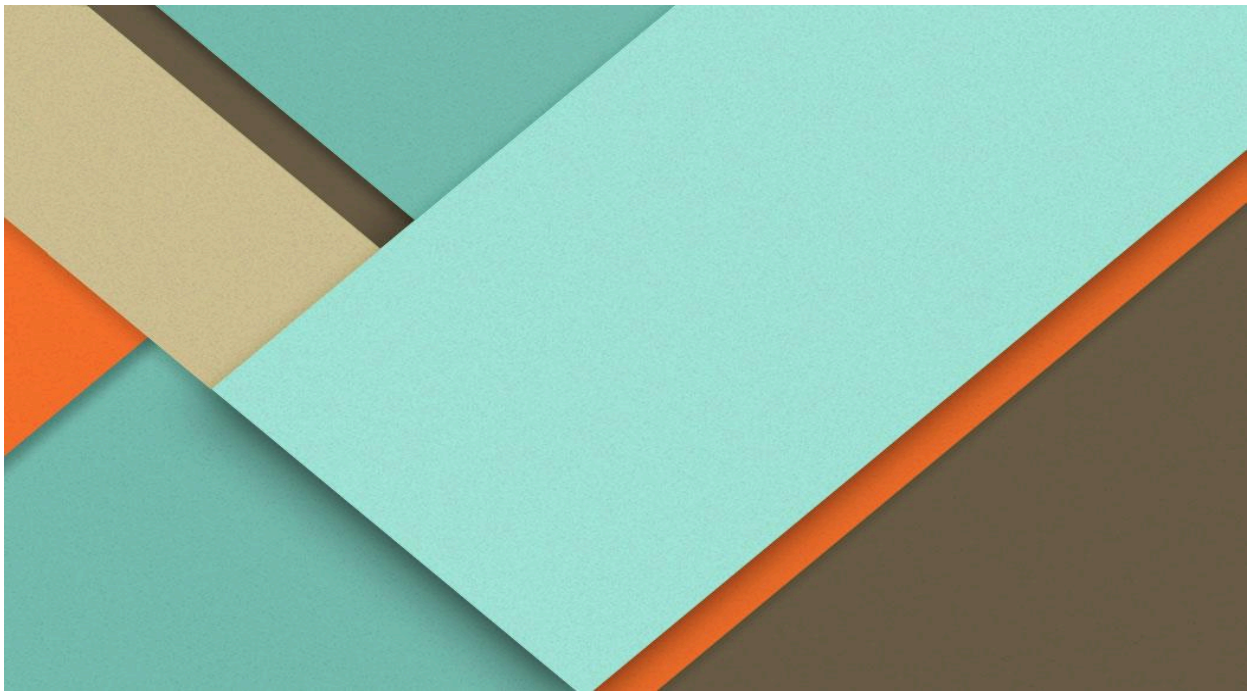
Bookit: A Room Booking Web App - DEVELOPMENT REPORT AND TIMELINE

[09/04/2024]



By: Noah Tambala (ntamb002@gold.ac.uk), Ben Craddock (bcrad001@gold.ac.uk), Khar Chew (kchew001@gold.ac.uk), Syed Sahaf (ssaha003@gold.ac.uk), Jake Brunnen (jbrun001@gold.ac.uk), Spike Elliot (selli006@gold.ac.uk), Shaquille Muhammad Uddin (suddi006@gold.ac.uk)

Group name: Kelvin's Kittens

Second Year Computer Science BSc: IS52018C/F/S - SOFTWARE PROJECTS/ COMPUTING PROJECT 2 (2023-24)

Goldsmiths University of London

Preface

This is our report for the web application “Bookit” - detailing development and evaluation for said application.

Source code: [Bookit roombooking (github.com)]

Hosted website on Goldsmith’s virtual servers: https://www.doc.gold.ac.uk/usr/199/

Bookit was developed by Kelvin’s Kittens:

- Kelvin:
    - Khar “Kelvin” Chew
- The kittens:
    - Noah Tambala
    - Jake Brunnen
    - Syed Sahaf (Over 6ft tall)
    - Spike Elliot
    - Ben Craddock
    - Shaquille M Uddin

Contents

Introduction

Problem statement

When consulting our lecturers, as well as the two society leaders that are both members of our project group, we identified several inadequacies with the current method for booking rooms at Goldsmiths, University of London. The current system relies entirely on Google Forms for data entry and emailing said forms back and forth between coordinators to confirm the booking. This system is inefficient and discourages extra curricular planning due to the difficulty surrounding securing spaces for such events that society leaders face.

The current system also has shortcomings when it comes to selection of rooms when a society leader wishes to book - wherein they are able to select aspects of a room and are then assigned a room by coordinators that fits their preferences as best as possible - while this system does allow for societal events to have the necessary facilities - it is restrictive as it does not allow for the individual booking the room to choose the room they wish to book.

The current room booking system displays further inadequacies when regarding the scheduling of spaces for educational events like lectures: allowing for bulk bookings over long time periods. This results in rooms being booked for events that may not end up taking place – wasting vital university facilities.

What if all this was handled in the same application, and could be seen by all those involved?

Our solution

A comprehensive room booking application: Bookit.

We believe that through automation of the booking process, we will allow lecturers and societies to book appropriate spaces with ease.[1]

This is done mainly to enrich the experience of those wishing to reserve a space - allowing for more time to prioritise and plan their actual event as opposed to stressing about securing a space - improving the experience for all involved.

[1] Perera, K. M. P. B. N. LECTURE HALL SCHEDULING AND TIMETABLE MANAGEMENT SYSTEM. Diss. 2021. p. 1

According to Sangwon Park (et al.), users of online booking systems for hotels were likely to search for additional information by comparing specific attributes of hotel rooms.[2] With this in mind, we decided that our application should provide similar functionality for booking spaces at Goldsmiths.

The system will allow for bookings to be made, monitored, approved, denied and altered all in the same application - making things easier for all parties involved and remedying the aforementioned problem.

Scope

For this project we agreed as a group that we should follow the choice set model, a three-stage, consecutive and funnel-like procedure comprising an awareness set, consideration set and a final choice.[3] A lecturer or student/society rep will visit the booking page and be able to select from a menu to select the space they wish to book (awareness). The system will then provide the user with a list of bookable rooms (consideration). The user will then select a room to book to confirm this booking (final choice) – at which point, a coordinator would be prompted to approve this. The coordinator will interact with the system in the sense that they could confirm or deny bookings that have been presented to them by other users.

We also aim to reduce the lengthy process of risk assessment creation that society leaders must go through each time they wish to book a room. Bookit will automatically template a risk assessment for them should that be necessary to make their booking. This will also reduce the two-week wait period that is currently in place for confirmation of a booking by speeding up the very slow, non-automated confirmation process that relies on emails between coordinators as opposed to a single application like Bookit. We hope that by improving this process it will encourage more extra-curricular activities to take place at Goldsmiths.

MVP

Given the above considerations - we felt as a group that it would be beneficial to develop an MVP - minimum viable product - to reduce our proposal and considerations down to the most basic set of deliverable aspects to let our stakeholders know what to expect from our project. This also eliminates the possibility of unnecessarily bloating our application with unnecessary features that will not appeal to our intended users.

---

[2] Park, Sangwon & Yin, Yizhen & Son, Byung-Gak; Understanding of online hotel booking process: A multiple method approach. Journal of Vacation Marketing. 25 (2018)
[3] Um S and Crompton JL; The roles of perceived inhibitors and facilitators in pleasure travel destination decisions. Journal of Travel Research 30(3) (1992); p18-25

As detailed in our midterm proposal, we took an iterative approach to refine the model that we would follow for developing our MVP. From that process - we decided that our barebones MVP would be represented by the following model, which we have named R2:

R2

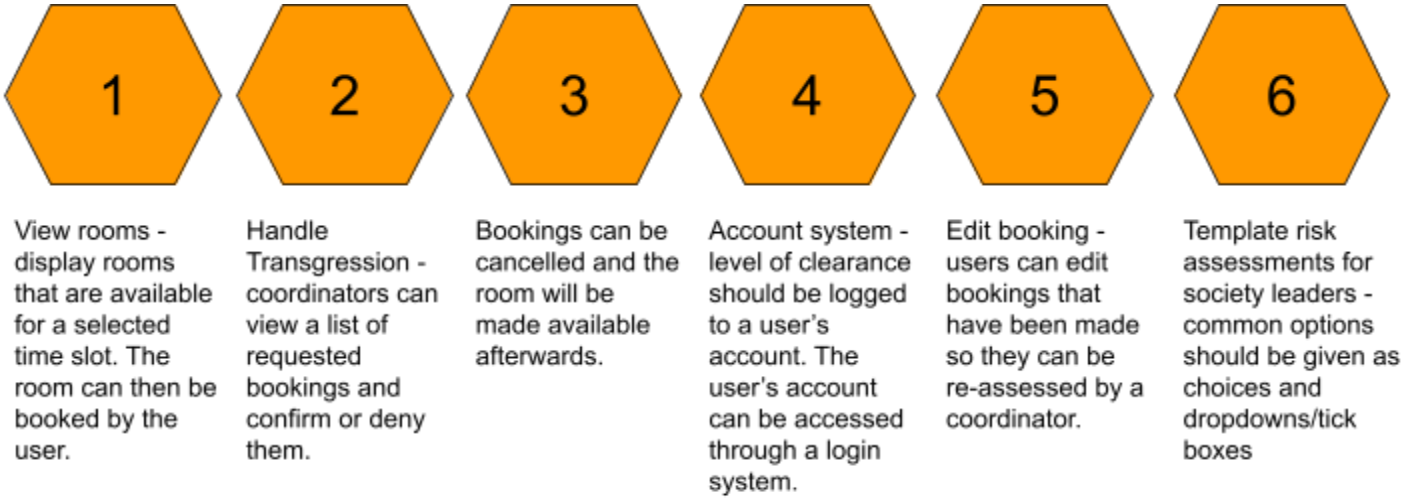


Figure 1: Revision 2 MVP Model

This details the absolute minimum version of the MVP that would satisfy all our aforementioned considerations for our defined problem. From this - we developed another model for our proposed solution that contained all planned features for the solution that were not added to the MVP as while they do directly improve the user experience they are not entirely necessary features for a basic version of Bookit.

