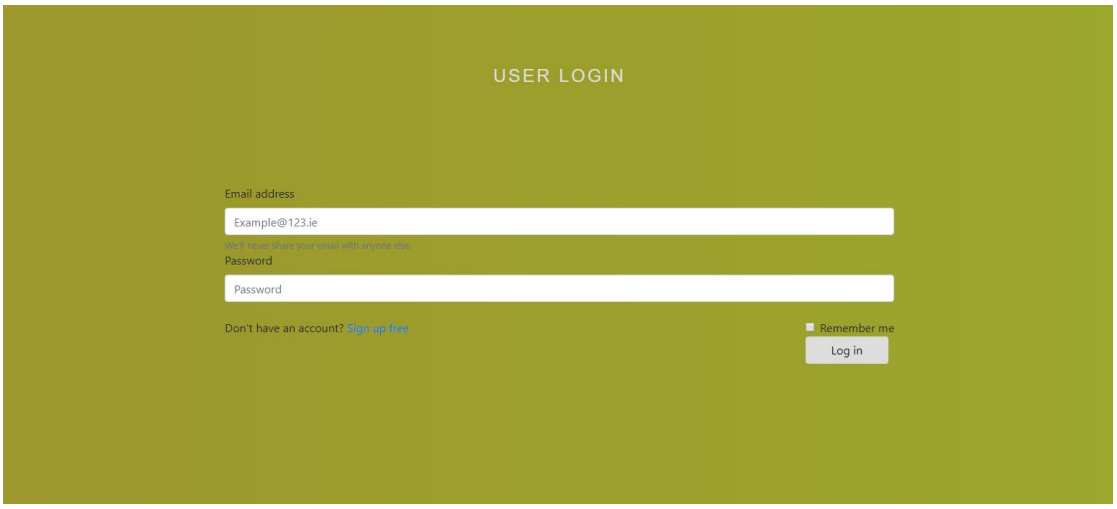


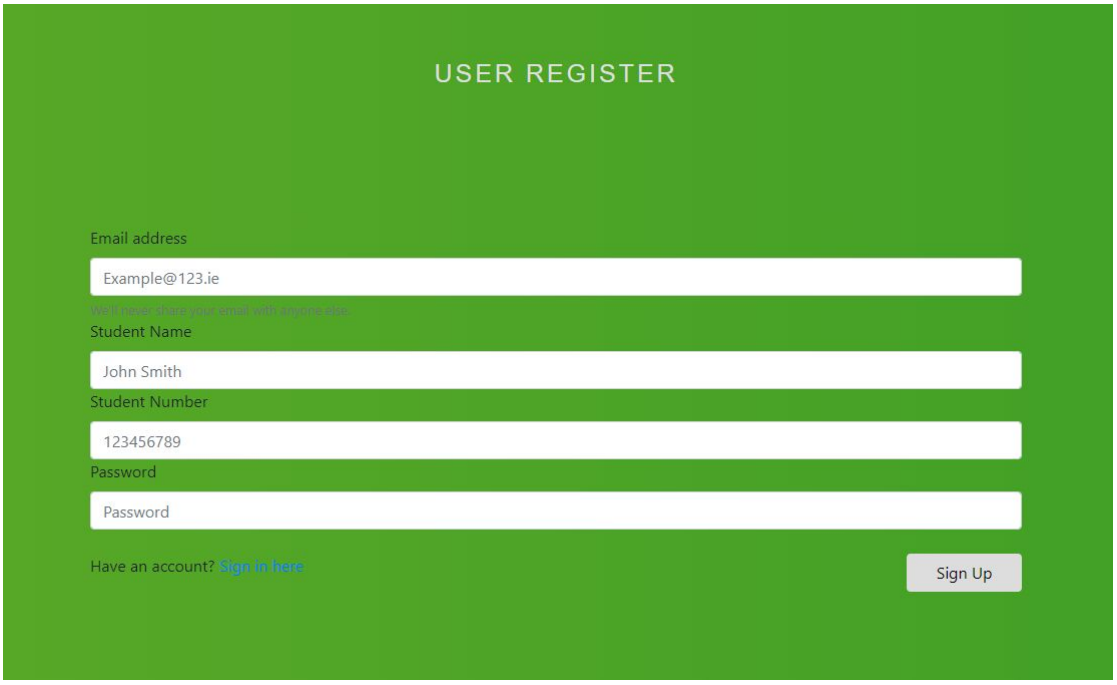
The 4th version was our final product, this would consist of bootstrap for the layout / template of the page and neat CSS and HTML within the boxes. The code to display all of the information within the boxes were all created within node.js.

```
childSnap.forEach(attributes => {
    if(attributes.key == 'aimage') {
        const img = document.createElement('img');
        img.setAttribute("src", attributes.val());
        img.setAttribute("style", "float:right");
        img.style.height = '200px';
        img.style.width = '200px';
        ul.appendChild(img);
    } else if(attributes.key == 'price' || attributes.key == 'availability') {
        const span = document.createElement('span');
        const li = document.createElement('li');
        li.innerHTML = jsUcfirst(attributes.key).bold() + ': ';
        li.appendChild(span);
        li.setAttribute("name", attributes.key);
        ul.appendChild(li);
        span.innerHTML = attributes.val();
    } else {
        const li = document.createElement('li');
        ul.appendChild(li);
        li.setAttribute("name", attributes.key);
        li.innerHTML = jsUcfirst(attributes.key).bold() + ': ' + attributes.val();
    }
});
```

Shown above is the node.js that retrieves the data from Firebase and includes a div box and appends it into a container within our HTML page. All of this is done dynamically and changes with the change of data.

