**Assignment 2**

**COMP3341 (Advance Web Application Development)**

**Assignment 2**

**Semester 2 - 2016/17**

**Student number = XXXXXX**

**Word Count: 1169 (Excluding Annotations/Tables)**

□ I do not want my work to be used anonymously to help future students

**Assignment 2 - Web application and Final Project Report.**

**Part 2: Final Project Report (1200 word equivalent)**

1. **URL for the ‘live’ site.**

[www.shaunguy.worcestercomputing.com/benefitbooking](http://www.shaunguy.worcestercomputing.com/benefitbooking)

Username: preslaw1

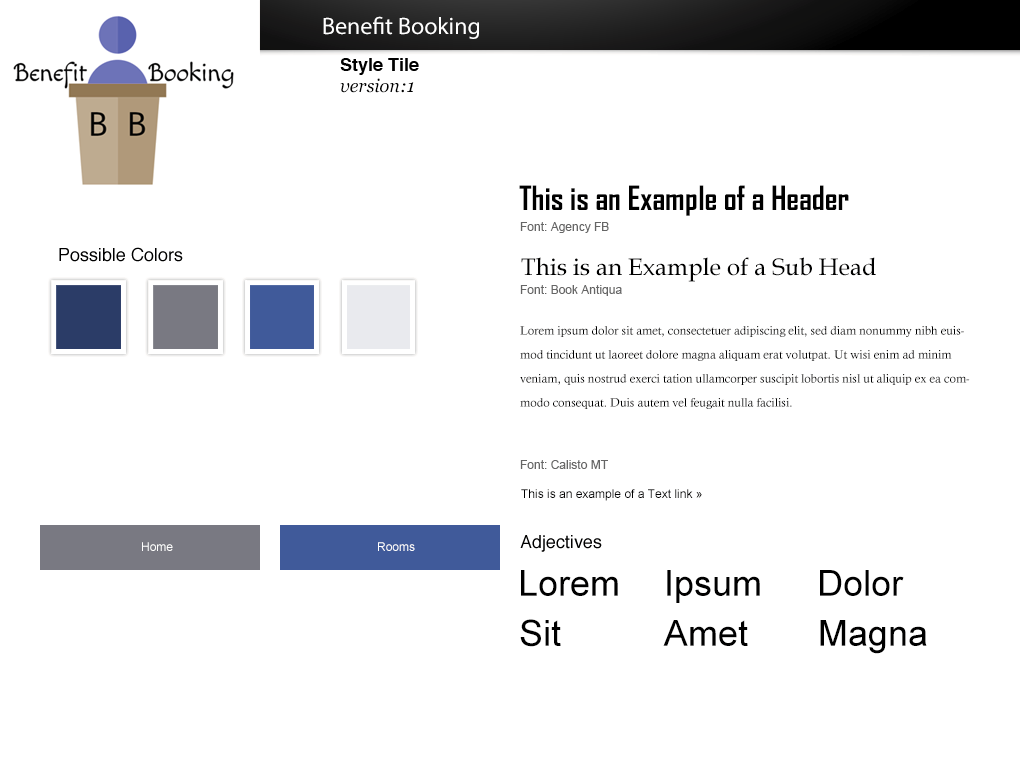
Password: Testing123

Email: [testing123@gmail.com](mailto:testing123@gmail.com) - example email address.

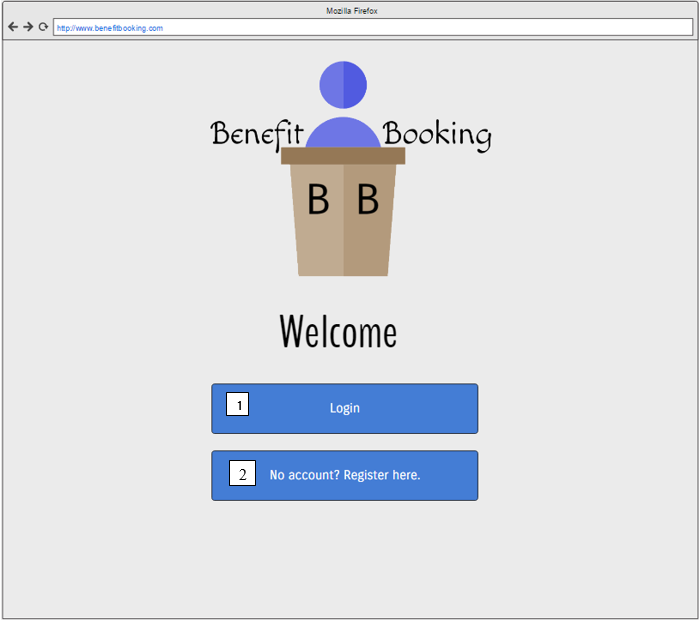
Although an account has been created already, the e-mail address used for the default account will not be linked to an official e-mail. To view a number of the features that the Benefit Booking application offers, please create a new account with an e-mail that can be checked.

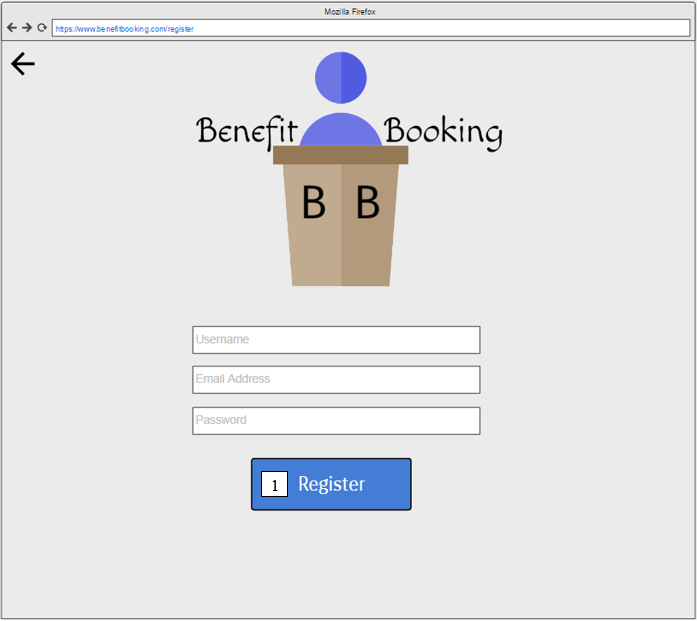
1. **Outline of the web application (160)**The Benefit Booking application aims to provide a comprehensive, minimalistic booking system for conference rooms. The application is designed for use by professionals or small business owners that need a conference room with various amenities such as projectors, WiFi and PCs. The application utilises a registration, login, forgotten password as well as a My Account section where clients can delete their existing booking. All rooms are stored individually in the database and displayed dynamically on the website but require the user to be logged in before booking commences to reduce the need for additional input fields. For simplicity purposes, the project focuses on room booking for 1 day periods rather than specific timeframes such as 14:00 – 17:00. With other popular booking systems online such as Trivago (2017), Benefit Booking uses the niche of offering conference rooms to professionals with a slogan of ‘Professional rooms, competitive pricing’ alongside an array of functionality and end-user considerations to provide a pleasant, user-driven experience.
2. **The development lifecycle of the web application (450)**

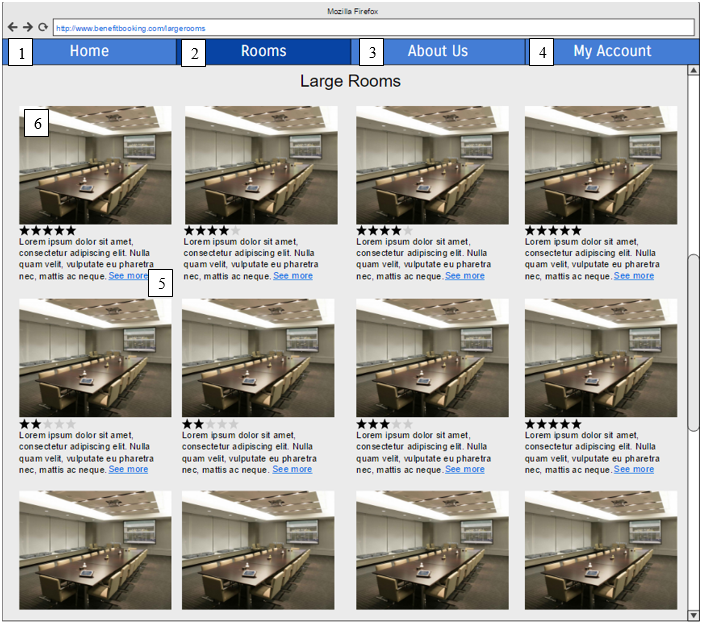
**Initial Design**

Figure 1 – Initial Style Tile

The project specification showcased various style tiles that could have been used throughout development. Of these, the blue/white style tile by Facebook was chosen for its simple but professional image.

Figure 2 – Initial Prototype (Welcome)

Figure 3 – Initial Prototype (Register/Sign Up)

Figure 4 – Initial prototype (Rooms Page)

Initial design appeared very cluttered in practice and was changed later in development.

**Development Changes and Progression**

Figure 5 – Login and Signup on Navigation Bar

Changed the login and signup buttons on navigation bar to keep styles consistent.

Figure 5a – Login and Signup style changes

Figure 6 – Navigation Bar (Development Iteration 1)

Removed logo from Figure 6 🡪 Figure 6a, the logo was too small and was not aesthetically pleasing.

Changed colour of Navbar from grey to blue to make text easier to read.



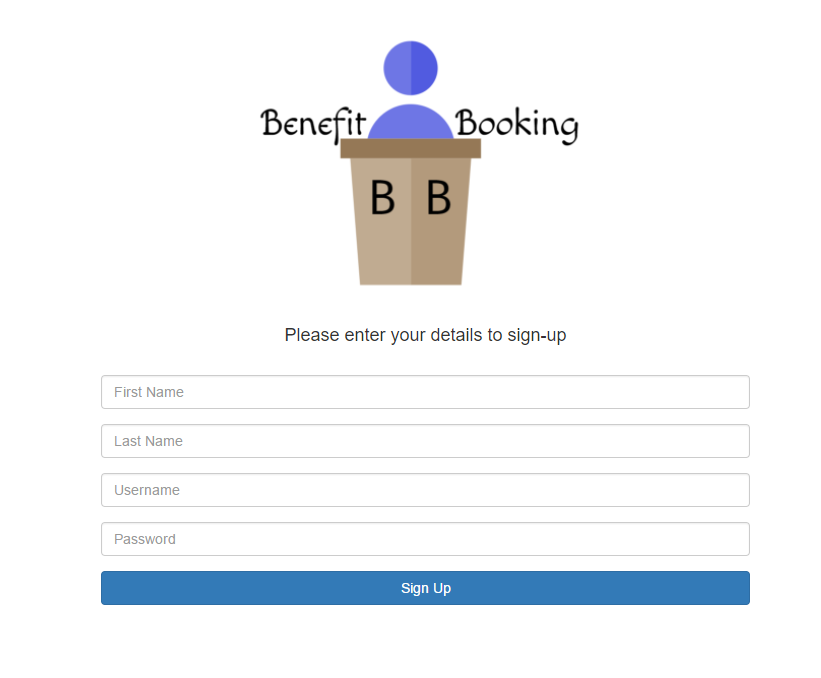
Figure 6a – Navigation Bar (Development Iteration 2)

Separated ‘Login’ into ‘Login’ and ‘Sign Up’ to reduce the number of clicks required.