We decided to call this model of the proposed solution R4:

R4

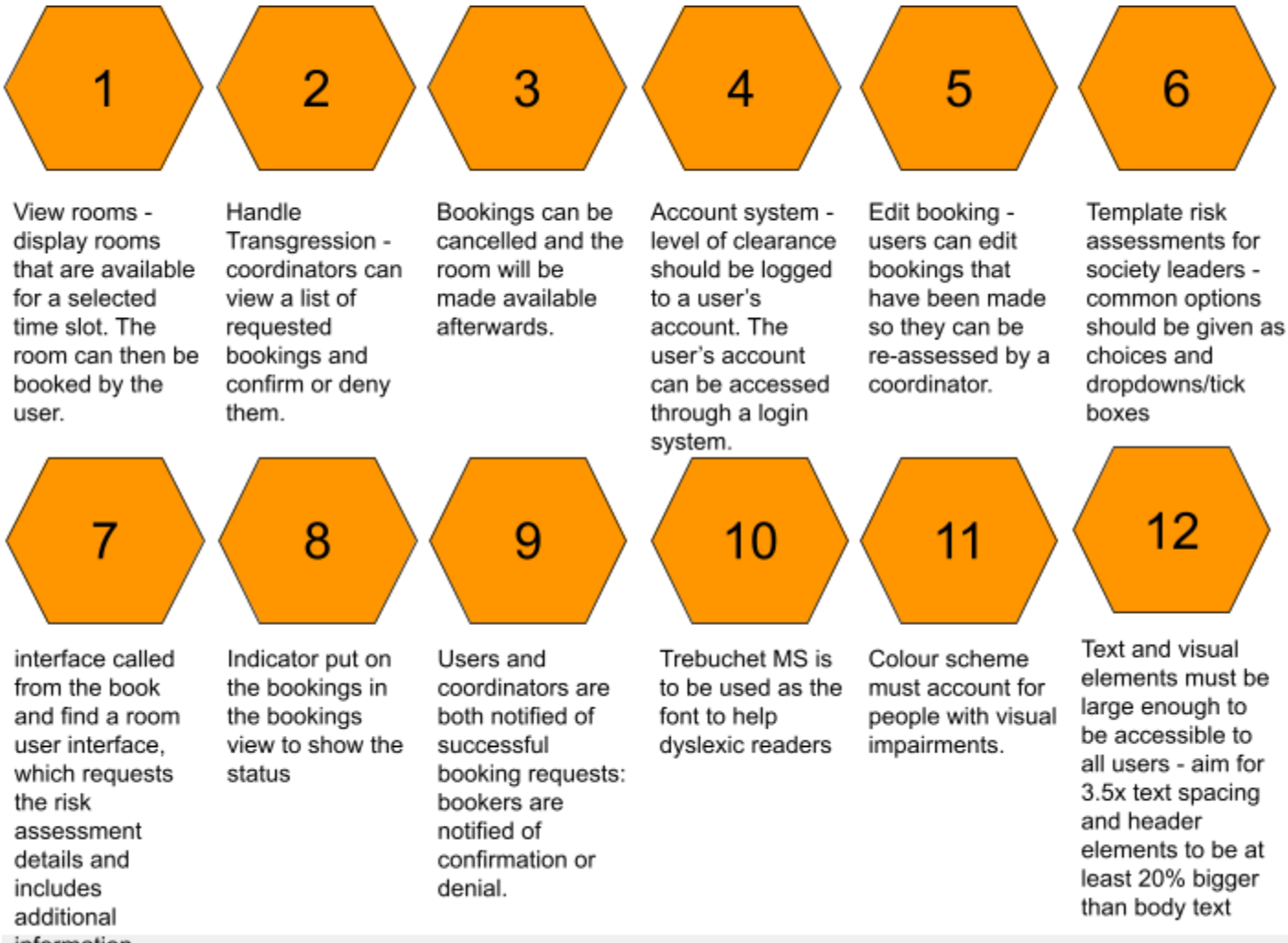


Figure 2: Revision 4 MVP Model

Planning

Our Approach

From the start of our project we agreed to follow an iterative approach for as many aspects of our project as possible. We agreed to meet each week with as many members of our group as possible each Friday afternoon to discuss progress and tasks moving forward. Furthermore, due to the larger size of our group it was not a disaster if we had one or two no-shows for these meetings as it usually just meant that these individuals would get the last pick for their weekly tasks. We felt it was necessary to determine a week-by-week structure for the development of our MVP and these weekly meetings were instrumental in

ensuring that progress was made consistently. This structure also allowed for us to effectively distribute and manage our time resources both as individuals and as a group when it came to working on the project.

In the first few weeks of the course we also drafted a list of each group individuals' strengths when it came to software development (see the Appendix). This proved useful in assigning roles that would suit each member of our group throughout the project (see - "Group management").

Because this application is designed to simplify a pre-existing process, it is crucial that it is easy to use. This means it must suit the preferences of our stakeholders and proposed users. Therefore, we applied user-centred techniques to our development process - by ensuring we continue to take an iterative approach and consistently feedback on changes to our system with our stakeholders and users. Furthermore, there are two society leaders in our project group that were happy to also act as product testers to ensure the system is intuitive and fulfils the purpose that it intends to. In addition to this, we secured peers close to us that were also willing to help test our system and ensure it serves its purpose efficiently.

Our user-centred techniques were also applied during the requirements gathering stage - where we carried out card sorting activities and questionnaires among our target audience (Goldsmiths university students) - and used the results to best refine our decisions during development (see appendix for card sorting resources).

The gantt chart we devised to roadmap the creation of Bookit predicted four iterations of the MVP, each growing the system with more features - with each iteration providing functionality that was increasingly non-essential but still desirable (see "Time management"). We discussed this with the allocated testers at the time. We provided them with the current version of the system and a set of tasks to carry out using that version - then collected their opinions on the usability of the system at that point. This feedback would then be considered when continuing development and ensured that we produced a system by the end of the module that was thoroughly tested in all necessary aspects.

Iterative development and Agile

To decide how we were going to approach our project we researched both waterfall and agile methodologies - ultimately deciding to follow more of an agile approach as we believe

it aligned more with our iterative process - as well as our proposed focus on usability and user-centred design.[4][5][6]

Key aspects of agile development that we chose to apply to our project are as follows:

- Iterative approach - start by adding the barebones of a feature then add functionality gradually. Ensure that functionality takes precedence over aesthetics and ensure that you circle back round after completion - perform unit tests and continue until functionality is acceptable; then move onto aesthetics.
    - Pair this with frequent communication with proposed users and stakeholders. Collect feedback on developments and apply this feedback to the product - this ensures that our target audience will actually like Bookit. This ensures that the development progress is transparent to the stakeholders.
- Control over the schedule - focus on communication within the team both in person and online to keep track of progress and ensure that work does not end up getting done twice (or not at all). Document progress on Discord and changes on Github, and ensure the entire team knows about issues and bugs so that transparency is maintained throughout development.

Group management

Group roles

From proposal stage through to our developmental phase, group roles and task allocation have been at the forefront of many group discussions. We have discussed our availability, experience, skills and interests with the primary concern being that every member of the team felt accounted for, accommodated and challenged to excel in the tasks they would take on when developing Bookit.

Most importantly these conversations allowed us to address an important fact. We are a diverse group. Some members have years of experience, and for others this project serves as the first genuine test and challenge of the skills we have garnered from our time at Goldsmiths. This meant to us that defining group roles would quickly become nonsensical

---

[4] Palmquist, M. Steven, et al. "Parallel worlds: Agile and waterfall differences and similarities." Software Engineering Institute 1.1 (2013): 1-3.

[5] Andrei, Bogdan-Alexandru, et al. "A study on using waterfall and agile methods in software project management." Journal of Information Systems & Operations Management (2019): 125-135. P126; 129-130

[6] Balaji, Sundramoorthy, and M. Sundararajan Murugaiyan. "Waterfall vs. V-Model vs. Agile: A comparative study on SDLC." International Journal of Information Technology and Business Management 2.1 (2012): 26-30.

and borderline unfair. This is because learning is not linear, but also because nobody should feel burdened with the majority of the tasks

Instead of defining group roles we aim to allow for the contrasting experiences and skills in our team by giving each other adjacency over the tasks we take on. Our hope is that this allows us to scale and evolve group roles as we work and as the type of workload changes.

.

Time management

In the first semester (i.e., the first half of this project); we planned the majority of the project: creating high fidelity wireframes that would correspond with R4. Alongside R4 we also planned three other revisions to the MVP that we could produce in sequence before our final, aspirational deliverable in R4. For developing that model, we came up with the following GANTT charts to figure out what we were going to do when it came to the development stage of the project:

Gantt chart for developing the MVP model

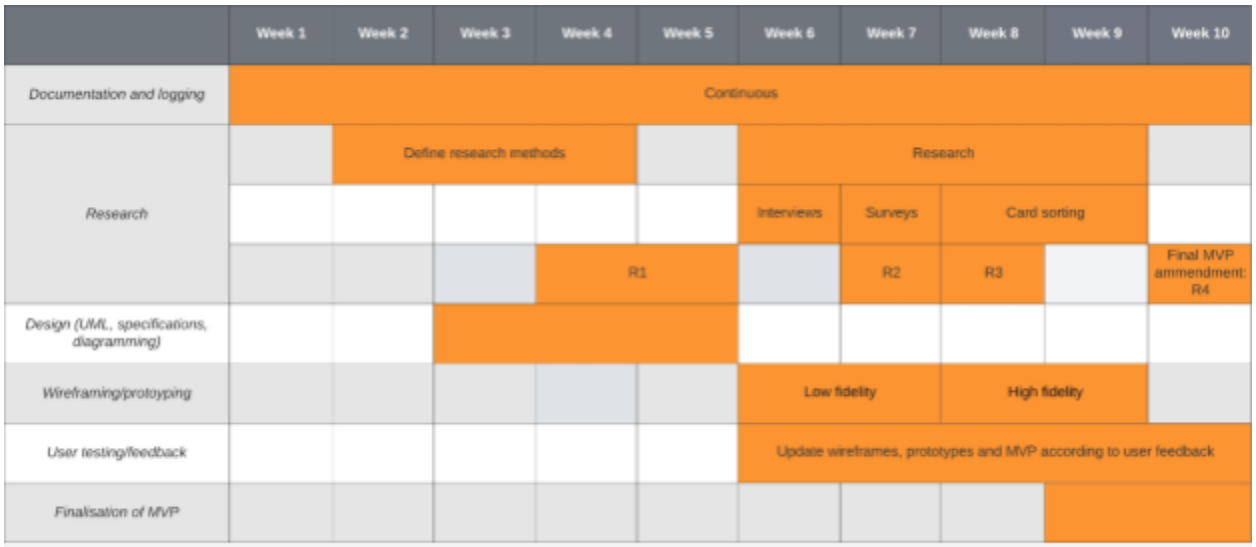
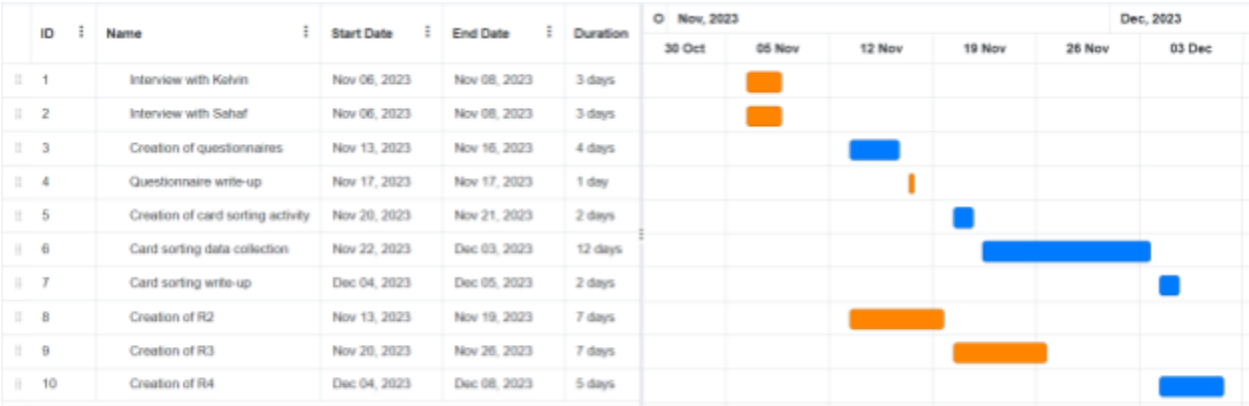


Figure 3: Gantt chart for developing the MVP model

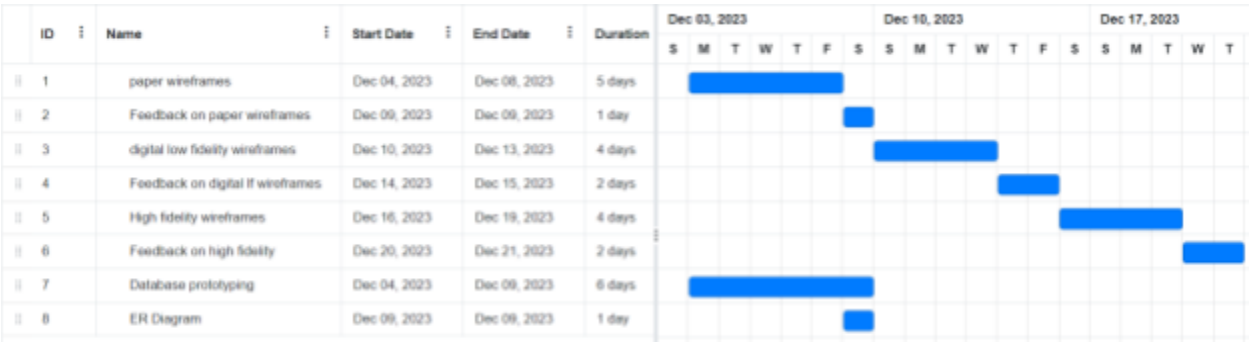
Gantt chart for requirements gathering



ID	Name	Start Date	End Date	Duration
1	Interview with Kelvin	Nov 06, 2023	Nov 08, 2023	3 days
2	Interview with Sahaf	Nov 06, 2023	Nov 08, 2023	3 days
3	Creation of questionnaires	Nov 13, 2023	Nov 16, 2023	4 days
4	Questionnaire write-up	Nov 17, 2023	Nov 17, 2023	1 day
5	Creation of card sorting activity	Nov 20, 2023	Nov 21, 2023	2 days
6	Card sorting data collection	Nov 22, 2023	Dec 03, 2023	12 days
7	Card sorting write-up	Dec 04, 2023	Dec 05, 2023	2 days
8	Creation of R2	Nov 13, 2023	Nov 19, 2023	7 days
9	Creation of R3	Nov 20, 2023	Nov 26, 2023	7 days
10	Creation of R4	Dec 04, 2023	Dec 08, 2023	5 days

Figure 4: Gantt chart for requirements gathering

Gantt chart for prototyping



ID	Name	Start Date	End Date	Duration
1	paper wireframes	Dec 04, 2023	Dec 08, 2023	5 days
2	Feedback on paper wireframes	Dec 09, 2023	Dec 09, 2023	1 day
3	digital low fidelity wireframes	Dec 10, 2023	Dec 13, 2023	4 days
4	Feedback on digital lf wireframes	Dec 14, 2023	Dec 15, 2023	2 days
5	High fidelity wireframes	Dec 16, 2023	Dec 19, 2023	4 days
6	Feedback on high fidelity	Dec 20, 2023	Dec 21, 2023	2 days
7	Database prototyping	Dec 04, 2023	Dec 09, 2023	6 days
8	ER Diagram	Dec 09, 2023	Dec 09, 2023	1 day

Figure 5: Gantt chart for prototyping

Initial proposed GANTT chart for time management
From these GANTT charts we managed to come up with the previously detailed models for R2 and R4, as well as stripped down versions of these models that we could develop and test prior to these versions named R1 and R3 respectively (these models are displayed in the appendix). To match this proposed structure for development, we developed the following GANTT chart for development in the second half of the project (See below Figure #). It was necessary for us to ensure we incorporated user testing at each revision of the application to ensure Bookit remained usable. Our larger group size of seven individuals also allowed us to allocate tasks to more than one person to ensure the task's completion should a team member not be able to complete the task for whatever reason - this was a management technique that we employed frequently during development.

Amended plan for time management

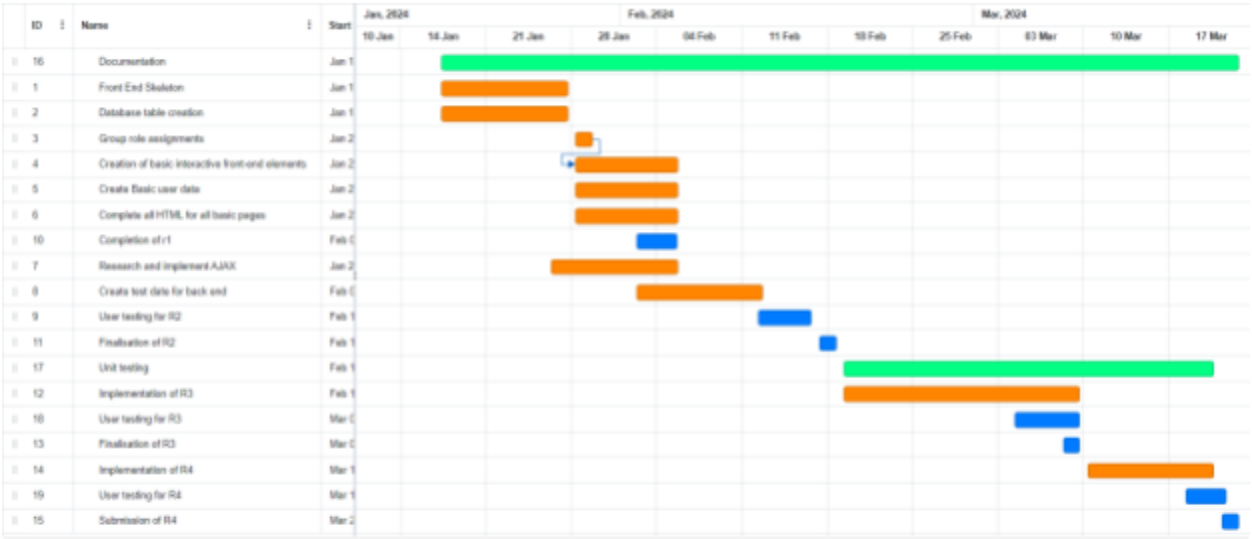


Figure 6: Initial GANTT chart for time management

As we began development of Bookit - we realised that the workload was causing stress amongst our group and reducing productivity. This, paired with the added pressure of work from other modules, meant that it was necessary to reduce our workload as much as possible while retaining our targets. To solve this, we combined R1 and R2 into a revised model for R2, and we did the same for R3 and R4. This meant we were focusing on two revisions of the project as opposed to four, which we decided to name v0 and v1 respectively.

v0 Contained the bare minimum essentials of the project - being a usable interface, a skeleton for the entire web application and the majority of features detailed in the aforementioned R2 model. v0 Also had minimal security consideration, but did not account for all possible security risks as it would have been impractical to test them all at this stage in the project.

v1 Contained the “fleshed out” version of Bookit. It contained all features detailed in R4 as well as comprehensive security measures and a user-friendly and aesthetically pleasing interface.

This report was written largely after development was concluded by using information logged in our group’s private Discord server detailing what work had been completed and when, as well as the report from the Github project we set up after the completion of v0. Furthermore, each week we made sure to document what progress had been made by each group member in a sprint template to ensure that all code could be reported on (see Appendix).