Our project also consisted of a simple login / registration page where the user can easily provide their details into the text area and log in or for registration.



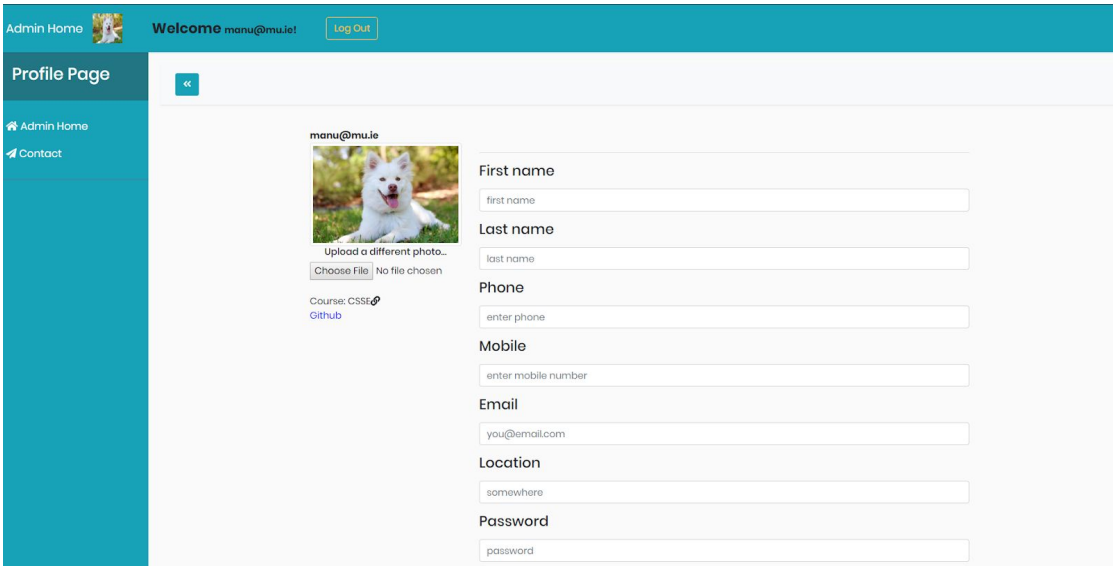


Bootstrap was really important in our project, as we imported it and used it for many things. We would import bootstrap into all of our html pages and call it from there.

Most of the input boxes and buttons were made from bootstrap. We implemented a colour changing background by using javascript and it's webkit library.

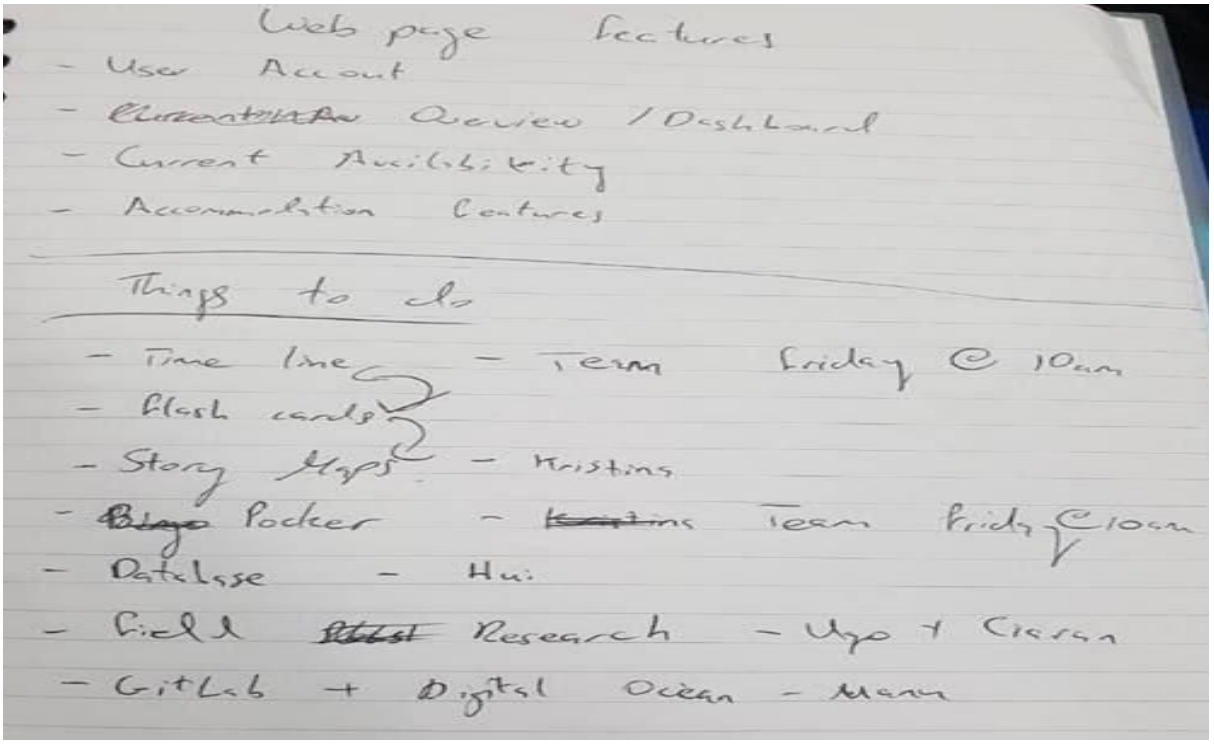
```
#colour {
background: linear-gradient(270deg, #07b0aa, #379d26, #99bd2a, #9f892f);
background-size: 800% 800%;

-webkit-animation: AnimationName 30s ease infinite;
-moz-animation: AnimationName 30s ease infinite;
animation: AnimationName 30s ease infinite;
}

@-webkit-keyframes AnimationName {
    0%{background-position:0% 50%}
    50%{background-position:100% 50%}
    100%{background-position:0% 50%}
}
@-moz-keyframes AnimationName {
    0%{background-position:0% 50%}
    50%{background-position:100% 50%}
    100%{background-position:0% 50%}
}
@keyframes AnimationName {
    0%{background-position:0% 50%}
    50%{background-position:100% 50%}
    100%{background-position:0% 50%}
}
```