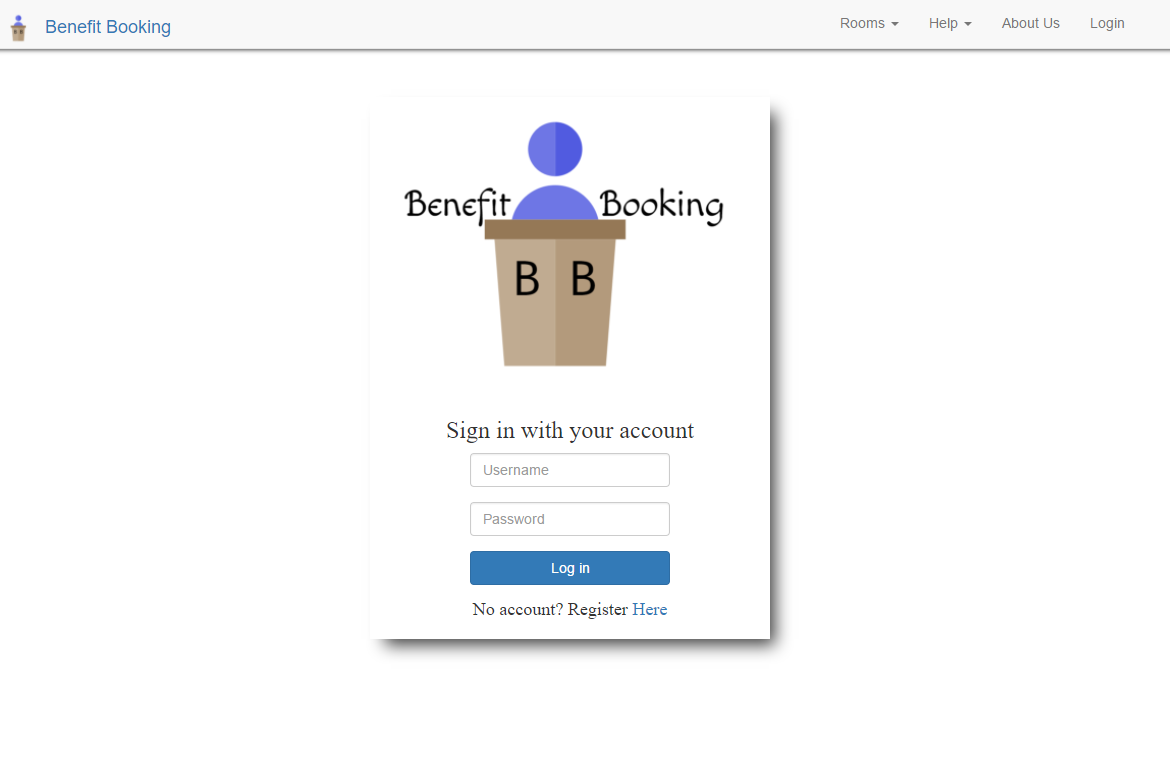


Figure 6b – Navigation Bar (Development Iteration 3)

Regarding the drastic colour change from iteration 1 to iteration 2, (Osborne, 2015)

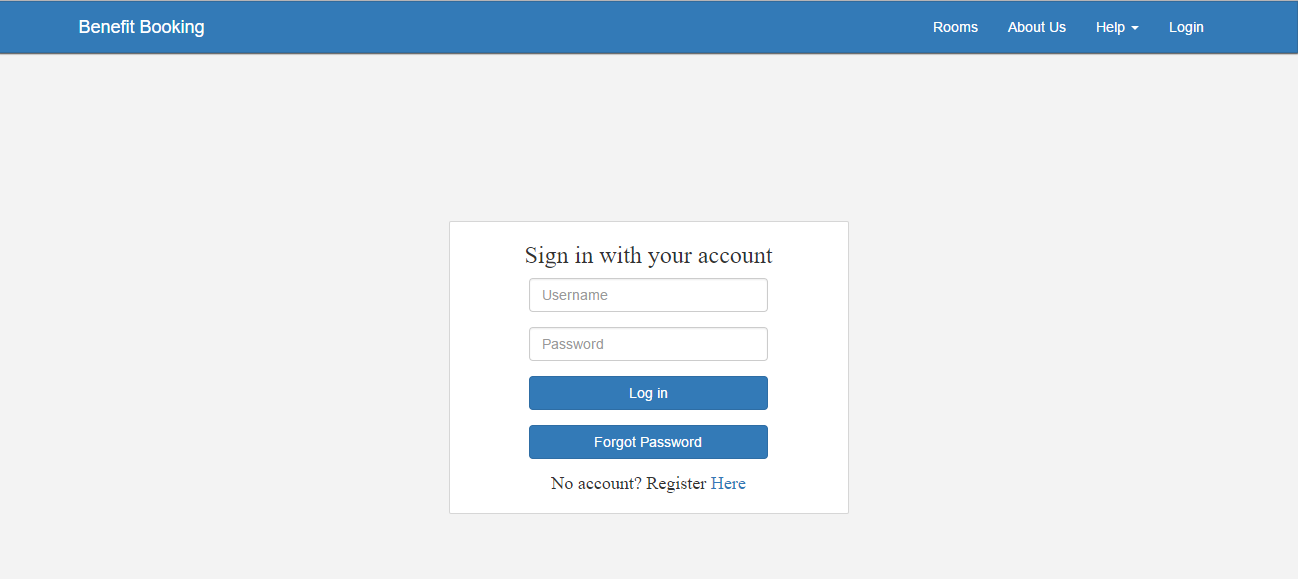
Figure 7 – Sign Up/Login Form (Development Interation 1)

Initial design iteration consisted of a form on a plain background with no user of containers. This was due to limited understanding of Bootstrap at the beginning of development.

Figure 7a – Sign Up/Login Form (Development Iteration 2)

Added hyperlink to redirect to signup form, reduces clicks and prevents users from hitting a ‘dead end’ in the application.

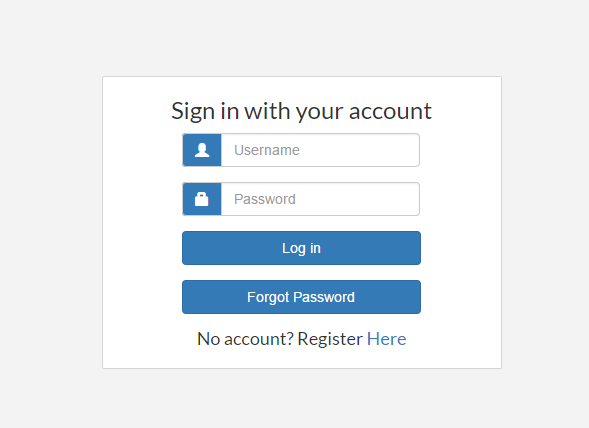
Used a container to store form and logo, this design choice allowed the use of a background without interfering with form contents.

Figure 7b – Sign Up/Login Form (Development Iteration 3)

Removed the logo from container: The logo was often very disruptive when trying to consider mobile users.

Added an option to request a password reset. Functionality decision. (See Figure 10)

Figure 7b shows a revamp to the overall theme of the application. The border and styling of the container was reduced to give a ‘card-style’ affect. Cao (2016) describes the card-style design as ‘flat’, ‘minimalistic’ with many websites utilising the design trend. This design change made a drastic improvement in terms of the appearance of the application and resulted in a clearer, more professional view.

Figure 7c – Sign Up/Login Form (Development Iteration 4)

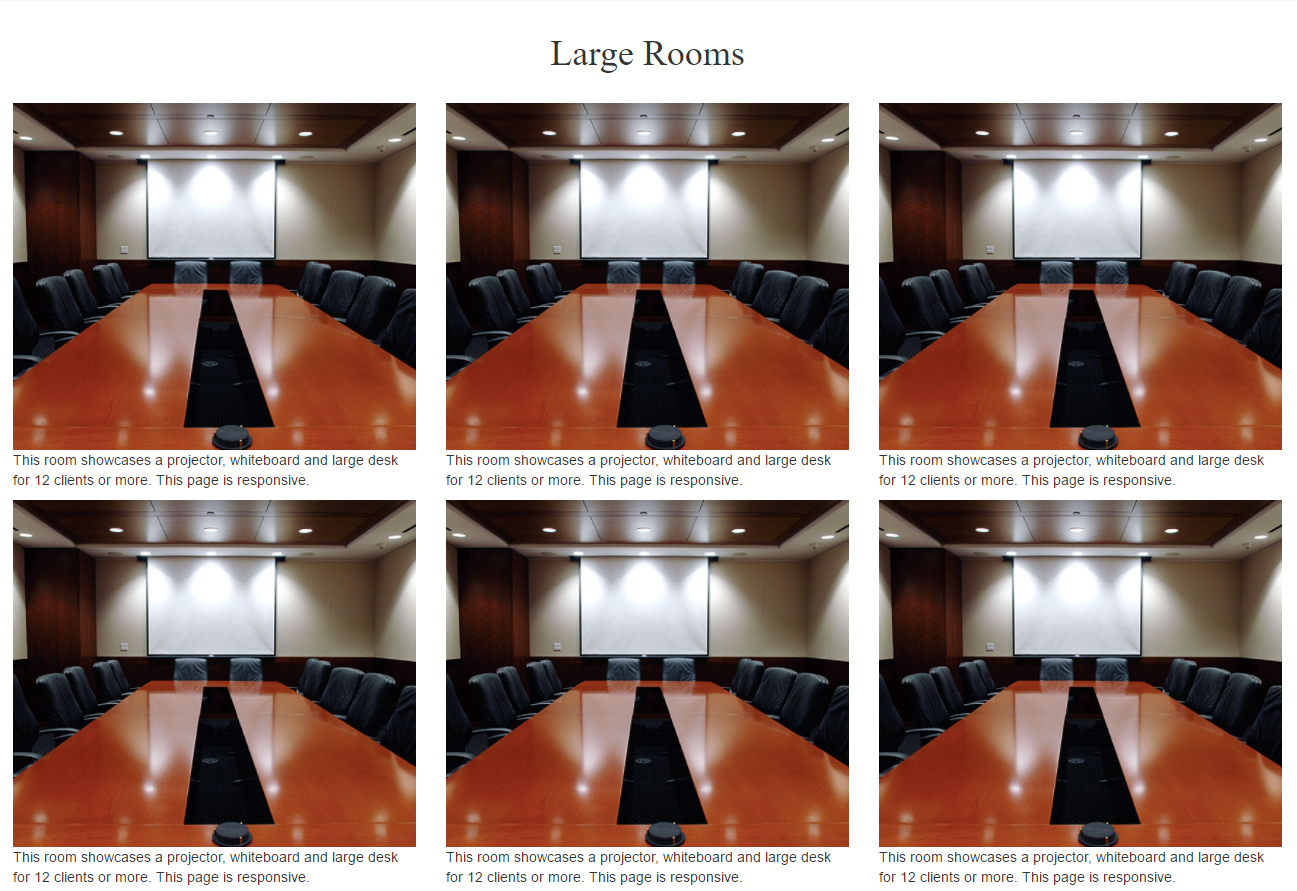
Added blue icons to text fields to further establish colour and design choices.