Considering these amendments to the previously displayed time schedule, we decided to revise it as follows with considerations for v0 and v1 of Bookit - with the following, more simple, time schedule:

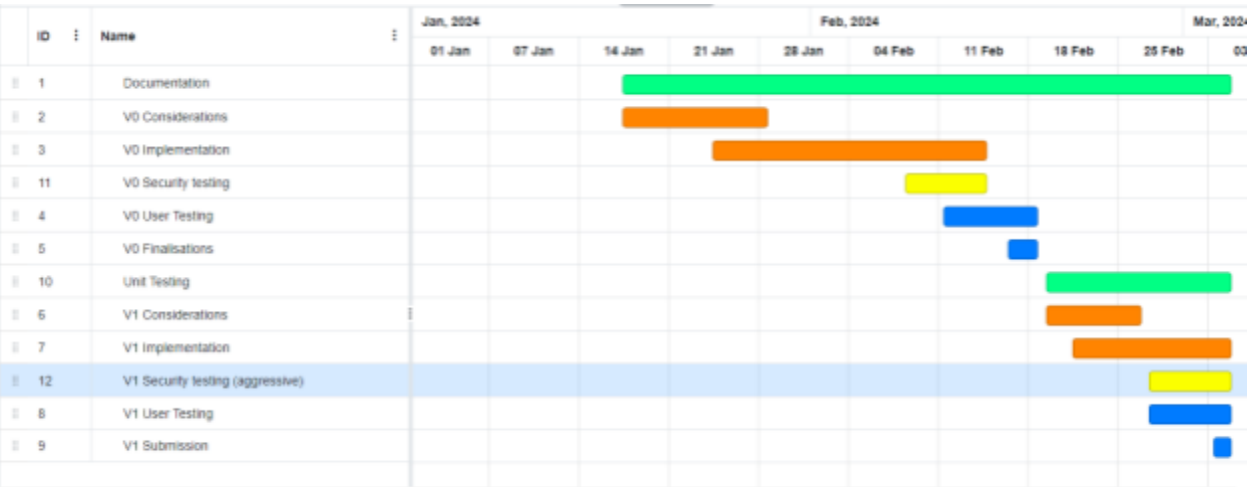


Figure 7: Amended GANTT chart for time management

We used both of these gantt charts in conjunction with one another - using the first one for a more detailed breakdown of our minted process and the second one to ensure we were on track as the project progressed.

Software and technologies used to manage the project

Github

Github is the industry standard for overseeing the development stage of a software project such as this one so it made sense for us to use it during the development stage of our project. Github allowed us to use one repository between the seven of us to synchronise changes across multiple machines. We made sure to clearly document changes with each commit to the repository and ensure we had good version control throughout the project.

After the completion of v0, we also used Github to manage progress, raise issues and log developments to the application through the use of github projects as opposed to sending loads of messages on our Discord server - this not only was more efficient and less time-consuming, but also allowed us to more effectively prioritise tasks due to Github's categorisation functionality allowing us to distinguish between priority bug fixes and "nice to have" enhancements to the application.

Discord

Discord was our primary communication method. We used a private server to converse about progress, problems and solutions. Having a private channel that was available for all members to see was instrumental in allowing our group to effectively and asynchronously work. Furthermore - Discord allows for direct messages between two group members should the entire group not need to see the conversation between them. Specific topics could also be sorted into threads and channels - and important information could be pinned to said channels for easy access (for example, the commands to access the database).

We maintained frequent text contact via discord throughout the project, and when the whole group was needed for a discussion we were able to do group calls via discord as well - and notes taken during these calls could be shared right there in the Discord server.

As briefly mentioned above - we originally handled delegation of tasks and logging progress through Discord messages, but we switched to Github projects after the completion of v0 for the sake of time convenience and more effectively logging our progress - as well as who was working on what.

Planetscale

We decided in the proposal stage of the project that we wanted to use a mysql database, not only due to it being easy to maintain and secure but also because it is the database system in which our group members are the most well-versed. To host our database online we used Planetscale - as it is efficient and includes a free hobby plan that was adequate for our purposes.

Development Environment

During development the application was hosted on the development team's individual computers and connected to the shared Planetscale database.

Production Environment hosting application and database

We decided to host the production version of the application and database on the Goldsmiths Servers using a ubuntu virtual machine.

Google Docs

To collaboratively work on the project report we used Google Docs as it allowed for simultaneous editing both online and offline. This proved to be a good choice as we never had any issues with version differences throughout the entirety of the project.

VSCode

VSCode was our development program of choice, as it is free, lightweight and very customisable with packages. It also allowed for easy connection to Github and allowed us to collaborate remotely in the same space - meaning it was a good idea for us to standardise VSCode as our coding program across our group.

Analysis and research

Initial requirements gathering

In the first half of the project we conducted interviews with the society leaders in our groups to get a general idea of which features to include in our application. This gave us a better idea of what features to include but did not give us the level of detail we needed surrounding what was vital for such an application and what was not. To solve this, we conducted two sets of questionnaires that were handed out to Goldsmiths university students and graduates: first to get data on what these people would expect from a room booking application, and then the second set of questionnaires to further refine this data was handed out specifically to those running university societies to gather opinions on experiences booking rooms at university. The questionnaires reached a wide range of individuals and allowed us to more effectively develop our model for our MVP.

The interviews showed us that we needed to include the following:

- Template risk assessments
- Account system to log level of clearance
- Edit bookings - at which point they can be re-examined by the coordinator

Opinions on customer service from our first set of questionnaires were varied - most opinions lean towards neutral or negative because they either do not need to contact customer service or customer service has not satisfied the user with what they needed. To have positive customer service experiences, users expect quick solutions, assistance and effective communication.

All users find user reviews extremely helpful, rating it 5/5. One user being the exception rating it 4/5.

All users find navigation around booking apps to be neutral or good - rating it either 3/5, 4/5, or 5/5. There have been no answers that are below 3/5.

Most users do not stop using a booking app because of poor functionality and interface; those who do mention that things that stop them from using apps are complex navigation, technical glitches, lack of features, and poor design. Some also stopped using booking apps because there isn't any availability when trying to book something.

We then conducted a card sorting activity to gather an idea about how to group our newly identified features within our proposed application. From this; we saw that it would make the most intuitive sense to place the majority of features in a single booking screen as this was the most popular category for the card sort. Furthermore, we should place a larger portion of our focus on streamlining the navigation process and allowing for easy access of details and information in a user's account.

This activity also showed us that usability is not a direct consideration of our proposed users; so we should still aim for a high level of usability and an easy to use UI for our system, while also ensuring we implement as many useful features as possible.

Performance also seemed to be an important consideration for this system when it comes to the opinions of our proposed users. Therefore, we have decided to make it a priority when it comes to non-functional considerations.

We came to the realisation that the majority of functionality could therefore be handled by a single page/layout that changes in use depending on what the user needs.

The full breakdown for our card sorting results is in the Appendix. Full breakdown on our questionnaire results is given in our project proposal.

Market research

For our project proposal submission we decided to look at a range of alternatives for a room booking system. We ended up conducting an empirical analysis of the following sites:

- Booker[7]
- Schoolbooking[8]
- RoomBookingSystem[9]

Booker is the room and resource booking system used by the University of Cambridge. Booker was developed by the University of Cambridge and in partnership with EventMAP to simplify their previously varied and complicated booking process.[10] We empirically analysed the strengths and weaknesses of Booker and came away with the following considerations:

[7] https://booker.eventmapsolutions.com/
[8] https://schoolbooking.com
[9] https://roombookingsystem.co.uk/
[10] www.eventmapsolutions.com. (n.d.). Developing Booker with the University of Cambridge. [online]

- Specific user roles for level of clearance is instrumental in ensuring the booking process functions as intended for different types of users.
- A filter system for which type of room is desired is a necessity - as this is one of the most popular features that Booker has.
- One of the main complaints with Booker was that there was incorrect information surrounding the capacities and facilities available with specific rooms - to avoid this we must be careful to include correct information on our back end systems.
- Overbooking, or instances where lecture rooms get overcrowded is possible with Booker. We therefore considered holding reservations so multiple concurrent bookings do not occur - a type of “hold for review” system.

schoolbooking.com is a room booking system used by schools. Bookit has a slightly different scope due to its university setting and therefore we included this in our considerations. After analysing schoolbooking.com we came away with the following considerations:

- schoolbooking.com has a host of features that are not necessary for the basic function of a room booking system such as Bookit but are good “quality of life” improvements - such as integration with a calendar app and email notifications for when your booking is coming up. We therefore consider features such as these as aspirational targets for Bookit.
- schoolbooking.com keeps menus and windows to a minimum and keeps the majority of functionality on one page. This also aligns with the data we gathered in our card sorting activity during our requirements gathering process as users would prefer all the functionality to be readily available and not have to bounce around several pages. We aimed to implement this sort of structure for Bookit.

roombookingsystem.co.uk is a similar application to that of schoolbooking.com in that it is catered towards schools. Upon conclusion of our analysis of this application we reinforced our belief that levels of clearance should be logged to a user account, as well as reinforcing our original consideration specified in our original project definition that users should be able to filter rooms by specific room facilities. Furthermore, this application also has a “super user” feature, which is essentially an admin that can edit booking and room information at any point - this is something worth including for Bookit.

We also considered comparing the site traffic for these websites, but upon considering that Bookit was a solution for a problem concerning an inadequate pre-existing system as opposed to filling a gap in a more open market, we did not consider it necessary.

To conclude our market research, we realised that stakeholders would be different for a university room booking system than they would be for a school room booking system. While the systems would be similar and have similar functionality, conducting frequent user tests and ensuring we follow our agile and user-centred development practices would

be very important to ensure we satisfy the needs of our stakeholders and potential users. The breakdown for these considerations is given below.

Stakeholders

A stakeholder for Bookit is anyone with an interest or concern in the application and its development and use.[11] Considering this, as well as the findings of Nothaft and Wiesche - we concluded that the following are the main stakeholder categories for Bookit - and we have analysed their concerns and interests in the application.[12]

Developers

Developers are anyone directly involved with the production of Bookit itself as a system - this includes all of us as group members (see Preface). This also extends to maintenance of the system of which the responsibility also falls to our group members.

To ensure we kept a good workflow and kept our workloads per member to a reasonable level, we regulated the MVP model in the proposal stage of the project to ensure we avoided feature creep. Furthermore, we made a harsh analysis of our scope halfway through the project development stage to ensure the workload would be manageable for all group members. We also planned to have the application be as maintainable as possible rom within the application itself (e.g., giving administrators access to adding rooms, deleting rooms, removing/creating new bookings, etc.) - which would therefore eliminate the need for further time and monetary cost considerations when it comes to figuring out who will maintain these aspects of our application - as the users of the application would be able to do so.