Another page of our project is the profile page. This was made so that the user can change their profile information by filling our the boxes and submitting the changes.



Chapter 3:

During our first Scrum meeting, we delegated tasks to everyone that was part of the group to work on for the coming week. Everyone had an idea that they presented to the group. There were many brilliant ideas that if we had more time to do, we would have done but in the end the idea that was chosen was to make a platform where student could find a place to stay while attending college. This idea was one that the group believe that could be improved on and each group member believed that we could finish this in the time that we had to complete it. This idea was something that most of the students in the college would use and we believed that the service that is in place now had a few flaws that could be resolved. The below picture shows what was discussed in the first meeting and the tasks delegated to each member shown:



So, we then proceeded with our Story map creation. Flashcards were drawn up for different Admins and for different student that would use our service. During the same week that the flashcards were created, we held another team meeting which was used to see how everyone was getting on and to see if anyone had run into any problems. Team meetings were of vital importance to the group as we could determine what was remaining and made sure that members were not working on the same tasks. There were a few problems that occurred, and these were dealt with correctly and efficiently. We then set a date for our first sprint. We decided that it

would start on the second of October and end on the twenty-fourth of October. During this period, we held a few meetings which were used to keep track of progress during the sprint. We, also, brainstormed out solutions to problems that were encounter which helped greatly as it cut the time it took to complete each task that had to be done. The first sprint was a good indication of how much we could get done in a period. It helped us organise the project so that we could delegate tasks to everyone in the group and have each person complete a task in a given period of time. Every person would then go and complete the task and return to the group with a completed task. The first sprint was the most important because it was the beginning of the project which also brought the beginning of the development process that we went through.

All in all, the initial Scrum meeting was of vital importance to the group work and our outlook on the project. We got to understand the tasks that was at hand better and got to understand the time that we had to complete each part of the project as well as the effort that was needed to complete it to the best of our ability. This process was a vital stepping stone to the further progression that was to come and the sear work that was involved to get the project to where we wanted it to be.