Changed text to be more clear and concise.

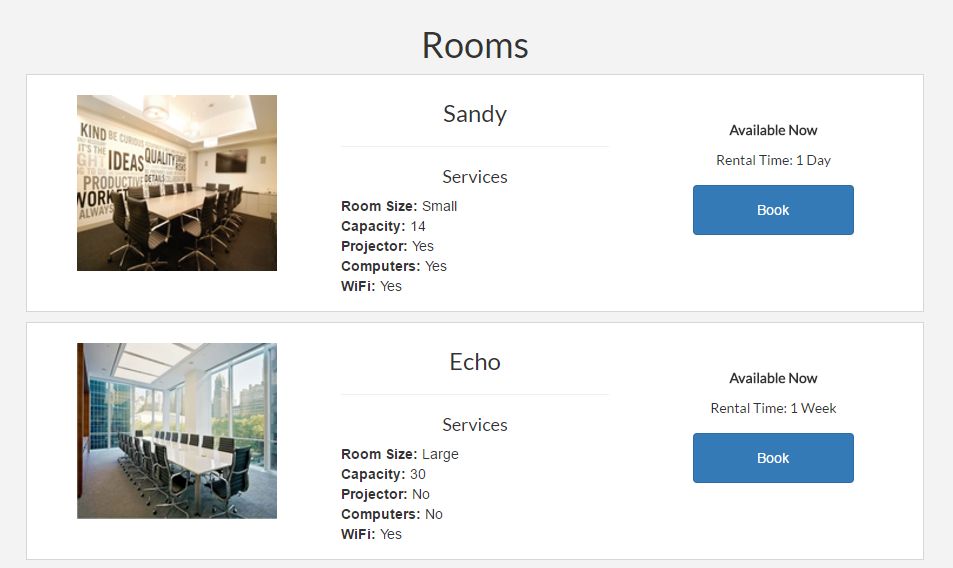
The application utilised bootstrap alerts to notify the user of various errors throughout the application such as invalid form entry, room cancellation, email notifications and more. Egri and Bayrak (2014) mention that if users make mistakes, errors should be informative and intuitive. Additionally, since the demographic of Benefit Booking is working professionals, it is reasonable to assume that some of those professionals may be older individuals. Li et al. (2012) explains that the complexity of tasks has a different affect on a variety of older users. Additionally, Li et al. (2012) also mentions that dividing web form information into sets helps to provide a smaller number of large decisions. With clear, concise validation and errors, the data entry process is made more understandable, particularly for older users.



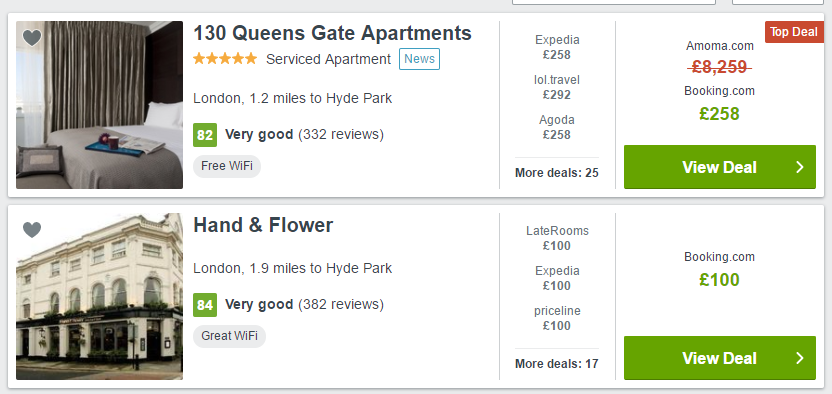
Figure 7d – Sign Up/Login Form (Development Iteration 4)

Figure 8 – Rooms Page (Development Iteration 1)

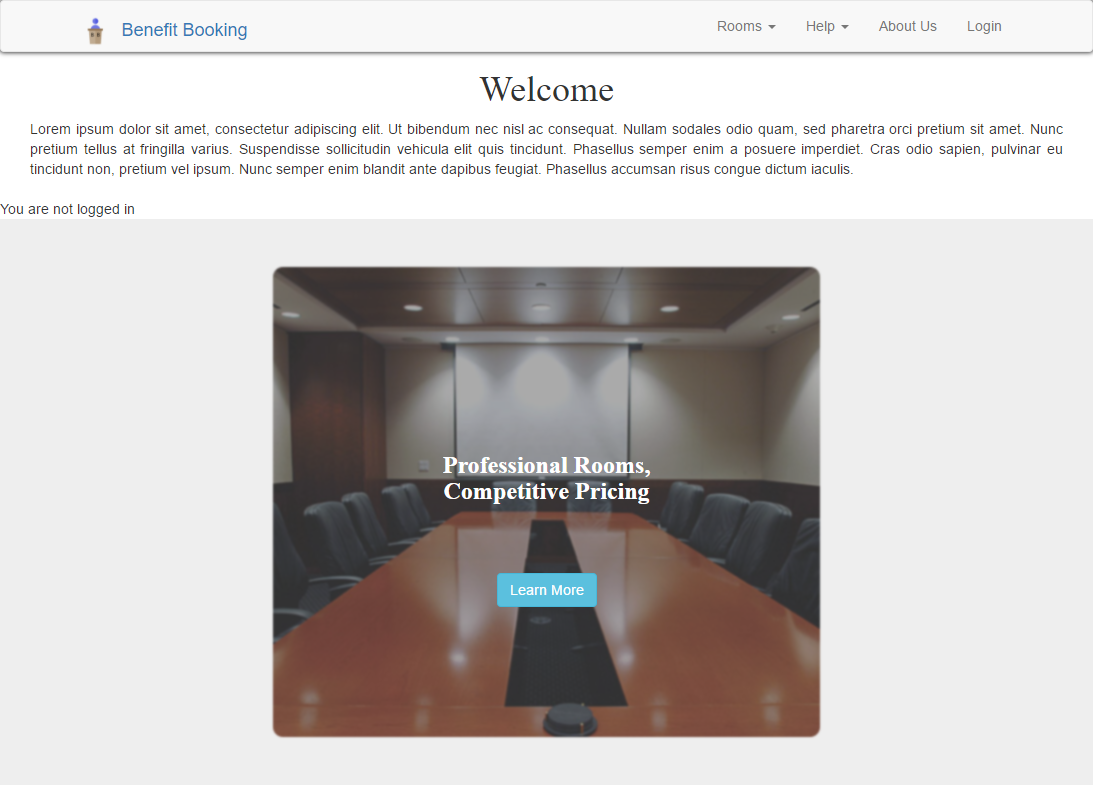
Used Bootstrap rows/columns to adhere to initial design choices regarding ‘Rooms’ section of application. Overall, it was found that this section, while incredibly responsive, looked very cluttered and overwhelming.

Figure 8a – Rooms Page (Development Iteration 2)

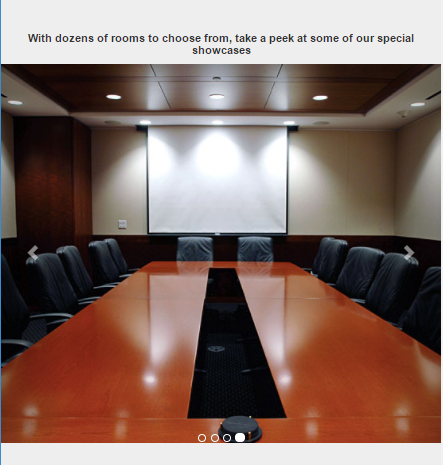
Utilised the container style from Figure 7 to completely re-design the ‘Rooms’ section of the website. (See Figure 8b)

Figure 8b – Trivago

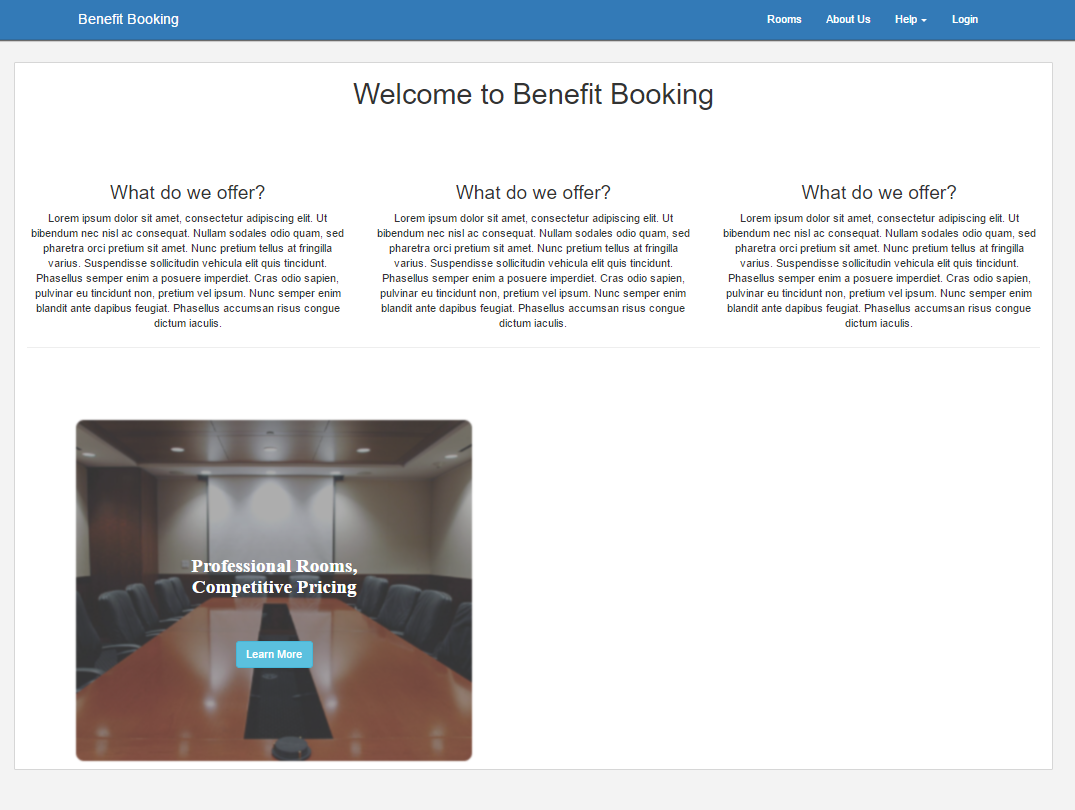
Inspiration for the card-style layout came from Trivago (2017) who arranged and listed records with a strong appeal to space usage and responsive design. (See figure 7b)