Users

Users include anyone who uses Bookit outside of development.

Our intended users, as mentioned previously, include Goldsmiths Society leaders, lecturers, and current room booking coordinators and administrators. This user base is limited, but consistent. Therefore we will need to make the application as accessible and user-friendly as possible for these people (considerations for this are detailed in the R4 model).

---

[11] Dragos, Paul. "The impact of stakeholders in agile software development." The Annals of the University of Oradea. Economic Sciences 30.2nd (2021). p. 356

[12] Huck-Fries, Veronika, Francisca Nothaft, and Manuel Wiesche. "Investigating the Role of Stakeholders in Agile Information Systems Development Projects: A Mixed Methods Approach." (2021).

Society leaders

Society leaders will want something that will easily get their booking application through to a coordinator with the facilities they want. Because of this - the app will need to be easy to navigate and provide this functionality clearly.

Lecturers/staff

Lecturers will wish to use Bookit to schedule in spaces for their educational events - as a result the process should require minimal steps - as this is a task that needs to be performed frequently.

Coordinators

Coordinators will want to use the application to approve or deny applicants for bookings efficiently. Therefore, this process should also have minimal steps with minimal inputs - and both booking applications and the options that coordinators have available for said applications should be clearly presented and easy to understand.

Administrators

Administrators will want a larger reign of access over the data in the application. Therefore - Bookit should allow them to amend, add to and delete the rooms, bookings, and application data all within the app while ensuring that they cannot make any changes that break other portions of Bookit.

Testers

Testers are anyone who aids in the production of Bookit by testing our application and providing feedback on the functionality and user experience. The testers will include us, as group members for unit tests - as well as two of our group members Kelvin and Sahaf for their input as society leaders.

We also plan to run automated security tests via OWASP to ensure the security of our application.

Suppliers

Suppliers are parties that provide services that allow for the development and maintenance of Bookit - while not including developers like us as group members.

Suppliers therefore include parties such as the GOldsmiths servers (Igor) - for providing a hosting platform for the Bookit prototype, Planetscale for providing a database hosting platform - Node.js for providing a free middleware service and the various free node modules that are upheld and maintained voluntarily by developers on GitHub.

If we were to deploy this application on a different web hosting platform in the future in the event that use of Bookit becomes accepted at Goldsmiths - then that hosting platform, as well as Goldsmiths as a university, would also be considered supplier stakeholders.

Monetary cost assumptions

When considering the financial implications of creating a web application like Bookit it is vital to consider not only the development of the application itself, but also the potential costs of the prototype launch - as well as the potential costs of prolonged hosting and maintenance. Because of this - we broke down our assumptions for the monetary costs of these three considerations below.

Development of the application

We do not plan to spend any money for the development of Bookit as the intention is to only reach a prototype that can be presented as a submission.

- All of the services and applications used (See “Software and technologies used to manage the project”) are free.
- We aim to host this prototype on the Goldsmiths servers provided to us along with Planetscale for the back end. Both of which incur no further charges.
- Time is not paid - all group members are using their own time willingly and voluntarily for the project.

Prototype launch

- Users are not expected to pay to use Bookit - therefore there are no considerations when it comes to revenue.
- We can use Planetscale or another free hosting platform like Igor for our database to showcase our prototype as we are unlikely to be handling large amounts of data at this stage.
- When considering development costs - as mentioned previously - all our group members are student volunteers - and therefore there are no cost incursions at this stage of deployment.

Upholding Bookit long term

- As this is intended to be an application used at Goldsmiths - a university - we have no intentions of gaining revenue from this project via marketing/advertisements. However - a subscription fee may be considered if the application is deemed useful enough - which comes with considerations for revenue for those involved in the development of Bookit.

- We would also have to allocate a person/employee to handle maintenance of the back end as well as correspondence and enquiries. This would incur a annual charge of £24459 per annum according to TotalJobs average salary for a web support role:[13]

Figure 8: Graphical representation of the average salary for Web Support jobs.

- As the level of data grows on Bookit we would also have to consider the costs of hosting this data. However - this would likely be hosted by Goldsmiths should the use of Bookit be accepted.

General analysis (STEEPLE)

A STEEPLE analysis is used to measure the impact of an application such as ours on various aspects of society - and according to James Cadle (et al) - it is vital to ensure we create an application that will benefit our intended audience - our considerations for these impacts are given below.[14]

Societal

While a university room booking system may not seem like it has potential for greater societal impact - improving the process for a system such as this would encourage more extra-curricular events to take place and therefore enrich various communities throughout the university - possibly leading to a greater draw for applicants to Goldsmiths and improving the community and societal aspects of Goldsmiths and the surrounding area. reinforces these assumptions - stating that demand for a greater user experience is higher than ever since the introduction of more E-learning and online platforms at Universities -

[13] Totaljobs. "Jobs | UK Job Search | Find Your Perfect Job - Totaljobs." *Totaljobs.com*, 2020, www.totaljobs.com/.

[14] Cadle, James, Debra Paul, and Paul Turner. Business analysis techniques. British Informatics Society Limited, 2010. p. 3

and therefore our focus on UX is justified when considering societal impact and how more of University is being moved online in modern times.[15]

Technological

As Bookit is a proposed solution to an unoptimised system that is currently in place, we aim to make a positive technological impact on the current room booking process here at Goldsmiths by streamlining and simplifying it.

At Goldsmiths there is a laptop rental service as well as a large amount of Mac computers for students to use at no extra cost - as well as most students having access to their own personal computers. This means that any society leader will be able to access our application. To create a more inclusive technological space; we could consider creating a layout for mobile devices as they are also in widespread use at Goldsmiths.

We will also have to consider making the application able to gel well with current Goldsmiths systems that are in place. This means keeping the risk assessment information comprehensive as it is currently, as well as ensuring that all aspects of the current room booking system are included in Bookit.

Economic

While the economic impact of Bookit is likely to be minimal - we aim to produce an app that will save the staff that coordinate room bookings time - and therefore we can account for an economic impact that reduces costs - time especially - for these staff when it comes to reviewing, handling and denying booking applications made by society leaders through the current Google Forms setup. Elfriede Dustin (et al) claims that automation within processes is one of the best ways to save time and money - this is with regards to the software development process but we feel this can be applied to the room booking process at Goldsmiths as well.[16]

Environmental

As Bookit is a web application which requires computers to access and servers to host - it will have a decent carbon emissions level as with any web application project. In the grand scheme of things, however, the footprint of our application will be relatively small. We will aim to use free services which do not have poor environmental track records as part of our considerations for hosting our application. According to Yoast - a web page with 10000 monthly visitors produces 20 KG of CO2 a year. We do not anticipate that our application

---

[15] Nguyen, Thao. "Improving students' UX in online learning platform: Case study: LAB faculty of business and hospitality management." (2020). p. 43

[16] Dustin, Elfriede, Thom Garrett, and Bernie Gauf. Implementing automated software testing: How to save time and lower costs while raising quality. Pearson Education, 2009. p. 21

will exceed that - at least in its current state, and therefore we concluded that we could make further environmental considerations should Bookit need to be scaled up.[17]

Political

The responsibility of handling booking applications will fall to those allocated by the Goldsmiths staff - it is important to not let political bias determine what bookings get chosen from certain people of varying political alignments.

Legal

Security testing is of utmost importance to ensure that sensitive user data is kept safe. We will also need to comply with data protection laws - keeping all user data under a reasonable level of encryption as well as protecting our application from basic attacks such as SQL injection.[18] This is to make sure that sensitive data such as emails, passwords and what room bookings are made by which users are kept private only to the intended parties.[19]

Ethical

Our ethical considerations are centred mainly around accessibility. We need to ensure the media we use to represent the rooms at Goldsmiths does not contain any offensive or culturally/ethnically insensitive material - and we also need to make sure our UX design is easily accessible to neurodivergent and dyslexic users. This means avoiding harsh colours and other potentially unwelcoming design choices that would potentially disorient certain users. This also means choosing a legible font of adequate size to ensure we accommodate dyslexic users.[20]

Research conclusion

The STEEPLE analysis in conjunction with our market research - alongside our findings from our requirements gathering process - has given us insight into how we could use these considerations to impact our development of Bookit. We believe that our research into the potential impacts of such an application as well as our considerations for how we can create the most user-friendly and comprehensive solution means we were well equipped to develop the application with accessibility, user experience and our monetary cost

---

[17] Toonen, Edwin. “The Carbon Footprint of Your Website and How to Reduce It.” Yoast, 19 Apr. 2023, yoast.com/carbon-footprint-of-website/.

[18] Spindler, Gerald, and Philipp Schmechel. "Personal data and encryption in the European general data protection regulation." J. Intell. Prop. Info. Tech. & Elec. Com. L. 7 (2016): p. 163.

[19] Sun, Yunchuan, et al. "Data security and privacy in cloud computing." International Journal of Distributed Sensor Networks 10.7 (2014): 190903. Section 5.

[20] British Dyslexia Association (2018). *Dyslexia friendly style guide*. [online] British Dyslexia Association

assumptions in mind - particularly with how these considerations relate to the needs and interests of our aforementioned stakeholders.

We believe that through this research process we have ensured we created a system that addresses the shortcomings of the current room booking system at Goldsmiths while also maximising inclusivity and maintainability without a huge financial impact.

By incorporating our research findings into Bookit we intended to create a room booking system that solves our originally defined problem while also conforming to broader societal values, benefitting the Goldsmiths and general community in unison.

Application requirements

Functional requirements

System requirements

As a web application - Bookit is responsive to all commonly used screen sizes for desktop/laptop (1920×1080, 1366×768, 1280×1024, 1024×768).[21] Should we have had time - we could have worked on developing a layout that can respond to other device screen sizes; such as mobiles and tablets. Furthermore, the nature of Bookit as a web application means it did not require any further hardware to run besides the necessary hardware to run the browser itself - which should be standard across any normal computer system with a functional CPU, GPU and RAM setup. By extension, we aimed to make Bookit compatible with commonly used browsers - such as Chrome, Firefox, Edge, and Opera.[22]

We will continually be fetching data from our backend to display to the user, but the data will not ever be exceptionally large and therefore we predict that the load times for Bookit will never be any longer than a few seconds at maximum - we also aimed to reduce load times by using AJAX to dynamically update page content so that very slow internet connections did not experience too much of a drop off in user experience.