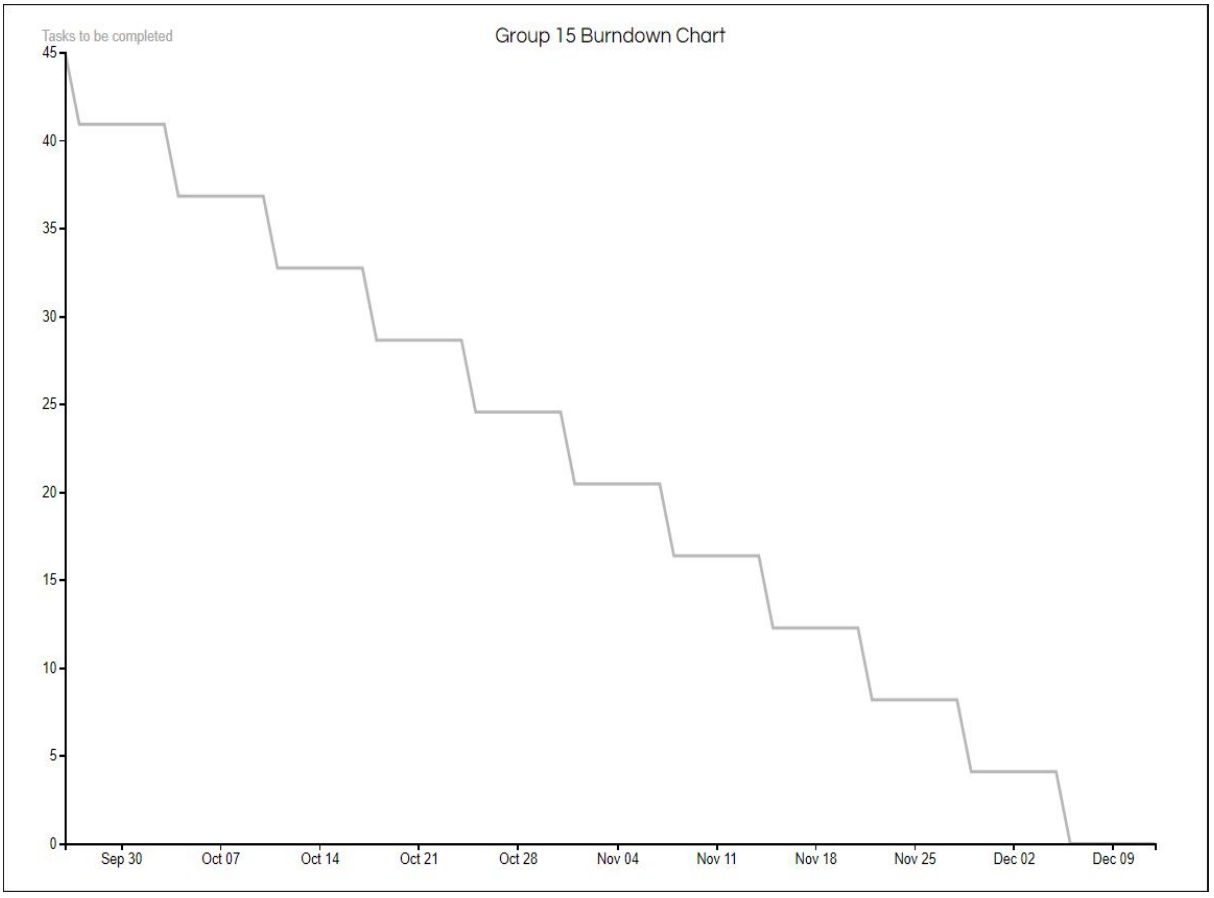
We used planning poker to get an estimation of what could be done during the first sprint. We went onto create separate periods using the planning poker of what had to be done at certain times of the project. This broke the project up into different components that had to be done. It also gave us a better outlook on how the project would look at different stages during the semester. It was of use to us when we compare what we had to what should be done to indicate to us if we were on track or not. We had an understanding of what part of the database and what research had to be done by each of the given times decided by the group. It kept us on track to finish the project in the given time. It was a collection of what each person was going to be doing over the coming week and paired with the group meetings each week, we had a guideline of what was being done. It took some time to finish the planning poker but, in the end, it saved us a lot of time when it came time to actually do the project. Therefore, we had more time to invest in the project and our project benefitted greatly from these extra hours we had to play with. As a result, we came out with the project being almost as we hoped it would be. So, it was definitely worthwhile spending time doing the planning poker and tracking the development of our project this way. It was a great investment of time on our part and we definitely came out with more benefits than anything else because of this.

Our group made the most of the minutes that were available for us to use and we held many meetings where we came together and talked about how different parts of the project were coming along but also to work together to really nail down certain parts. Everyone in our year had to meet up as usual on a Wednesday but we also made effort to meet up on a Friday to really get cracking on what needed to be done and to see if there were any problems that anyone had encountered. There were a few each time we met but it was useful to have everyone there to help out, so we could get over the obstacle that was place in front of us. These meetings on Fridays would consist of us working on code and research in the beginning. We split up into two groups, one working on the coding side of things and the other working on getting as much research as they possibly could as we needed that as a basis for what was to come.

Meeting up twice in the week was very beneficial to the work being completed by the group. Meeting up on Wednesday, we could see how people had got on with the work from Friday that they were doing and then, meeting on Friday, we could see how people had started the work they had decided that they were going to be looking at and then see if they needed something from another member of the group and then, if they did, have them link in with each other and get that section of the project completed. The meetings were a great asset to the team as we got a better understand of the project and what section were being done. There ended up being less confusion about what parts of the project were already being looked at by another member of the group and ended up indicating what parts still had to be looked at and completed. Again, we ended up being more informed as a whole and coming away with a better knowledge of where we stood as a group on our way to completion. The time injected into the meeting times greatly paid off as we knew what each of us were looking at and then had a decision of what to look at next and complete over the next week.