Figure 9 – Home Page (Development Iteration 1)

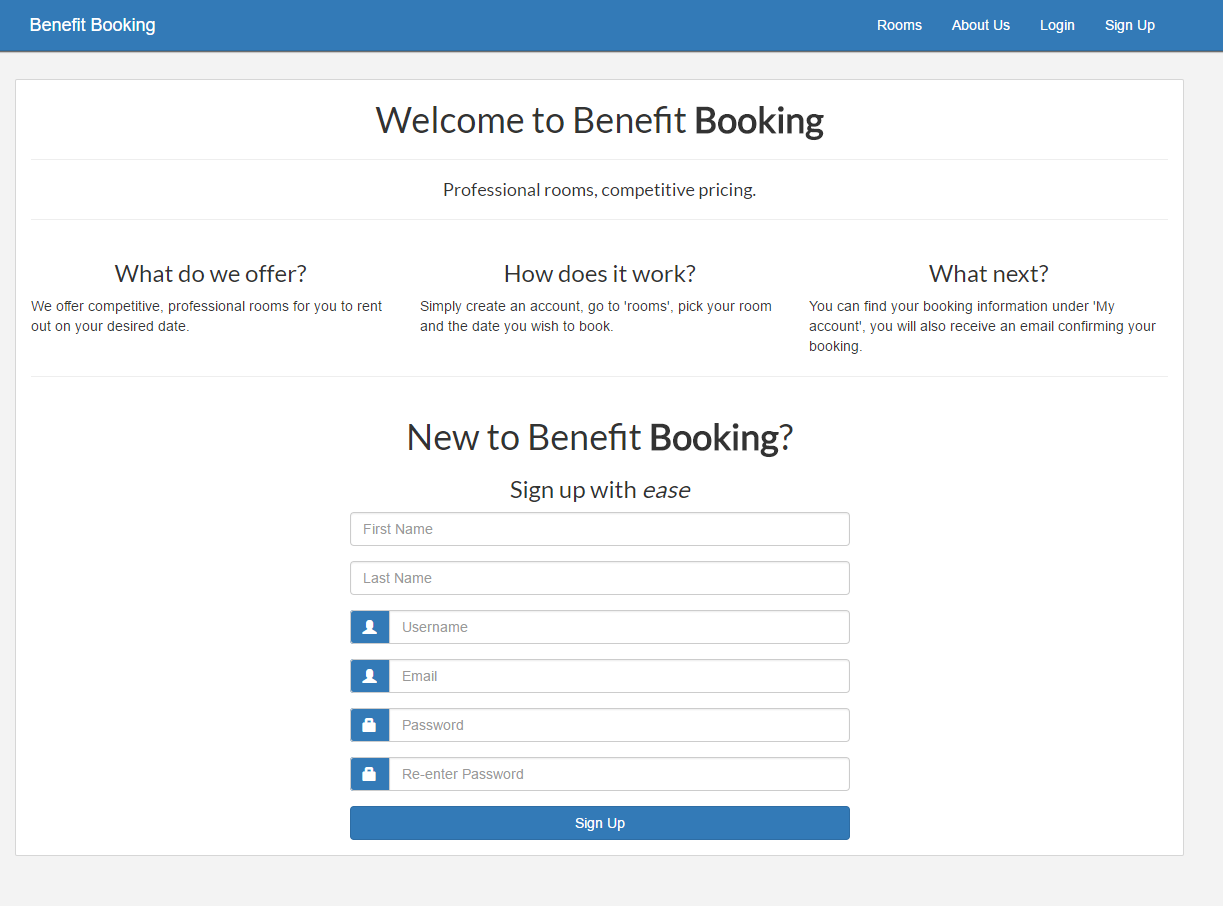
The initial development of the Home page followed the prototype as accurately as possible. The ‘Professional Rooms, Competitive Pricing’ was an experimental design piece used in conjunction with the Bootstrap Carousel (See Figure 9a)

Figure 9a – Home Page Extra

Many issues arose with the use of Bootstrap Carousel. Firstly, the carousel did not fit in with the ‘professional’ theme of the website and was often difficult to manage on different viewports. To combat this, the carousel was hidden on devices smaller than 767px using a media query.

Figure 9b – Home Page (Development Iteration 2)

Tried to use a combination of the two experimental pieces. The info box (See left) often collided with the carousel on smaller devices. As a further development of utilising homepage space, see Figure 9c.

Figure 9c – Home Page (Development Iteration 3)

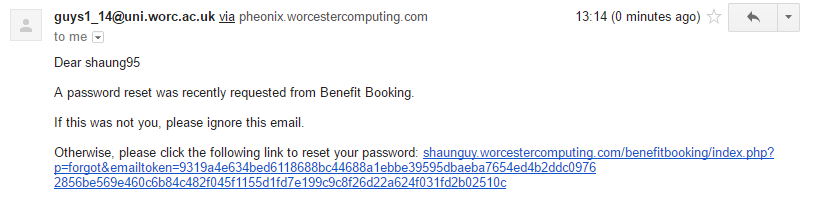
While a lot of work was put into the registration system to make it user friendly and accessible by reducing the number of clicks, it was considered more appropriate and effective to use the homepage as an alternative way of encouraging new visitors to sign up. The slogan ‘Sign up with ease’ encourages new visitors to enter a small amount of information to start the process of booking a room. In addition to this, the new homepage form was a lot more mobile-friendly and fit in with the professional aspect of the website.

**Additional Implementations**

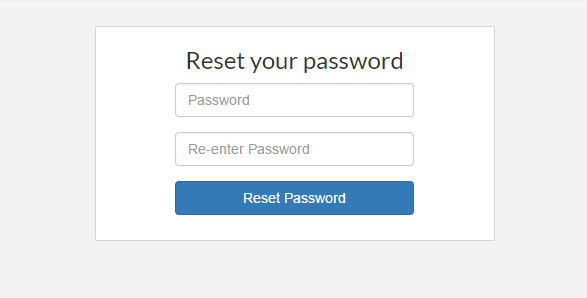
Figure 10 – Password Reset (Step 1)

The password reset system proved challenging in terms of PHP development while also providing functionality to the log-in system. Upon clicking ‘Forgot Password’, the user must enter their Email Address to retrieve the records of their associated account.

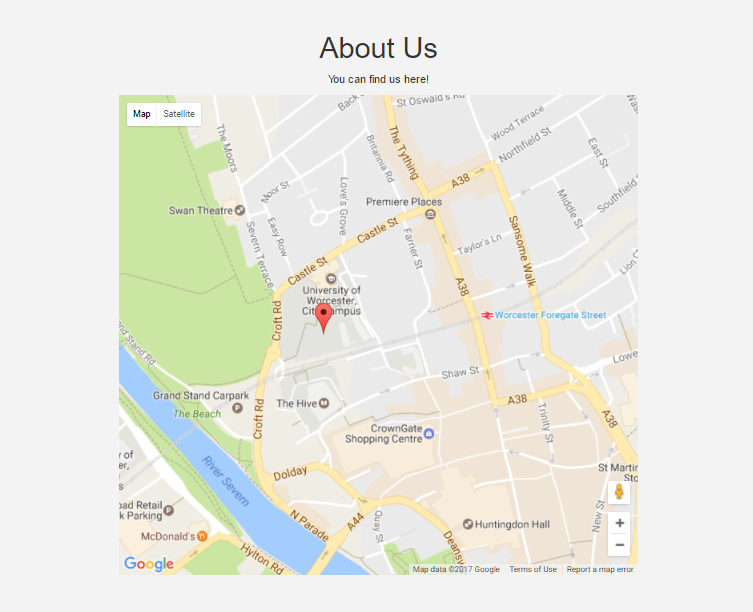
Login Page 🡪 Forgot Password 🡪 Enter Email Address 🡪 Email User if a record is found 🡪 Use URL sent to user to generate a reset form

Figure 10a – Password Reset (Step 2)

The URL is generated using a randomly generated emailtoken which is stored in the database under the found record. The token is appended to a URL which is sent to the email address of the user. When clicked, the user will be redirected to a reset password page. (See Figure 10b)

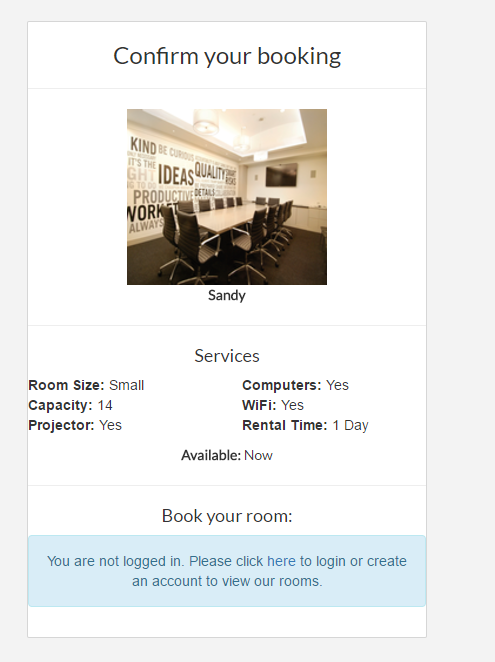
Figure 10b – Password Reset (Step 3)

The user is able to enter a new password of their choice (Using the same form validation as the signup page). Once a new password is entered, the record of the associated user will be updated.

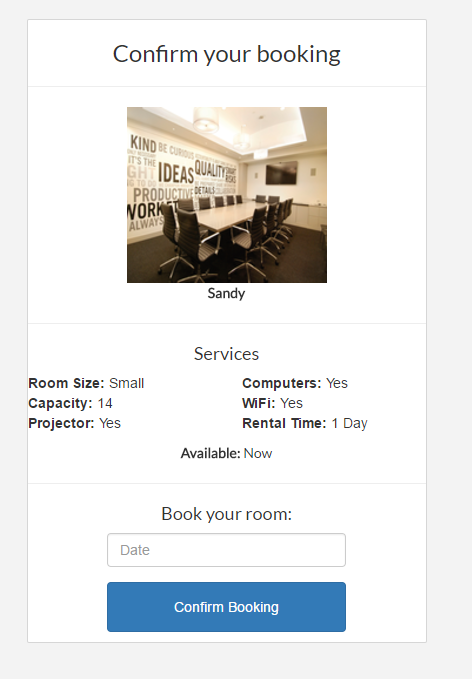
Figure 11 – Using ‘About Us’ page to showcase physical location

Google API used and adapted to show the physical location of the Benefit Booking headquarters.