User requirements

We developed a list of functional requirements for each type of user to illustrate how they should be able to utilise the application.

[21] DEVARAJ, KIRUTHIKA . "Common Screen Resolutions | What Are They & How to Test?" *Testsigma Blog*, 25 Oct. 2023

[22] "Most Popular Web Browsers in 2021 [Jul '21 Update] | Oberlo." *Www.oberlo.com*, 2023, www.oberlo.com/statistics/browser-market-share.

All user requirements

Functionality available to ALL USERS	
Log In	Log into account using a basic username and password system - done with text entry fields.
Create a booking	Click on a button to create a new booking
Select a date and time	- Use a calendar system to select the day on which the booking will take place. - Highlight the desired time for the booking using a drop-down menu. - Rooms that are unavailable will be removed from the list of selectable rooms to book.
Select room	- Rooms that are available can be clicked on to select them.
Search for rooms	- The user can select the type of room with the correct facilities they wish to have from a drop-down menu. - Rooms which do not have all the facilities specified will not be present in the selection list.
Confirm booking	The booking can be confirmed by clicking a button prior to the time of the booking - if confirmed the booking remains throughout the duration - otherwise the room is marked as vacant again.
View bookings	The user can see a graphical representation of which rooms are available at any selected time - occupied rooms will be removed from the list of bookable rooms at that time.
Cancel booking	Users can select a booking they have made previously and mark the room as vacant - indicating they no longer wish to use the booking they had previously made.
Edit booking	Users can edit a booking they have made previously and amend the data. - If a user edits a booking - any confirmation is lost and the booking re-enters a state of awaiting approval by a coordinator/admin.
Log out	Log out of account and stop accessing app.

Figure 9: Functional Requirements for ALL USERS

Society leader requirements

Functionality available to SOCIETY LEADERS/LECTURERS	
Template risk assessment.	- Society leaders/lecturers will be presented with a series of text fields when they wish to make a booking - this will allow them to input relevant data for a risk assessment when it comes to booking their desired activity. - The UI will consist of a template that aligns with the correct safety protocols for risk assessments when planning societal/educational events at Goldsmiths.

Figure 10: Functional Requirements for SOCIETY LEADERS

Admin requirements

Functionality available to ADMINS	
Access and amend room data	Coordinators will have full access to the room data - where the given facilities can be updated/changed according to the current situation.
Add rooms	Admins can access a menu to create a new room entry - with a name, capacity, type of room/facilities available and also have the functionality to upload an accompanying picture.

Coordinator requirements

Functionality available to COORDINATORS	
View list of booking requests from society leaders	Coordinators will be able to view a list of bookings that have been created/requested by society leaders alongside their attached risk assessments. They will be presented with information regarding which room, when it is being requested for, by what Society and which Society representative made the booking.
Approve/reject risk assessment	

Confirm/deny bookings	Coordinators can click a button to confirm or deny a booking - at which point the Society leader is notified accordingly via email.

Figure 11: Functional Requirements for COORDINATORS

It is worth noting that all aforementioned functionality is available to admins.

Non-functional requirements

Non-functional requirement	**How will we approach this?**
Performance - aiming for fast load times	- Reduce our file count - we should aim to keep our algorithms efficient and use code wherever possible instead of graphics. - When considering dynamic changes - we will make use of EJS and AJAX to effectively and quickly display changes to the user.
Capacity - need to account for a large amount of information on a large number of rooms	- Host all data in a MySQL database to efficiently store and retrieve data on rooms/facilities. - Remove redundancies - only keep the necessary data - such as location, facilities, and room name. - Store data for all required rooms.
Security - certain users should not have certain functionality available to them	- Login system - password and username provide a layer of security for individual users. - Introduce a clearance level for users - only if you have the given clearance level can you edit the room data or decline a requested booking, for example. - Two factor authentication, users should use a separate device to validate their login attempt. - Threats such as SQL injection should be mitigated. - All inputs should be sanitised for both HTML and SQL so that inputs on the front end cannot unintentionally affect the back end data.
Reliability	- As a web-based application, it should be hosted on a stable server with a consistent internet connection.
Scalability - we need the means to add more buildings/rooms in the future	- Storing rooms in a table-database structure based on the building will allow us to create more rooms and space representations using the same code.

Maintainability	- Coordinators can access the room data and add more as necessary - direct access to query and amend the database.
Accessibility - how can we make sure the people that need to use our system can actually use it?	- Images should have alt text. - Colours should account for common forms of colour blindness. - Text should be in a legible font and be large enough so that users with poorer eyesight can still read it.
Data integrity - how will we keep our data consistent and usable?	- Sanitise inputs - room data must follow a strict template (name, location) - and facilities assigned to a room must come from a predetermined selection. - Admins will only be able to input certain data types when amending the room data (e.g., only inputting alphanumeric characters when changing the room name).

Figure 12: Non-functional Requirements

Prototyping and iteration

This section will focus on how we decided on the features that Bookit has.

Concept development

Our initial concept was to have a graphical representation of the buildings at Goldsmiths that the user could traverse and select a room to book. While this concept would be better for newcomers to Goldsmiths that are perhaps unfamiliar with the layout of the buildings, we realised that creating these graphics and the logic to interact with them would be very time inefficient for a project the scale of Bookit. Therefore, we dialled back our concept to assume that the user already has some familiarity with the rooms and buildings at Goldsmiths - by instead providing a list of rooms that are available and allowing the user to filter by building, room type/facilities and capacity for seating.

We knew we wanted to streamline the risk assessment process by putting functionality for risk assessment handling right there in Bookit - but we also had ideas to include a booking confirmation system where users who have had a booking approved would need to confirm the booking on the day the booking was set to take place to let other potential bookers know that the space was certainly in use. This idea came about after a brief conversation with our module leader Dr. Sean McGrath - within which he stated that often educators would book large chunks of time for a certain room over several weeks - leading

other educators to believe that the room was unavailable at a time when in reality it wasn't being used. After commencing development - we decided to remove this feature from the MVP as not only was it a fairly large task, but also our current MVP model does not allow for bulk bookings and therefore eliminates this problem by proxy. If we were to include bulk bookings as an option in the future this potential feature would certainly be something that is considered. For more detail see "Evaluation - Going Forward".

With these considerations in mind, we began developing the R1 MVP model that was presented during the project proposal stage of the module. Which evolved further into the R2 and R4 MVP models you see in the Introduction.

Satisfied with our concept we moved on to prototype our user interface (UI).

UI

Concept

In the beginning we did not devise a model with a set amount of pages as we were unsure what page changes we were going to handle with separate page URLs and what we were going to handle with AJAX - this is because we were unfamiliar with AJAX at the point of the project proposal submission and it required further research. This proved to be the correct decision - as during development we found we could remove entire EJS routes and replace them with AJAX requests to present new information to the user - thus improving the user experience as well as reducing the size, and therefore increasing the performance of Bookit overall.

Considering that we did not go in with a plan on an exact number of pages to create - we decided to focus on a series of use cases for Bookit. The diagram for which can be found in the figure below.

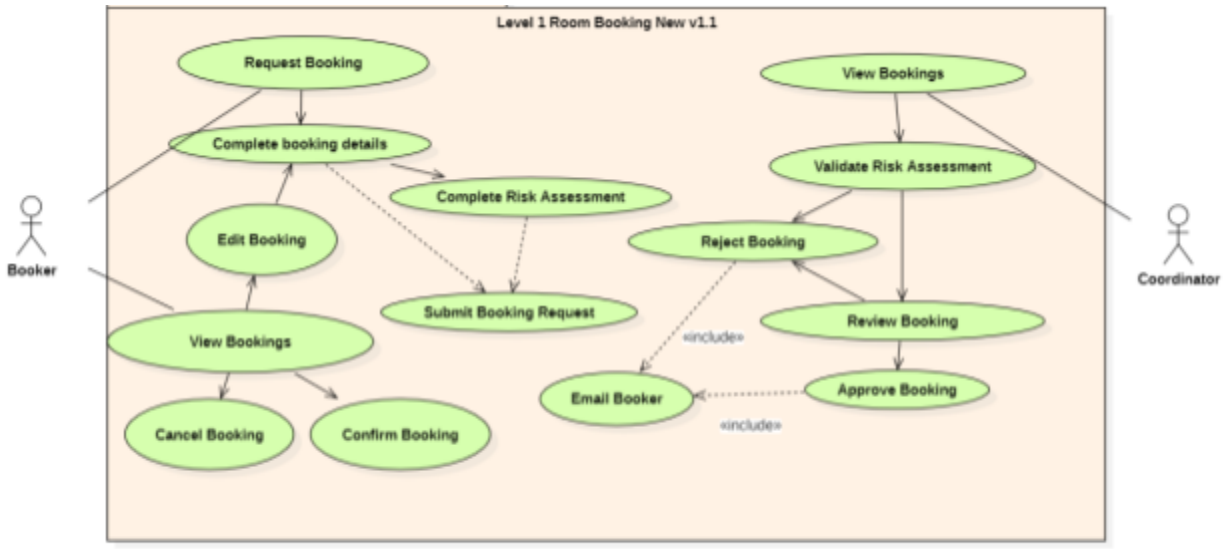


Figure 13: Room Booking Use Case Level 1

As detailed previously - we decided to remove the confirmation system from the series of use cases for our R4 MVP model (which later became V1).

In the project proposal stage of the module - we selected our colour palette and font we would use throughout the site. We ensured all colours we used had AAA contrast ratings and would be suitable for visually impaired users and dyslexic readers.

We decided to include basic pages that should be available at all times in a navbar at the base of the screen. We developed this as part of our project proposal and it remained unchanged throughout the development of the MVP.

Figure 14: Navbar concept

Our header navigation bar initially began as some clickable icons that you could use to go back to the main menu or log out with.