The Burndown Chart was a good indication of what had to be completed as the weeks progressed and how many tasks the group had to work through to stay on track for the project completion date. It worked out to be an estimate of about four or five tasks each week. So that worked out to be a task per each member which was a great benefit for the group. This, again, was another indicator to the group of when we were slacking and not performing to the best of our abilities. This was a great benefit to the moral of the group as it showed that with almost an ounce of work, we could get this project completed in time for when it needed to be presented. It would have been a real burden to have to think out how we were getting on and be worrying about being behind. But with this chart that was taken out of the equation completely. By looking at the Burndown Chart below you can see that pair with the end of our first sprint, we were around about the halfway point of completion when we came to the end of the sprint. This was great as that left us with a good chunk of

time remaining to finish the parts that remained. The Burndown Chart brought more benefits to the group than it did anything else. We had a great understand of how each person was performing and it took a lot of the minds of the members of the group which was great as we performed better as a whole as a result. We feel that this Burndown chart was important to the project even if it was not always used but it was when we needed to see if the group was were it needed to be.



The story maps were a great starting point for the aim of what the project was doing. We started off by doing out four flashcards. Two for students and two for Admins. The students were: 1) A prospect Student, 2) A Current Full-time Student. For the flashcard of the prospect student, we put down what the student would have to do in order to use the site. For example, register -> login -> browse accommodation ->

reserve preferred accommodation. This then gave us an indication of what was needed for that student to use the site and what needed to be put in place for this to happen. Then, for the current Full-time Student, we also wrote down what that student was required to do. But seeing as this student was already register in the college system, they could login using their details that they have been given. This, then, gave us two perspective of who would be using the site to find the accommodation and gave us an idea of what needed to be implemented to get the best out of the site. Then we moved onto the two Admins which included: 1) Site Admin, 2) University Admin. The Site Admin would be in charge of maintaining the accommodation availability service that had been put in place and resolving and issues that could arise as soon as possible. The University Admin would be responsible for the updating or removal of any accommodation that had either become available or been reserved. They would also keep the status of each accommodation up-to-date for the student to see.

From there we went on to do out some flashcards for the user account. These accounts would need some information, including name and gender, in order for this account to be created. This information inputted when the user is creating their account will be taken and used on their account profile page. We also created flashcards for the accommodation catalogue which is where users would browse through the available accommodations. We broke this section up into different sections which could be used to filter the users needs. These were: Location (Where exactly it is on campus), Current Availability (What is not reserved yet), Price (Cost for renting the room), Type (Disability friendly, etc.) and Number of Rooms (Selection of how many rooms there are in the accommodation). The Story maps were a great indication of what was expected from our project and what needed to be implemented in order to have the project running to the best of its ability.

The testing phase was the most important part of the project as we finally got to see all of our work come together, link in with each other and see if it ran correctly. This phase was the one that was most anticipated by our member as we got a glimpse of what users would see when they open and use the site. With testing we were able to identify any bugs that were encountered and make amends to the coding. Together with that we had many different versions of code that were updated regularly to add in different aspects of the design that needed to be done. Each version brought with it a new aspect that greatly improved our project. These versions went as far as version 14 which was the finished project. Having new version was a great way of knowing what worked in previous versions and if the new version did not run then we knew that there must be a problem with the new additions made. The testing of each

new version was of paramount importance as we needed to know if the software needed improvements or if it was up to our desired standards.

With there being fourteen different version came a lot of change and some mistakes, but the team was on had to resolve these issues. Two members of the group were working on coding from the very start and their hard work definitely paid off greatly as the group was very pleased with the outcome. They put in a ton of work and every time we met for a meeting, they had some new improvements to show the group. Testing was done on a regular basis and it was then easier to stay on top of mistakes that had to be resolved. Through testing and versioning, we were able to have a finished project to present to the class and because of this it was working as we had hoped that it would. Without testing, we would not have known were we stood and how well the code worked. So, having fourteen versions was worth it in the long run as it showed us how we were improving the project constantly and it then had shown us how many tests had already been performed on the code to that version.

We also looked at the project and thought about how the project would shape up for future releases. We could have done a lot more with the project if we had a lot more time. The project was great for the time given and we were pleased with the outcome. If more time was given, we could have added in a bunch more features to the coding, but time restrictions prevented that. Each member of the group gave ideas that had to be shelved as there was not enough time to accomplish them as they were difficult and would have taken a considerable amount of time. Many aspects that were dropped would have made drastic changes to how the project function but with this then comes the possibilities if have issues. If more time was available to progress, we could have made a considerable amount of progress to our project and it would have been exactly where we wanted it to be. Everyone in the group had at least one idea that they wanted to implement into the project, but they had to be shot down. So even with at least four ideas it would have made a huge difference to the project. But we were happy with the outcome with the time that we had, and the project ended up in a great place.

Group management and communication had its patches but overall it was good. At some points there would be some miscommunication, but this would then be resolved, and everything was sorted. The management of the group was on point as we all took turn being in charge of the group each week and this then took the weight off just one person being in charge. This gave everyone a taste of what it is like to be in charge of a group and provided them with that experience for the future. The group came together to get the project done and this was attributed to good team work. But as stated above there were times were the communication was not where

it should have been. This led to there being a considerable amount of confusion between member of the group and wasted a bit of time. This was cause by many different elements and really wasted the time that the group had to complete the project. But it was resolved in the end and we ended up where we wanted to be. The meetings again had a huge impact on the communication between the group as we got to discuss in person what each person was doing. We also used an app called telegram to communicate with each other and it was here where people were able to ask each other questions and receive updates. We were able to use this app to send other group members links to the files or documentation that we were working on. This app greatly helped with the communication aspect of the project and was very beneficial to our group. When we split up at the start the management and communication was perfect and we got all of the basics done very easily because of this. Each individual had to play the role of Scrum master also and each of us done a great job being the leader for that meeting.