Room information is passed through the URL.

Figure 12 – Book.php (User not logged in)

This notification will appear if the user is not logged in. To reduce the need for input fields to gather information from a customer, customers are required to log in before booking a room.

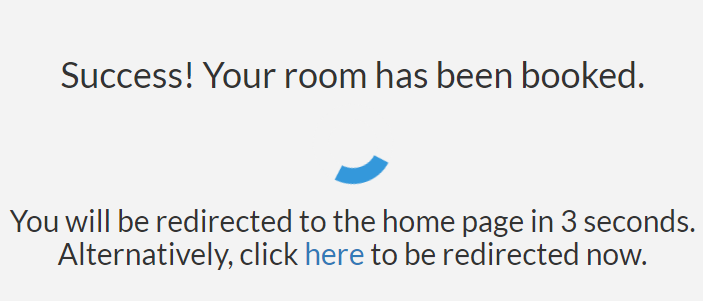
Figure 12a – Book.php (User logged in)

Once the user has logged in, the only information necessary to gather is the date they wish to book the room for. It may cause more inconvenience if a customer had to enter duplicate details if they already had an account.

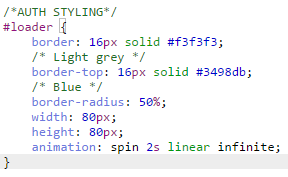
Figure 12b – Book.php ‘Date’ field utilising calendar

When the date field is picked, a calendar opens for the user to choose a date.

Previous dates are disabled for validation purposes.

  
Figure 13 – Auth.php

Originally the auth.php was used for login.php. However, this was added after feedback from Jo Law and Sean Preston showed that visitors do not want to wait 3-5 seconds after log in and a ‘processing time’ felt more suitable for a room booking section.

  
Figure 13a – Auth styling

This code was taken from W3Schools (2017) and provides style for the spinning loader. The code was adapted to make the size bigger and change the colours to keep the design and colour theme consistent.

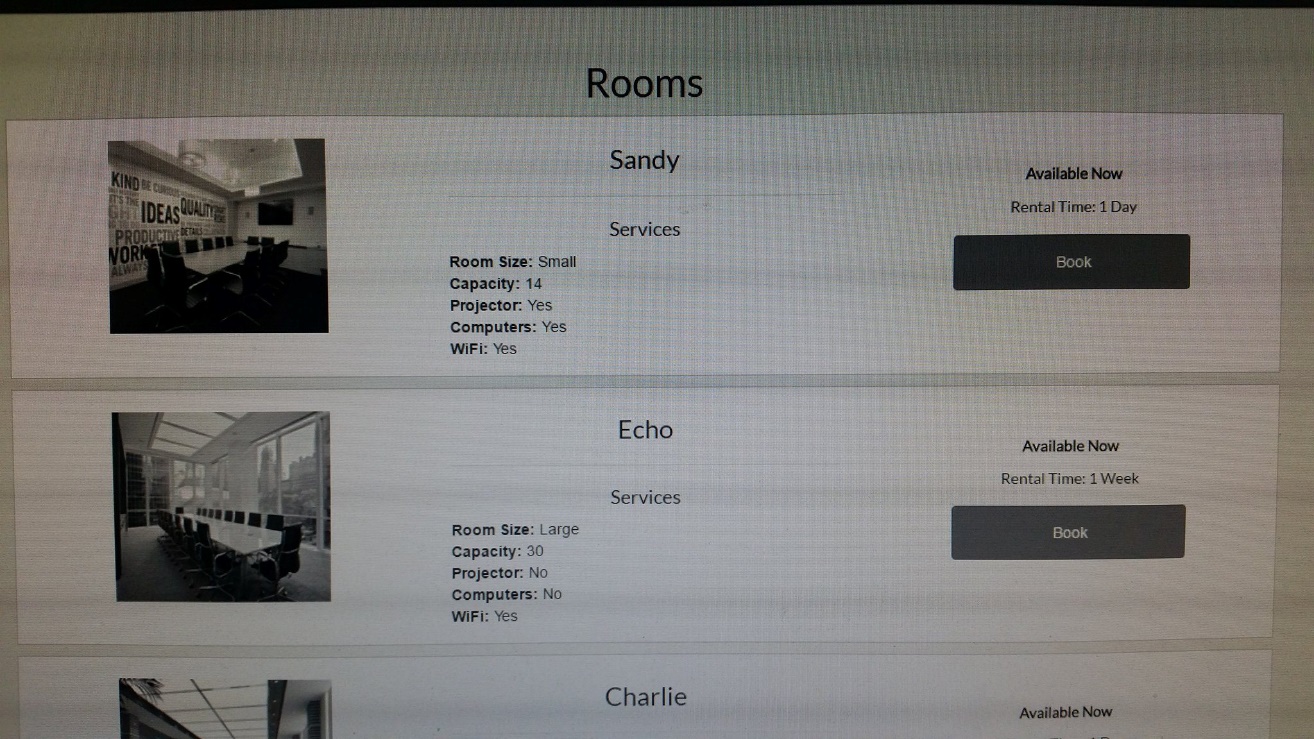
1. **Developing in accordance to web standards and end user requirements (429)**

**Accessibility**

W3 (2017) state that achieving all three accessibility standards (A, AA and AAA), all three priority checkpoints must be addressed. The following priority 1, 2 and 3 criteria listed below are taken from W3 (2017) for reference. Kumar and Owston (2016) mention that there are two methods available for accessibility testing, one of which consists of testing with end-users while second route involves manually testing the application to ensure that it conforms to WCAG. Since user testing is not entirely feasible for this project, the following screenshots will demonstrate the application adhering to some of WCAG’s accessibility criteria.

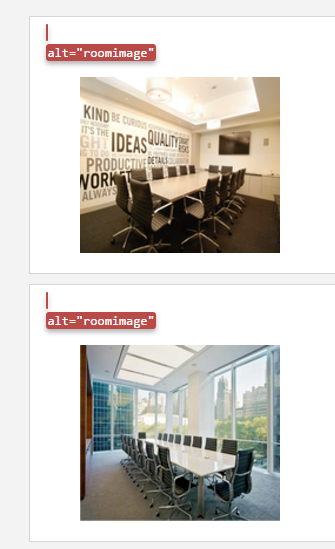
**Example Priority 1 criteria met:**

* **Ensure that all coloured content is understandable without colour:**

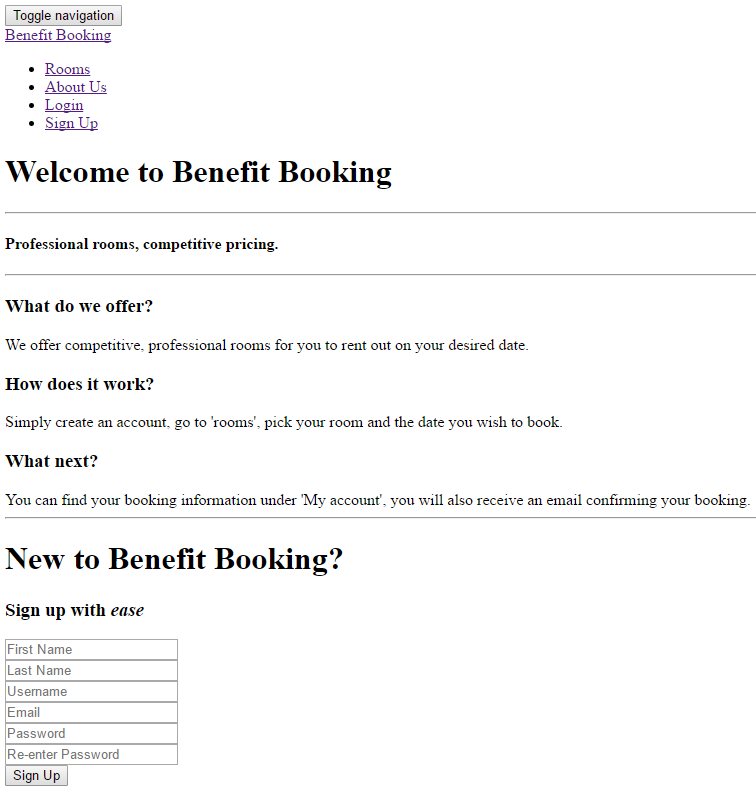
  
Figure 14 – Rooms page displayed without colour.

* **Provide an alternative for every non-text element:**

  
Figure 15 – Display Alt Attributes

  
Figure 15a – Rooms page with alt attributes shown

* **Organise documents so they may be read without style sheets:**

   
Figure 16 – Home (No stylesheet) Figure 16a – Rooms (No stylesheet)

**Example Priority 2 Criteria met:**

* **Use style sheets to control layout and presentation (e.g. CSS ‘font’ property rather than HTML ‘font’ element:**

All layout and presentation has been accomplished throughout the use of a dedicated, personal stylesheet, bootstrap or in-line styles for unique elements.