Figure 15: navbar concept

We then wireframed out the remainder of our pages. We created a clickable prototype of 10 pages here: https://app.uizard.io/p/739c5641

The MVP evolved slightly to have 16 pages instead of 13 (as our final result contained pages for credits, bug reports and frequently asked questions - see below) - but the flow of the pages remains the same. A diagrammatic representation of the page flow prototype can be seen below.

Figure 16: High fidelity interface flow

This gave us a good conceptual baseline for us to develop our user interface. Further information on the individual pages and how we iteratively evolved them can be seen below.

V0

The V0 footer navbar emulated the one presented in the concept wireframe.

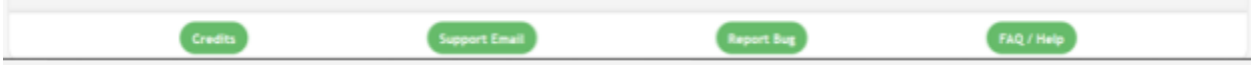


Figure 17: V0 footer navbar

The V0 header navbar also emulated the one presented in the concept wireframe, but we opted for use of more words as opposed to icons to guarantee clarity.

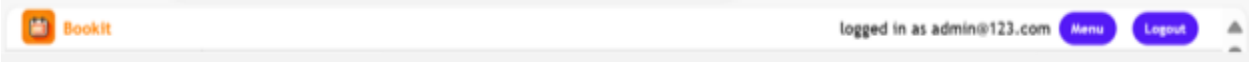


Figure 18: V0 header navbar

We felt this was a good baseline. Changes made in V1 were down to feedback given by our user tests after the production of V0.

V1

After the production of V0 - users asked for a dark mode in the unit tests as the light mode was harsh on the eyes in low light areas. We included this site-wide as part of the header navigation bar as a tick box to turn dark mode on and off. More information on dark mode can be found in “Evaluation - User Experience - Dark Mode”.

Figure 19: V1 header navbar

As you can see from the above figure - we also included the “Order by” buttons for room and booking lists on the header navbar for certain pages. Our rationale for this design decision can be found on the iteration analysis for those pages below (See “Rooms List” below).

The footer navbar remained the same from V0 to V1 - but in V0 there was no bug reporting page, so by adding this page in V1 the navbar gained full functionality.

Going forward

While we don’t anticipate that the navigation of the site will require much of an update in future - the header navbar is now quite crowded and the spacing is not aesthetically pleasing. A CSS overhaul that reduces the size of the buttons and improves the spacing between elements will be necessary to ensure a comfortable layout and better user experience.

Login page

Concept

The high fidelity wireframe for the login page intended to present a comfortable welcome to Bookit.

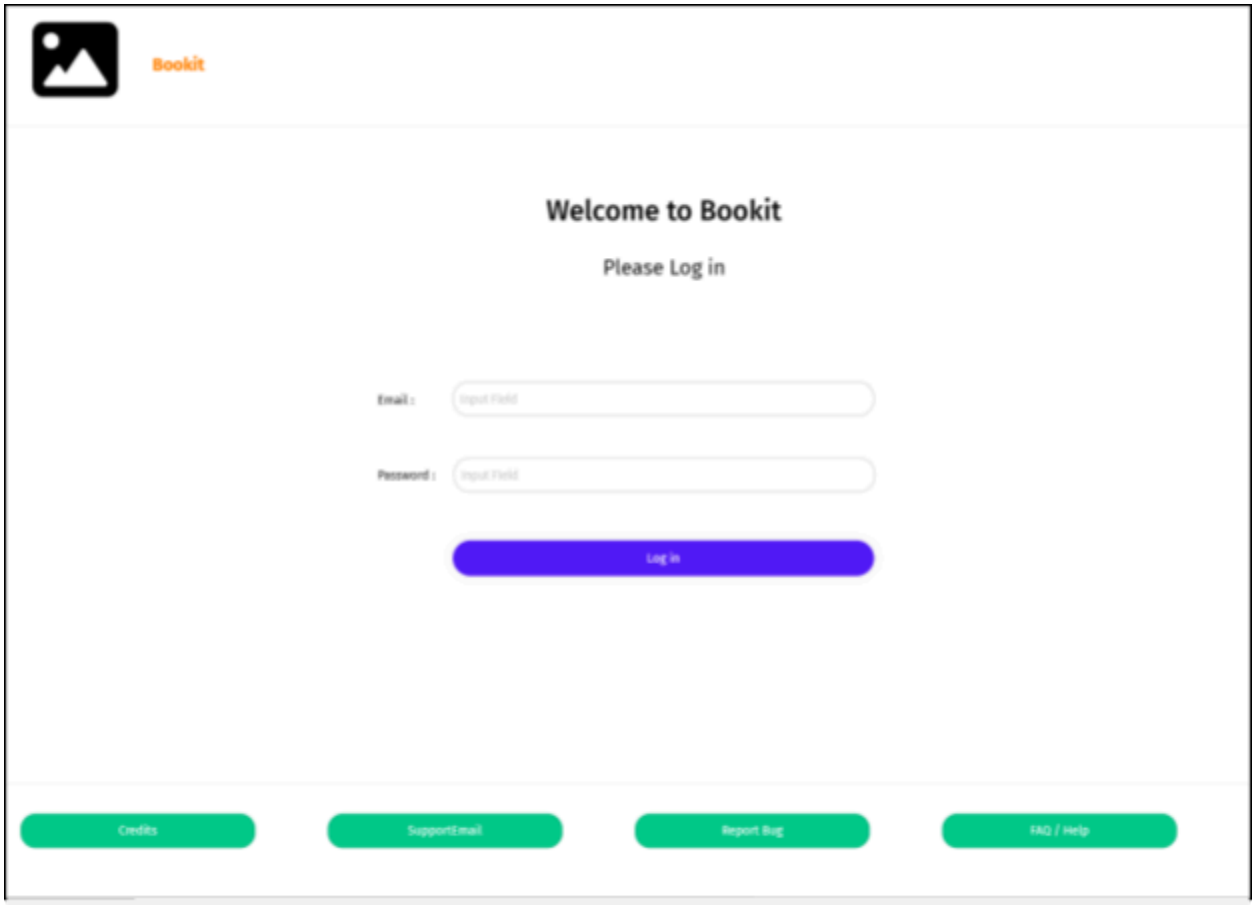


Figure 20: High fidelity UI login

V0

There was little evolution from the concept for the login page - as the login page only really has one function. We included text to show that there was no user currently logged in on the header as well as the newly created Bookit logo. We decided to keep things minimalistic so it was easier for users to understand what functionality was available to them.

While conventional login pages have a register link so that new users can access the site - ours does not include this as the intention is that only admins can register new bookit users, as room bookings should only be made by approved personnel such as registered university society leaders and staff.

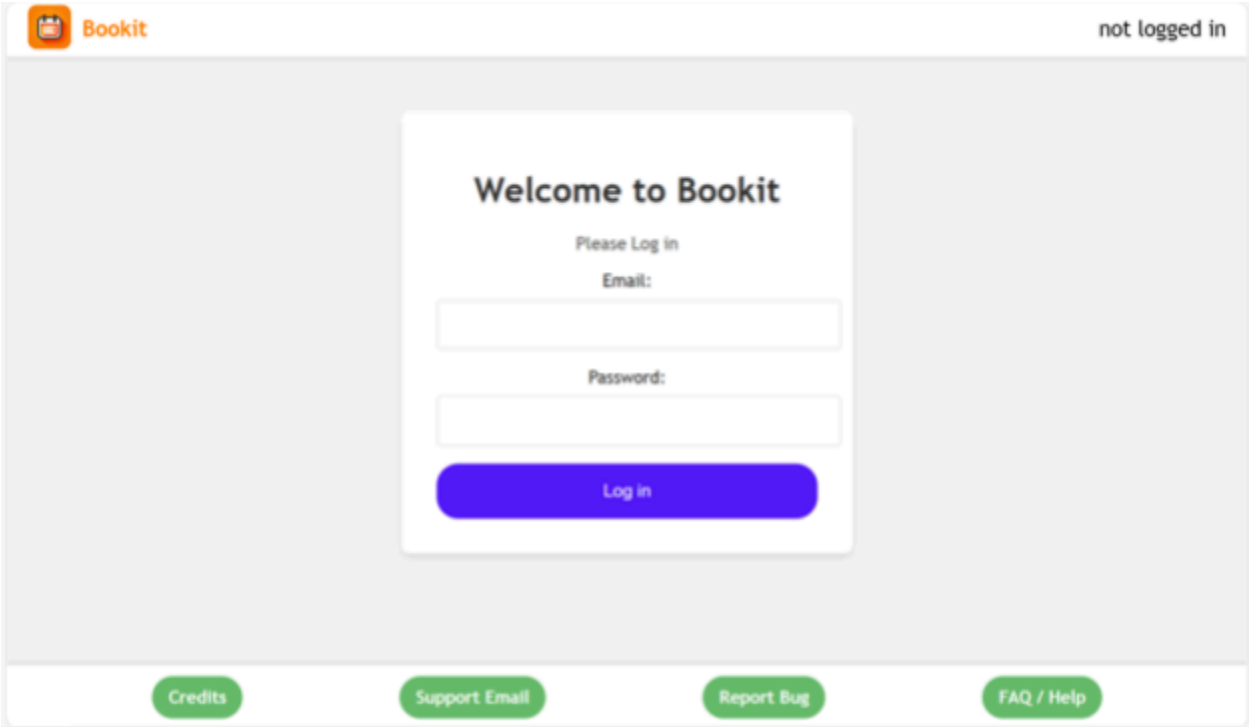


Figure 21: V0 UI login

V1

Aside from the addition of the dark mode checkbox and the option to have dark mode, there were no changes. Seeing as this was a site-wide addition, this evolution can also be applied to all of the below pages as well and will not be listed under the changes for said pages.