As mentioned above, the communication and management could have been better at time but in the end, it was to a good enough standard to have a finished project and allow us to present it to our peers. A lot of different factor caused the issue but, in the end, we done what we set out to do and we were happy with the outcome. These issues were no one’s fault as we are all human and we resolved the problems. We must let bygones be bygones and remember that it is not what you have but it depends on how you present it.

Chapter 4:

My contribution report is just a run down of my weekly diary with more details into my individual progress.

First few weeks:

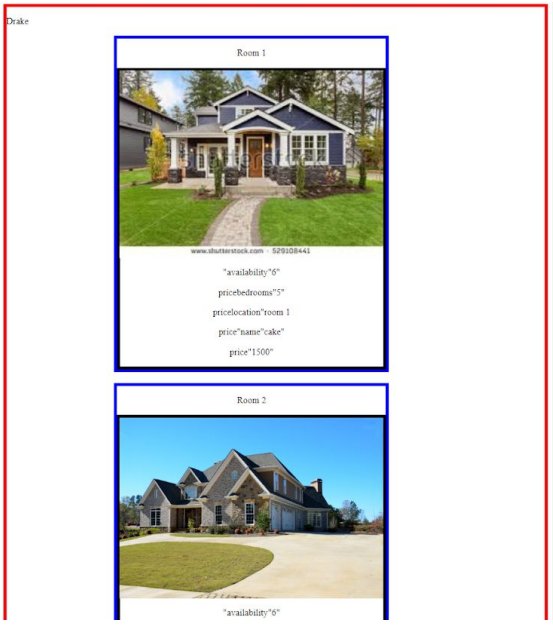
- I adapted to the new project plan (Student accommodation site) and made progress towards the backlog.
- After the backlog, I looked at the environments we could use, I did a lot of research on Firebase vs rethinkdb.
- I really liked Firebase and started setting up the database on my account. I also set up the GitLab and DigitalOcean for the team, and a Telegram account on my mobile phone.
- It was my first time using Firebase and I got along with it easily with the help of it's excellent doc system online.
- I made a template code to allow all my teammates to access Firebase via node.js, I have used node.js in the past and it was fun to use again.
- The backlog was later fully updated and we began our progress to actually making the site.
- I asked the team to try find some example data for the student rooms for the future in our project, this was a crucial step as I researched important features within other website that contained buildings.

Middle weeks:

- I started to make progress to our database. I included some example data that we had created over the week. When this data was inserted, I started to try grab this data with node.js into html and show the data onto the site.
- The rest of my team was tasked to make templates by the end of the week so that Hui and I could had worked with the main page and login page as soon as possible.
- I extensively researched on pulling data from Firebase into html, as this is one of the most important steps in our project. HTML and CSS would be fine for me, as I have taken a course and internship that included them.
- Later on, I grabbed all the data from the database and also included pictures into our data from a src link. I would then attach a src link to every data in Firebase.
- I started to test the database for bugs and anything that could be implemented better.
- I started to format the information gathered into boxes so it fit into a more appealing appearance.

(I also made a weaker template for the mainpage)

Neatly fit Room information into boxes



Login/Reg pages

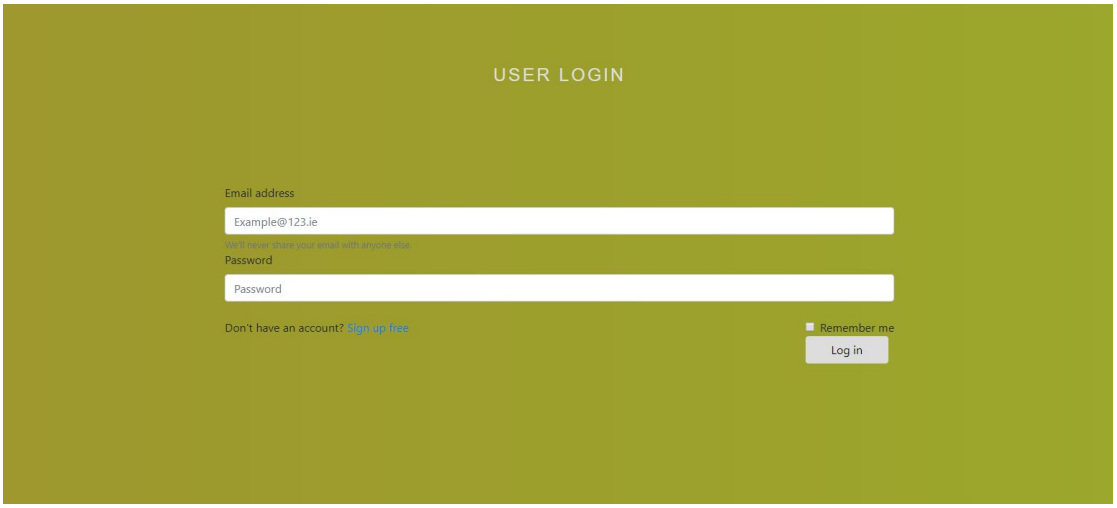
- The team had produced very weak templates for me to use. I decided to use bootstrap and my own implementation but asked them to detail the templates more.
- I created a login page and registration on the same page for those weeks.
- Following the backlog and as the leader of the team, I asked Hui to work with the functions for the mainpage. Including sorting / hiding / payment / admin page.