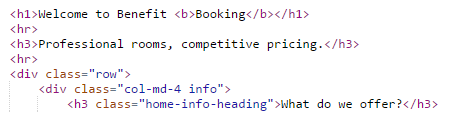
* **Use relative rather than absolute units for mark-up and style sheet property values (e.g. ‘em’ rather than ‘pt’ or ‘cm’:**

  
Figure 17 – CSS management example

* **Create documents that validate to published formal grammars (e.g. Include a document type declaration at the beginning of document):**

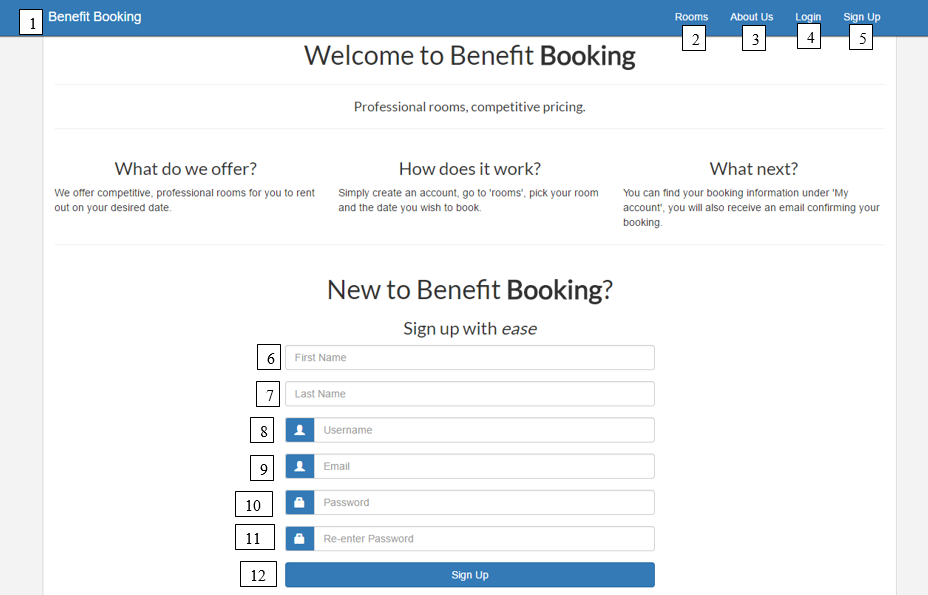
  
Figure 18 – Start of header.php

* **Use header elements to convey document structure rather than for font effects:**

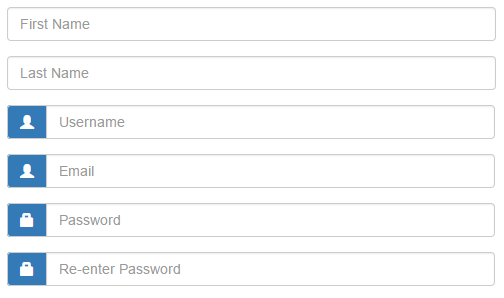
  
Figure 19 - Example of hierarchical ordering Figure 19a – Example of hierarchical ordering

**Example Priority 3 Criteria met:**

* **Establish a logical tab order for content on the website:**

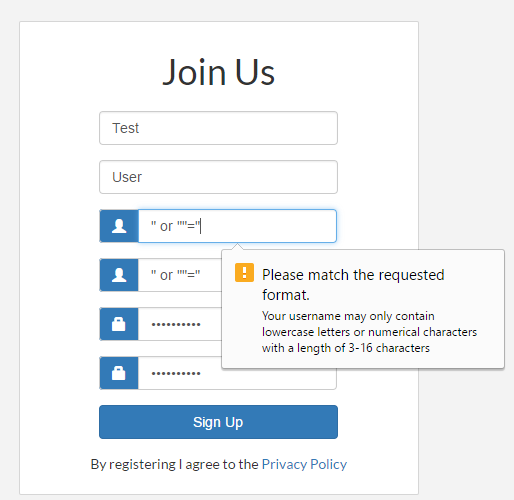
  
Figure 20 – Tab order

* **Until a user enters information, include place-holding characters in text areas:**

  
Figure 21 – Placeholder information

**Security**

Stringent validation has been implemented into the login system to ensure that injection threats are as mitigated as possible. (Williams and Wichers, 2017) list injection as the number 1 threat for Web Applications in 2017 with broken authentication and session management as the second biggest threat.

  
Figure 22 – Example SQL Injection attempt

**Not malicious (Prepared statements):**

Input is no longer converted to a string but is instead read as:

Username = “ or “”=”

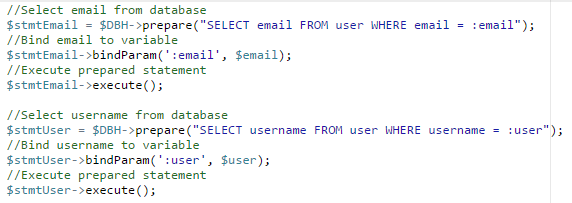
The result?

Your username is now “ or “”=”… Without validation.

**Malicious (Without Prepared statements):**

Input is converted to a string

The string entered in Figure 23 is an example of a SQL injection by W3Schools (2017) and would typically be used to gain access to information stored in the database. Cairns and Somerfield (2017) explain that limiting validation to accept only logical values can help to alleviate the risk of attack. In addition to this, prepared statements are used as a method of sanitising data input. PHP.net (2017) mention that prepared statements interpret data entered ‘as is’ rather than converting them to a string. (See Figure 23a)

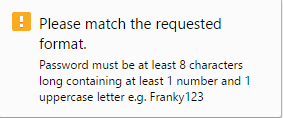
  
Figure 22a – Binding Variables

**Encryption**

Although there are several technologies that exist for encryption within a database (Fuhry et al., 2016), there have been many incidents involving reputable companies storing passwords using insecure algorithms such as MD5. (White, 2015) strongly credits the ‘password’ hash function as a method of encrypting data using PHP and furthermore explains that every time you use the password\_hash function, it generates a random salt for each password. Php.net (2017) also comments on the salting function and recommends using the default salting function of password\_default rather than manually generating one. As such, the below screenshot demonstrates the implementation of the password\_hash function within Benefit Booking.

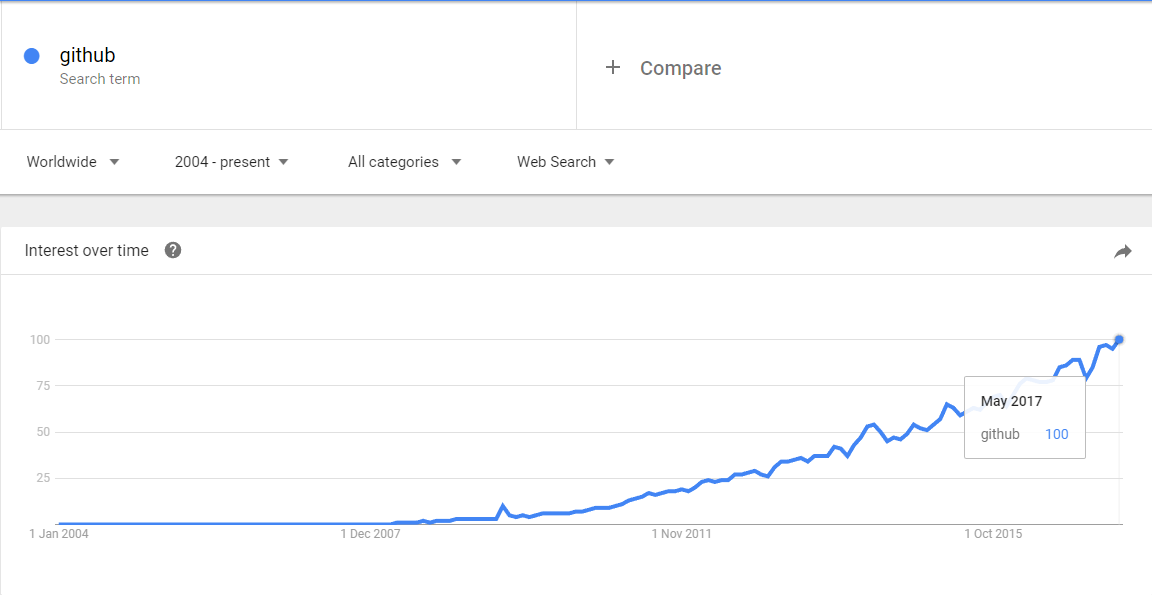
  
Figure 23 – Implementation of password\_hash function.

In addition to encryption, strong passwords are also an important implementation to prevent cracking. While OWASP (2017) suggests using a password length of 10 with a combination of uppercase, lowercase numerical and special characters, Shay et al. (2016) argues that an increase in the number of password creation attempts results in a ‘less usable’ application. To compromise between security and usability, a combination of security and accessibility is addressed in figure 24a.

  
Figure 23a – Password Rules

1. **Implementation issues and solutions**

Numerous issues arose using XAMPP that required small tweaks to work properly. Firstly, after setting up the PHP mail function for the forgot password page, no e-mail would send to the test email address. Secondly, uploading to the server caused unnoticed database login issues that were correctly after understanding that the student username and password was required in place of ‘root’. In place of XAMPP, GitHub, along with an FTP server, could have potentially been a more feasible solution to solely localhost development. Longo and Kelley (2016) list several advantages that the version control system offers including additional usability, project management features and online storage. Figure 25 shows the drastic increase in popularity that GitHub has received since 2007 and developers continue to use the version control system.

  
Figure 24 - (Google Trends, 2017)

|  |  |  |
| --- | --- | --- |
| Issue | Solution | Evidence |
| Login, My Account & Sign Up navigation are shown regardless of whether the user is logged in or not | Add a check to see if SESSION\_ID has been set. if it has, show Logout and My Account. If not, show Login. | Figure 25 – Code for SESSION\_ID |
| Database cannot connect – Fatal error | XAMPP did not require a login system (See figure 25a), after | Figure 26 - Issue    Figure 26a - Original Code    Figure 26b - Changed code |
| Email does not send to user | After searching for fixes, it was found that XAMPP does not have a dedicated email server and the application should be uploaded to a dedicated server (worcestercomputing in this case)  After uploading the application code, the email functionality worked as expected. | Figure 27 – Email received by Benefit Booking |