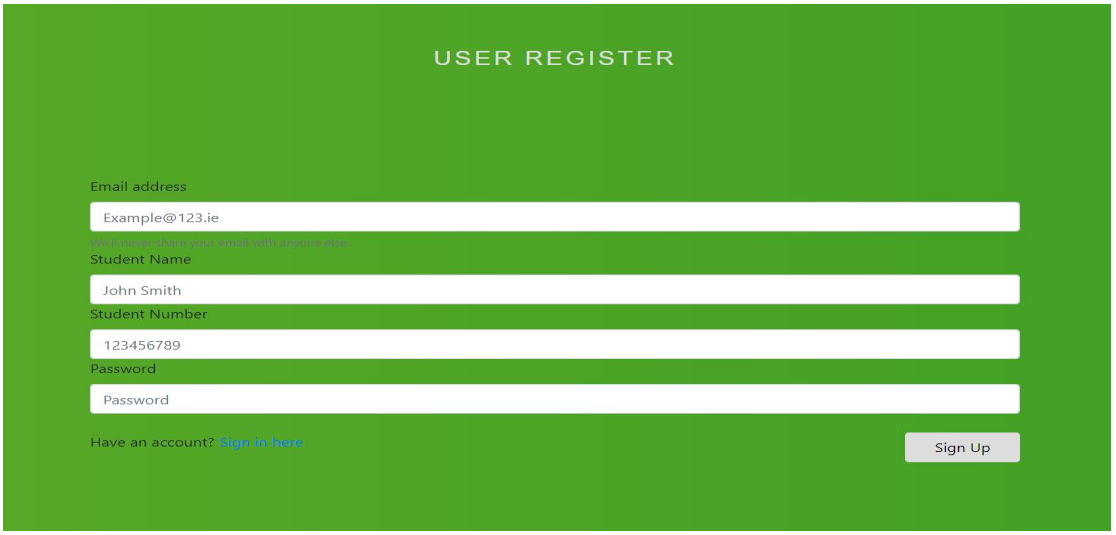
- During this week we had a team meeting on Friday and I asked a Ciaran / Ugo to try work on a profile page for the user to access when logged on. (Note: This was never made)

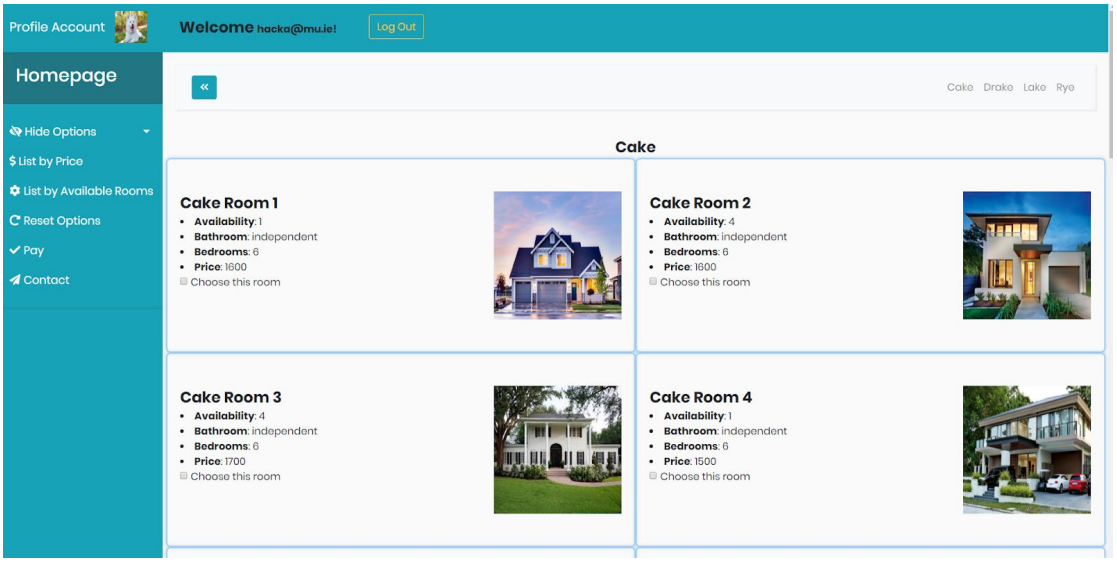
(I made a weak login page with functionality)

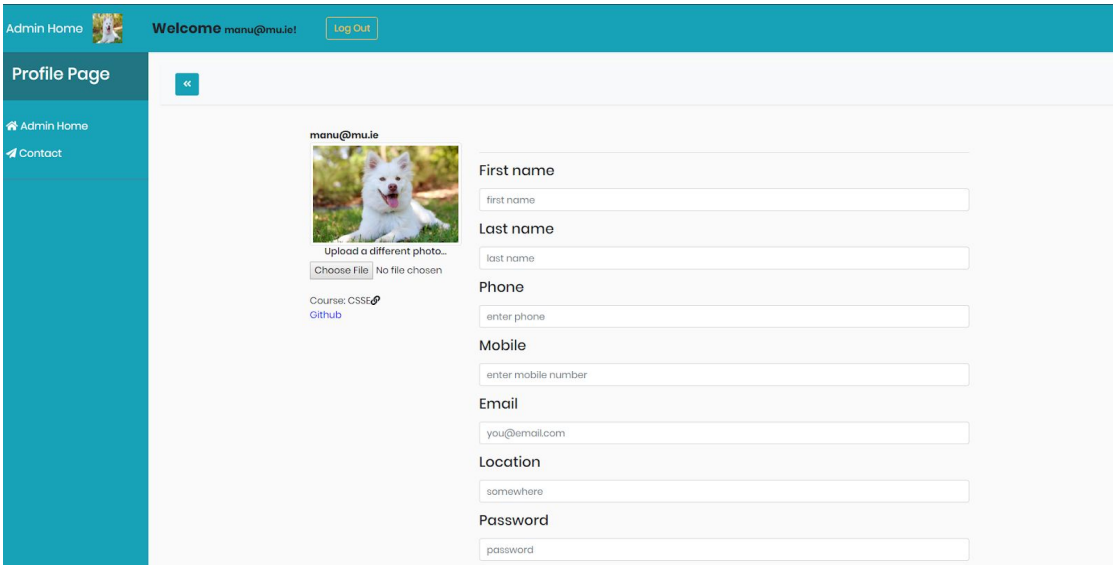
Final weeks:

- I took the Gitlab branches and merged everything together into a working application.
- I then fixed many bugs and determined what else was needed to finish the project.
- I created a profile page myself and made the main page fancier and also created a registration page.
- I created an additional admin page where the admin could modify the data.




USER REGISTER
Email address
Example@123.ie
Student Name
John Smith
Student Number
123456789
Password
Password
Have an account? Sign in here
Sign Up


Profile Account
Welcome hacka@mu.ie!
Log Out
Homepage
Hide Options
List by Price
List by Available Rooms
Reset Options
Pay
Contact
Cake Drake Lake Rye
Cake
Cake Room 1
Availability: 1
Bathroom: independent
Bedrooms: 6
Price: 1600
Choose this room
Cake Room 2
Availability: 4
Bathroom: independent
Bedrooms: 6
Price: 1600
Choose this room
Cake Room 3
Availability: 4
Bathroom: independent
Bedrooms: 6
Price: 1700
Choose this room
Cake Room 4
Availability: 1
Bathroom: independent
Bedrooms: 6
Price: 1500
Choose this room


Admin Home
Welcome manu@mu.ie!
Log Out
Profile Page
Admin Home
Contact
manu@mu.ie
Upload a different photo...
Choose File
No file chosen
Course: CSSE
Github
First name
first name
Last name
last name
Phone
enter phone
Mobile
enter mobile number
Email
you@email.com
Location
somewhere
Password
password

I tried to fit everything I have done into the end product, I would had left what wasn't necessary within the previous versions. Most of my research was between firebase, node.js, git, html, css and bootstrap. Node.js and firebase was definitely the harder obstacle to tackle.

Chapter 5:

The idea chosen was actually not one that I liked or wanted, as it was too simple and a replica of what is already out there. Our backlog was 100% achieved by the end of the project and we also included more features. This was a really great achievement and it made me aim for future goals if the project continued.

I had hoped for a group project ever since first year. I found my first year project very enjoyable and an amazing experience where I learned a lot more in Java. Unfortunately, this group project didn't meet my expectations, I did really enjoy working with one of the team members as we helped each other learn and create new elements to our project. I really couldn't depend on most of the team members during the entire 10 weeks. I learned a lot from their actions and spent a lot of extra time to create and produce what they wouldn't do. Although most of the team wasn't actively involved, I am truly happy that we achieved everything we hoped for in the backlog. We even included more functionalities and future updates etc.

I have learned a lot on using Git in a working environment. I researched many hours into firebase and am very familiar with it. I was familiar with the MEAN stack during my internship in 2018 Summer so I felt comfortable with node.js.
I am looking forward to working in a team environment during my internship at Amazon.

I also received some backlash on this group project by a teammate who did not produce a task on time. I was said to be doing everyone's part, I believe that I done everyone's part because they never done theirs on time and that we fell behind a week or two. Regarding this, I am happy that I could help finish the project on time.

Suggestions

See if a team is capable of achieving big together. Penalise members who do not participate. Maybe each team should have a mentor.