Figure 28: Issue and Solution Table

1. **Testing and debugging**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test No.** | **Date Tested** | **Test Case/ type (what you are testing)** | **Location (where is it located)** | **Test criteria/ (what does it need to do to pass the test)** | **Expected Result** | **Actual Result** | **Action to fix (if applicable)** | **Screenshots** |
| 1 |  | Form Validation (Registration) | Registration page | The form should not post information if any fields are left empty **Test case:** Leave a single field blank | The form **should not** POST any information. | Form does not POST any information, validation intervention. | N/A |  |
| 1a |  | Form Validation (Registration) | Registration page | The form should not post information if any fields are left empty **Test case:** Leave all | The form **should not** POST any information. | Form does not POST any information, validation intervention. | N/A |  |
| 2 |  | Form Validation (Registration) | Registration page | The form should only accept an email address with “@\_\_\_\_\_.\_\_\_” extensions. (e.g. test@test.com) **Test case:** [testcase@test.com](mailto:testcase@test.com) | This case **should** pass validation | Validation intervention. | N/A | Figure 29 – Email Validation |
| 2a |  | Form Validation (Registration) | Registration page | **Test case:** test@com | This case **should not** pass validation | Validation prompt incorrect, needs changing to ‘Follow the format @address.com’ | Changed error message in ‘Title’ element of sign-up form for email address validation. | Figure 30 – Incorrect email validation    Figure 30a – Validation changed |
| 2b |  | Form Validation (Registration) | Registration page | Abc123 | This case **should not** pass validation | Validation intervention. Does not include @ character. | N/A | Figure 31 – Validation for abc123 |
| 2c |  | Form Validation (Registration) | Registration page | @googlemail.com | This case **should not** pass validation | Validation intervention. | N/A | Figure 32 – Validation for @googlemail.com |
| 3 |  | Form Validation (Registration) | Registration page | Password should only accept passwords with the following criteria: One uppercase character, one lowercase character, 1 number and at least 8 characters long. **Test case:** Testing123 | This case **should** pass validation | Test case passed validation. | N/A |  |
| 3a |  | Form Validation (Registration) | Registration page | **Test case:** testing123 | This case **should not** pass validation **Reason:** No uppercase character. | Test case does not pass validation.  Reason: Validation intervention. | N/A | Figure 33 – Validation for password |
| 3b |  | Form Validation (Registration) | Registration page | **Test case:** testing | This case **should not** pass validation. **Reason:** Too short, no numerical characters, no uppercase character. | Test case does not pass validation.  Reason: Validation intervention. | N/A | Figure 34 – Validation for password |
| 4 |  | Form Validation (Registration) | Registration page | Test that password and re-enter password is verified before POST. **Test case:** Testing123 as password and re-enter password. | This case **should** pass validation. | Test case passes validation. | N/A |  |
| 5 |  | Form Validation (Login) | Login.php | Test that form does not POST if fields are empty. **Test case:** Leave field(s) empty | The forms **should not** POST. | Form does not POST.  Reason: Validation intervention. | N/A | Figure 35 – Login form validation |
| 6 |  | General functionality | Header.php and home.php | Ensure that the correct buttons and fields are shown depending on whether a user is logged in or not. **Test case:** User is logged in (Home page) | User should see the following options on navigation bar: My Account, Log Out.  The user should also **not** be able to see the sign-up form on the homepage. | Test did not pass | See Figure 26 in “Issue and Solution” table. Issue has been fixed. | See Figure 25 in “Issue and Solution” table. |
| 7 |  | General functionality | Header.php and home.php | Ensure that the correct buttons and fields are shown depending on whether a user is logged in or not. **Test case:** Logged out. | User should see the following options on navigation bar: Log in  The user should also be able to see the sign-up form on the homepage as well as under the ‘Login’ section on the home-page. | Test did not pass | See Figure 26 in “Issue and Solution” table. Issue has been fixed. | See Figure 25 in “Issue and Solution” table. |
| 8 |  | Page Speed | Entire application | Page should ideally load within 3 seconds. | Page will load within an adequate time. (Under 5 seconds) | The page loads with a good pagespeed result. | N/A | Figure 36 – Performance Score  Figure 36a – Page Details |
| 9 |  | Compatibility | Entire application | Test that the application functions on all browsers. | Application will function correctly on all browsers. | The application functions correctly on the following browsers: Google Chrome, Mozilla Firefox, Microsoft Edge and Internet Explorer. | N/A |  |
| 10 |  | Responsive content (Mobile) | Entire application | Test that the application is responsive for **mobile** devices. | The application should display information clearly and correctly. | Application displays correctly. | N/A | See Appendix 1. |
| 11 |  | Responsive Content (Tablet) | Entire application | Test that the application is responsive for **tablet** devices. | The application should display information clearly and correctly. | Application displays correctly. | N/A | See Appendix 1a |
| 12 |  | Responsive Content (Desktop) | Entire application | Test that the application is responsive for **desktop** devices. | The application should display information clearly and correctly. | Application displays correctly. | N/A | See Appendix 1b |
| 13 | 06/05/2017 | All links redirect correctly | Entire application | Ensure that all links are fully functional and redirect to the correct pages. | All links must redirect to their associated page. (e.g. My account must redirect to myaccount.php) | All links worked as expected. | N/A |  |

Figure 37 - Site test plan

1. **References**

Cairns, C. and Somerfield, D. (2017). The Basics of Web Application Security. [online] martinfowler.com. Available at: https://martinfowler.com/articles/web-security-basics.html [Accessed 5 May 2017].

Cao, J. (2016). Web Design Trends 2016: How Cards Dominate Design - Designmodo. [online] Designmodo. Available at: https://designmodo.com/web-design-cards/ [Accessed 4 May 2017].

Egri, G. and Bayrak, C. (2014). The Role of Search Engine Optimization on Keeping the User on the Site. Procedia Computer Science, 36, pp.335-342.

Fuhry, B., Tighzert, W. and Kerschbaum, F. (2016). Encrypting Analytical Web Applications. Proceedings of the 2016 ACM on Cloud Computing Security Workshop

Google Trends. (2017). Google Trends. [online] Available at: https://trends.google.com/trends/explore?date=all&q=github [Accessed 6 May 2017].

Kumar, K. and Owston, R. (2016). Evaluating e-learning accessibility by automated and student-centered methods. Educational Technology Research and Development, 64(2), pp.263-283.

Li, H., Rau, P., Fujimura, K., Gao, Q. and Wang, L. (2012). Designing Effective Web Forms for Older Web Users. Educational Gerontology, 38(4), pp.271-281.

Longo, J. and Kelley, T. (2016). GitHub use in public administration in Canada: Early experience with a new collaboration tool. Canadian Public Administration, 59(4), pp.598-623.

Osborne, T. (2015). Color Contrast for Better Readability | Viget. [online] www.viget.com. Available at: https://www.viget.com/articles/color-contrast [Accessed 5 May 2017].

Owasp.org. (2017). Authentication Cheat Sheet. [online] Available at: https://www.owasp.org/index.php/Authentication\_Cheat\_Sheet#Implement\_Proper\_Password\_Strength\_Controls [Accessed 5 May 2017].

PHP.net. (2017). PHP: Prepared Statements - Manual. [online] Available at: http://php.net/manual/en/mysqli.quickstart.prepared-statements.php [Accessed 5 May 2017].

Shay, R., Cranor, L., Komanduri, S., Durity, A., Huh, P., Mazurek, M., Segreti, S., Ur, B., Bauer, L. and Christin, N. (2016). Designing Password Policies for Strength and Usability. ACM Transactions on Information and System Security, 18(4), pp.1-34.

Trivago. (2017). https://www.trivago.co.uk/ [online] Available at: <https://www.trivago.co.uk>

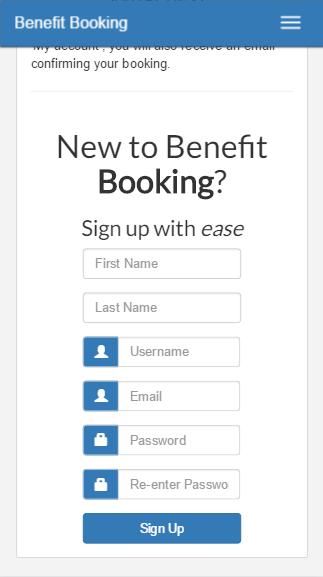
W3schools.com. (2017). SQL Injection. [online] Available at: https://www.w3schools.com/sql/sql\_injection.asp [Accessed 5 May 2017].

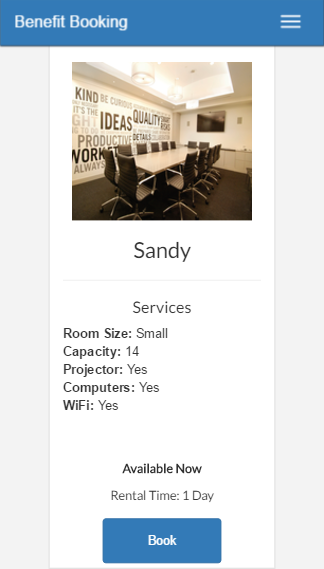
White, E. (2015). Password security in modern PHP. [online] Ibm.com. Available at: https://www.ibm.com/developerworks/library/wa-php-renewed\_2/index.html [Accessed 5 May 2017].

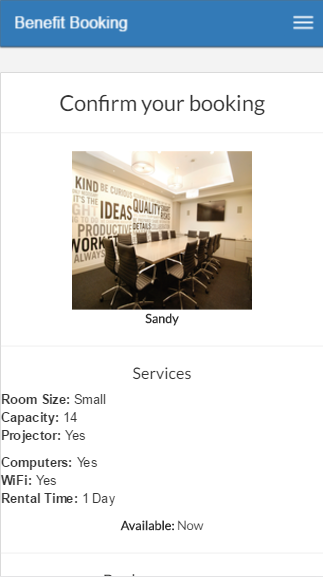
Williams, J. and Wichers, D. (2017). OWASP Top 10 - 2017. 1st ed. [ebook] OWASP, p.5. Available at: https://www.owasp.org/index.php/Category:OWASP\_Top\_Ten\_Project [Accessed 5 May 2017].

1. **Appendix**

**Appendix 1 – Mobile View**

  
Figure 38 – Home Page – Mobile View

  
Figure 38a – Rooms Page – Mobile View

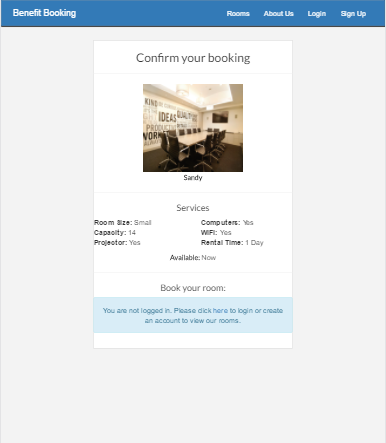
  
Figure 38b – Book Page – Mobile View

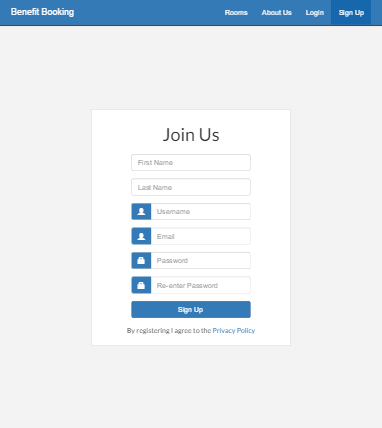
**Appendix 1a – Tablet View**

  
Figure 39 – Home Page – Tablet View



Figure 39a – Rooms page – Tablet View

  
Figure 39c – Book page - Tablet View

  
Figure 39d – Sign Up Page – Tablet View

**Appendix 1b – Desktop View**

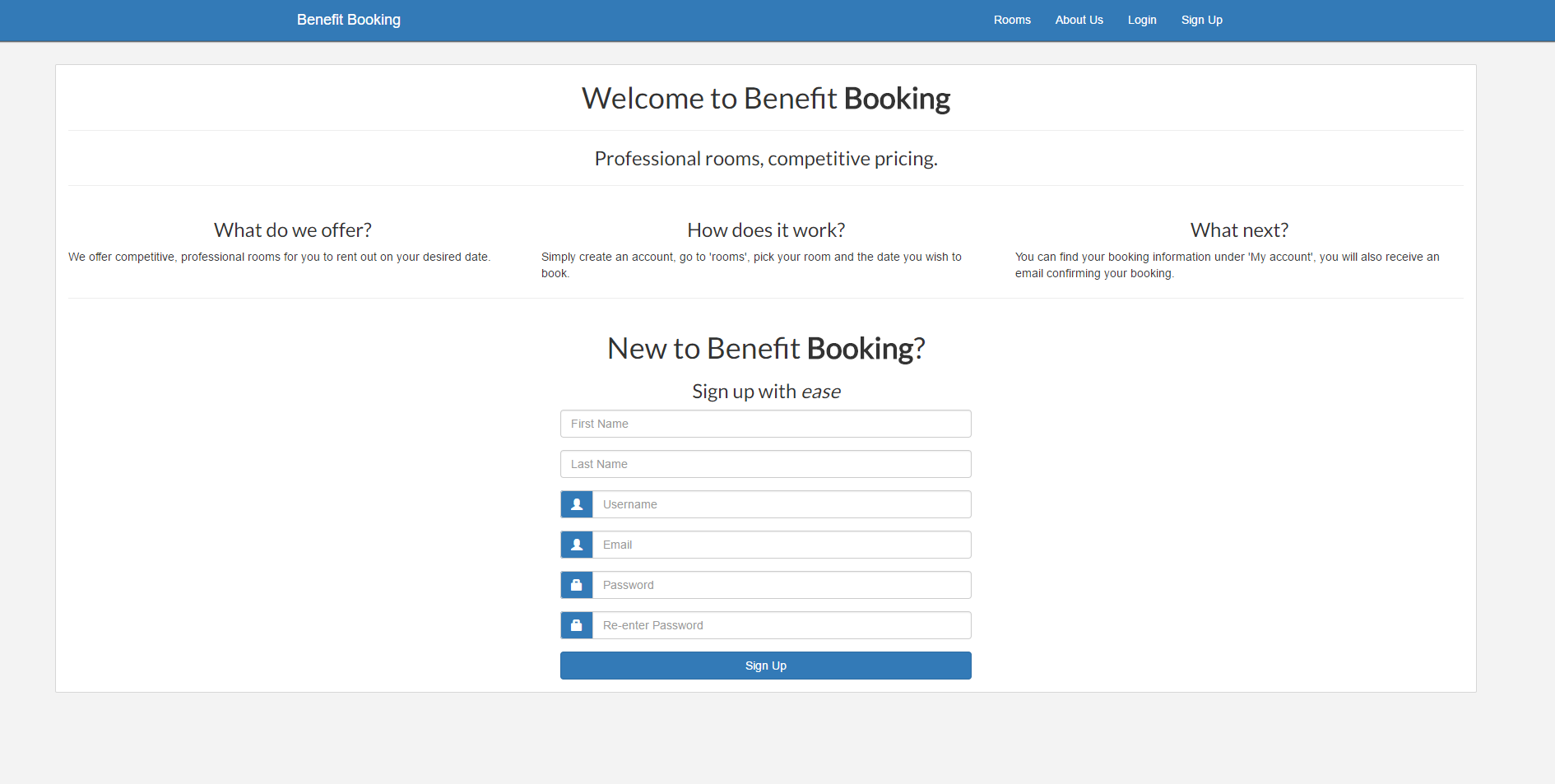


Figure 40 – Home page – Desktop View

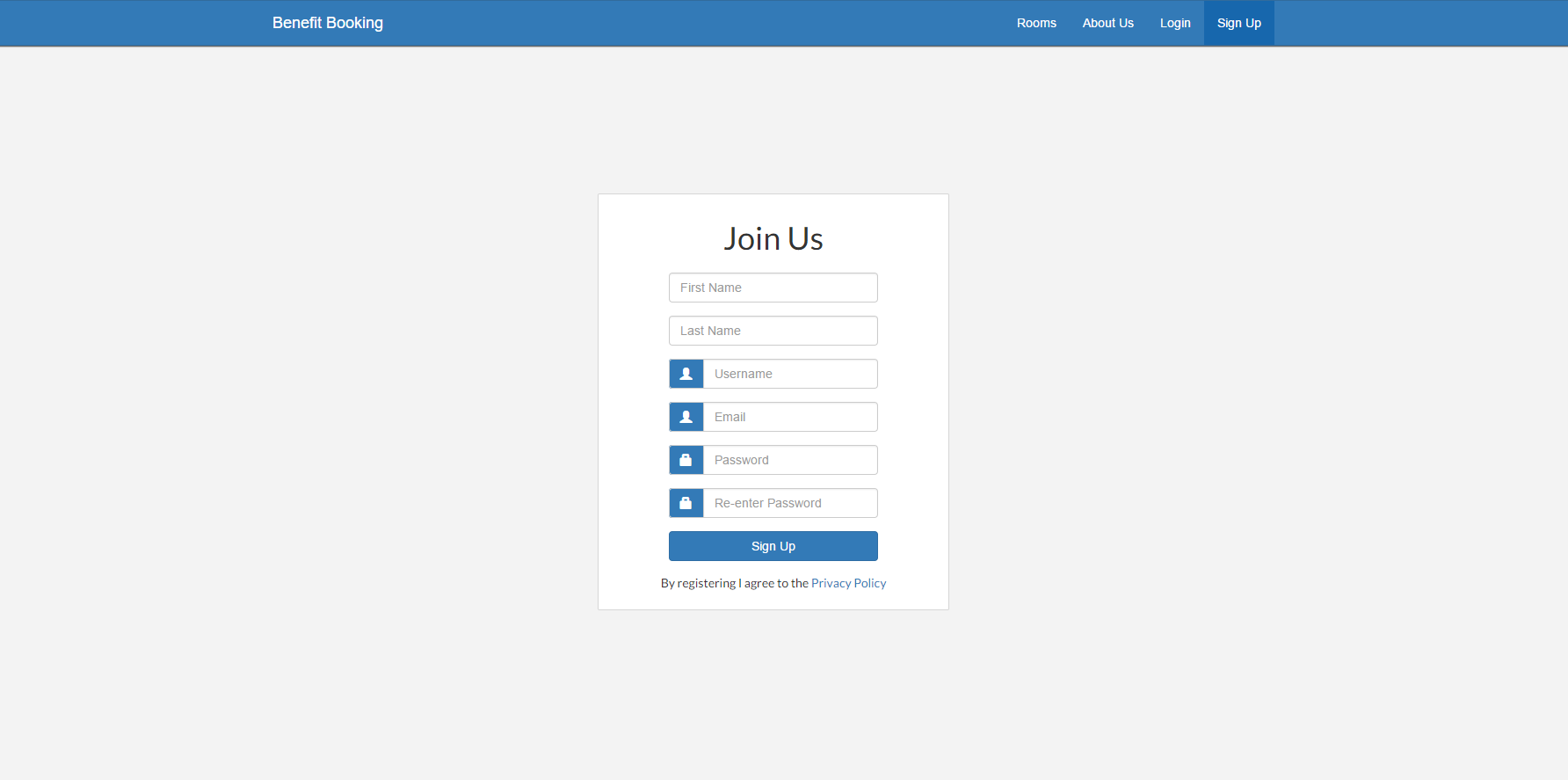
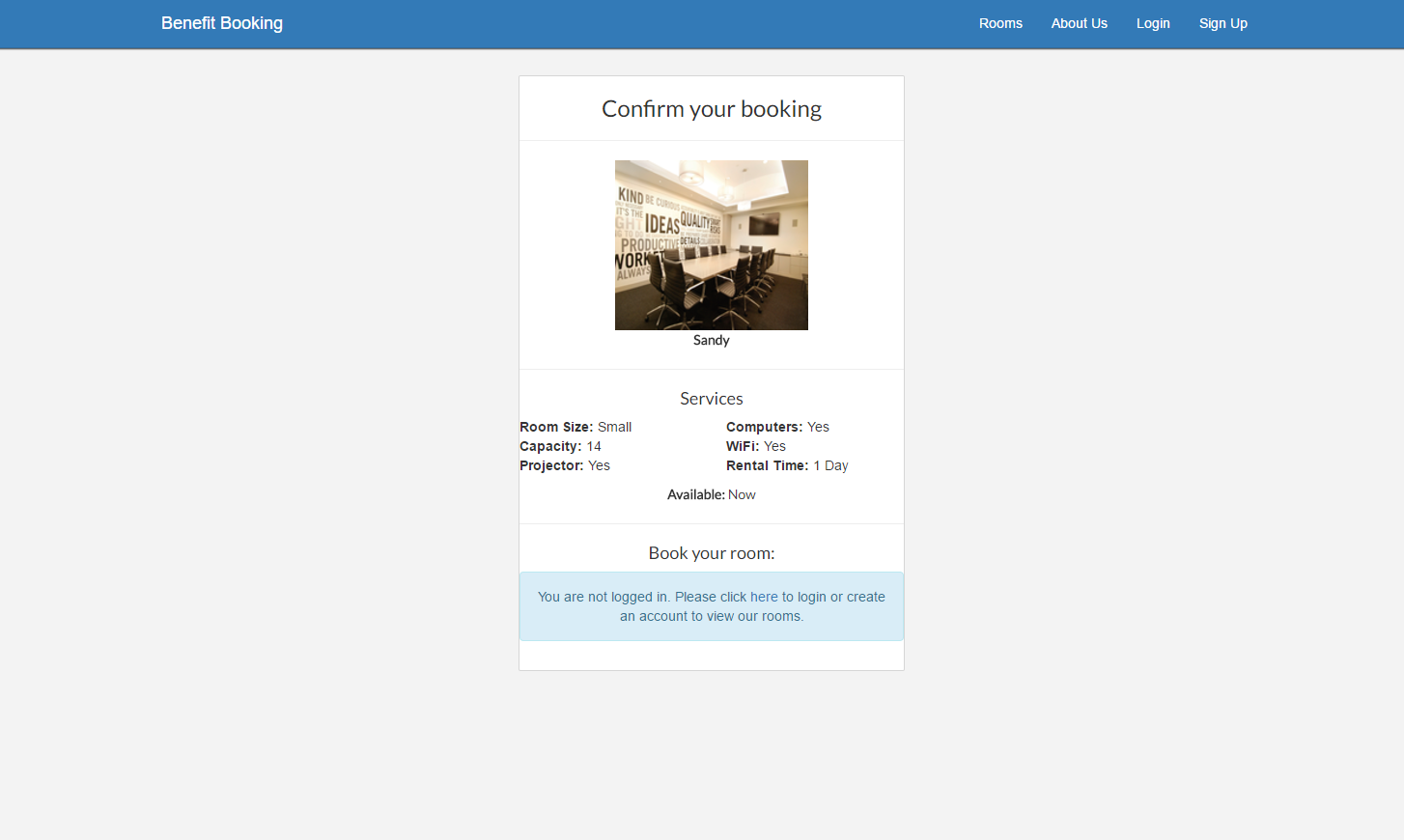


Figure 40a – Signup Page – Desktop View



Figure 40b – Rooms Page – Desktop View

  
Figure 40c – Book page – Desktop View