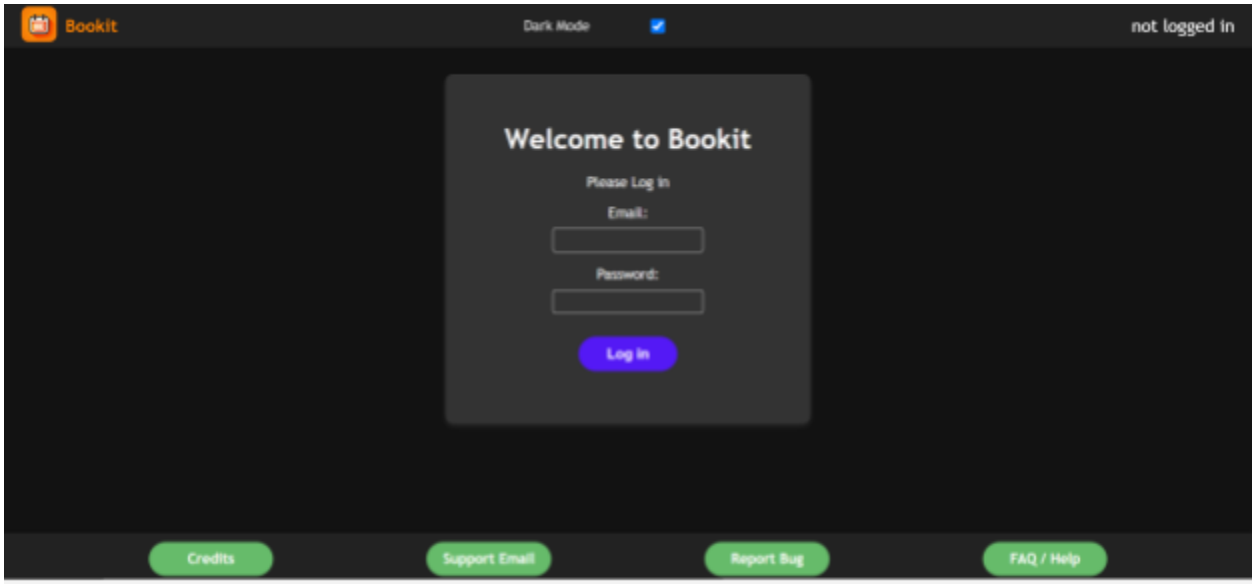


Figure 21: V1 UI login

Going forward

We believe the login page fulfils its purpose and do not plan any updates for it.

Login Success

Concept

This page was intended to be the “main menu” of sorts for bookit. This would display all possible routes that the user could take for all the basic functions bookit offered for that user’s level of clearance. We intended the login success page to somehow confirm to the user that they had logged in as expected and then give them clear options on what to do next. With this in mind, we prototyped the two following high fidelity wireframes to reflect this criteria.

Society leader/lecturer

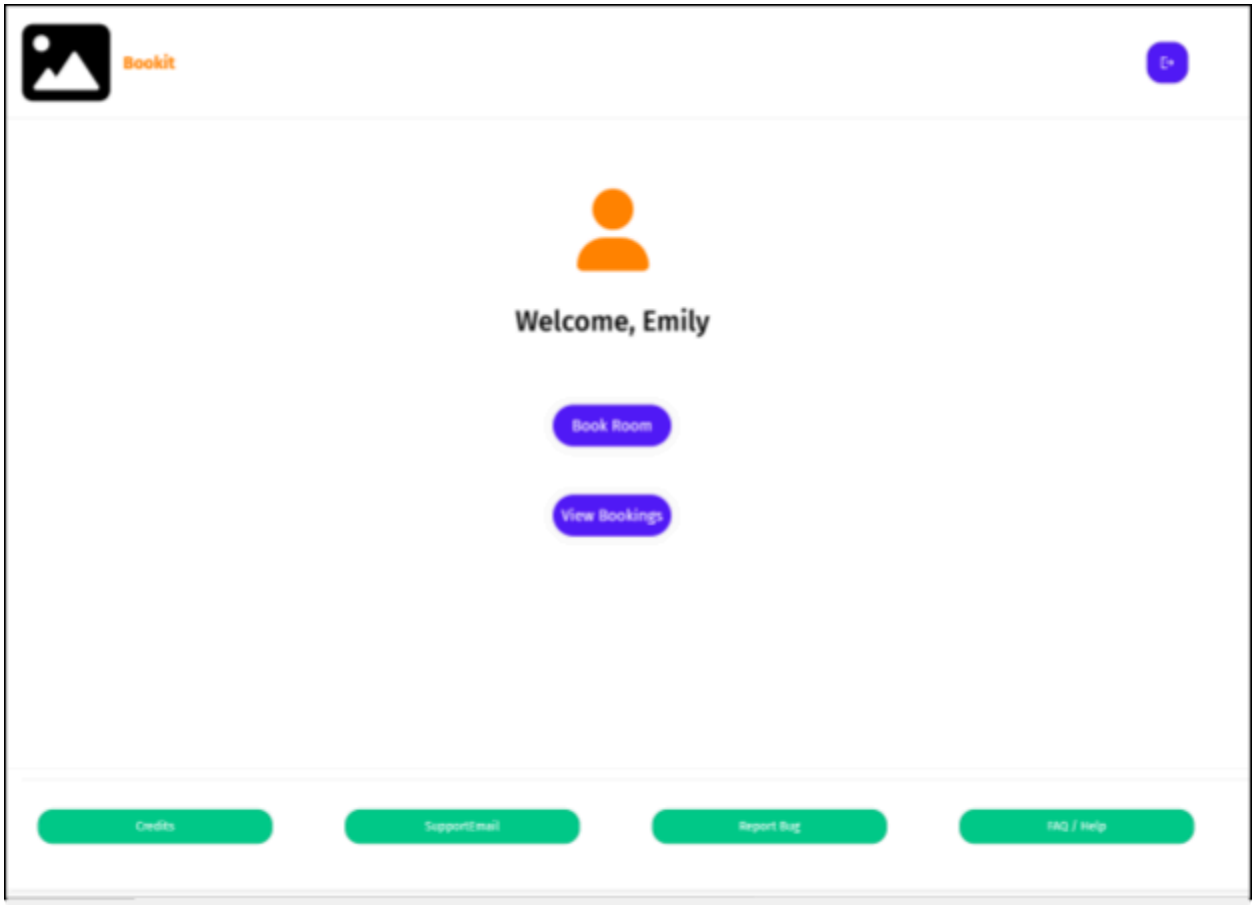


Figure 22: High fidelity UI main menu for Society Leaders and Lecturers

Coordinator

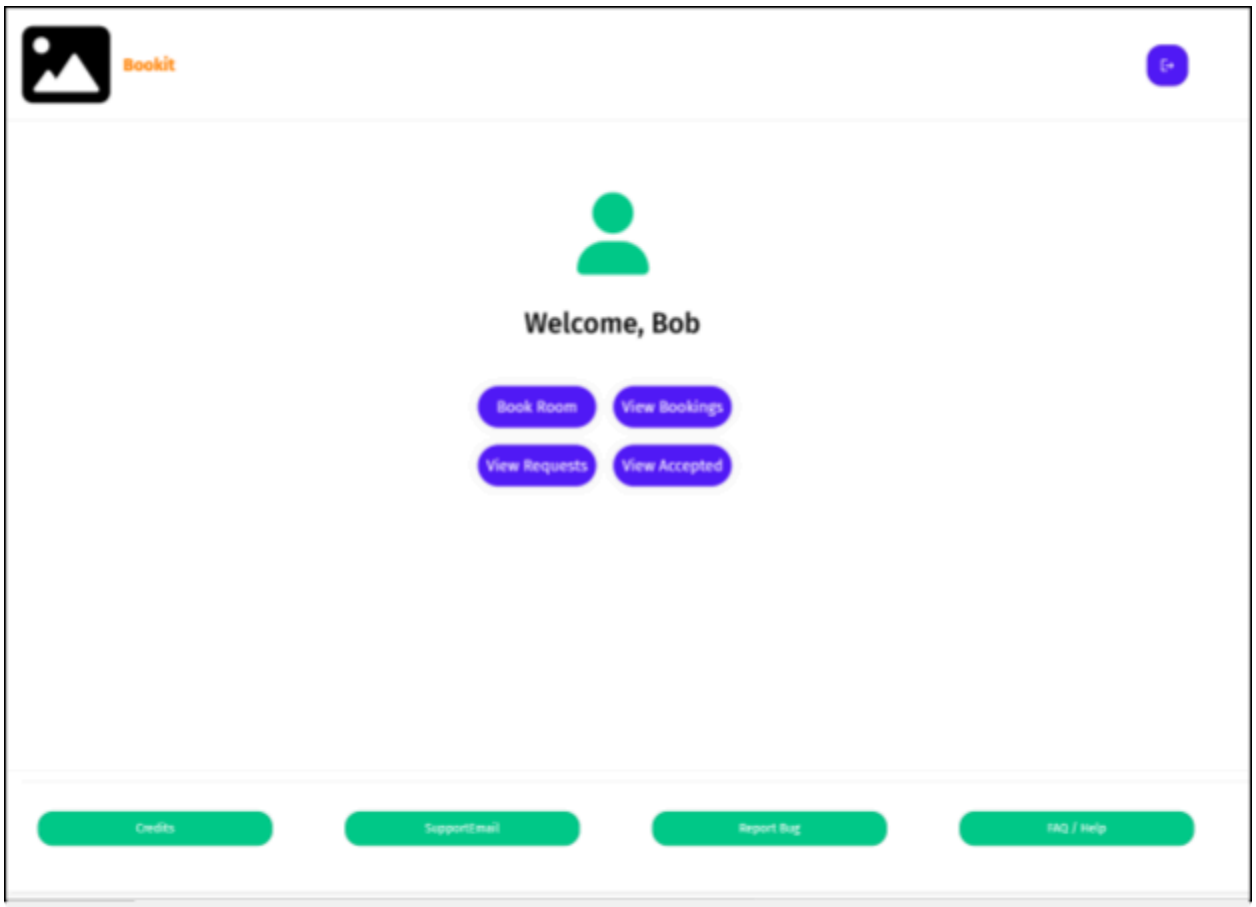


Figure 23: High fidelity UI main menu for coordinators

We did not end up producing a high fidelity wireframe for the admin users at this stage as we did not yet realise the need for a separate menu. This evolution of our concept came during the development of V0, as seen below.

V0

The V0 iteration of the login page featured all of the buttons for standard users and coordinators for their respective login screens - as well as a welcome page for admins as seen below:

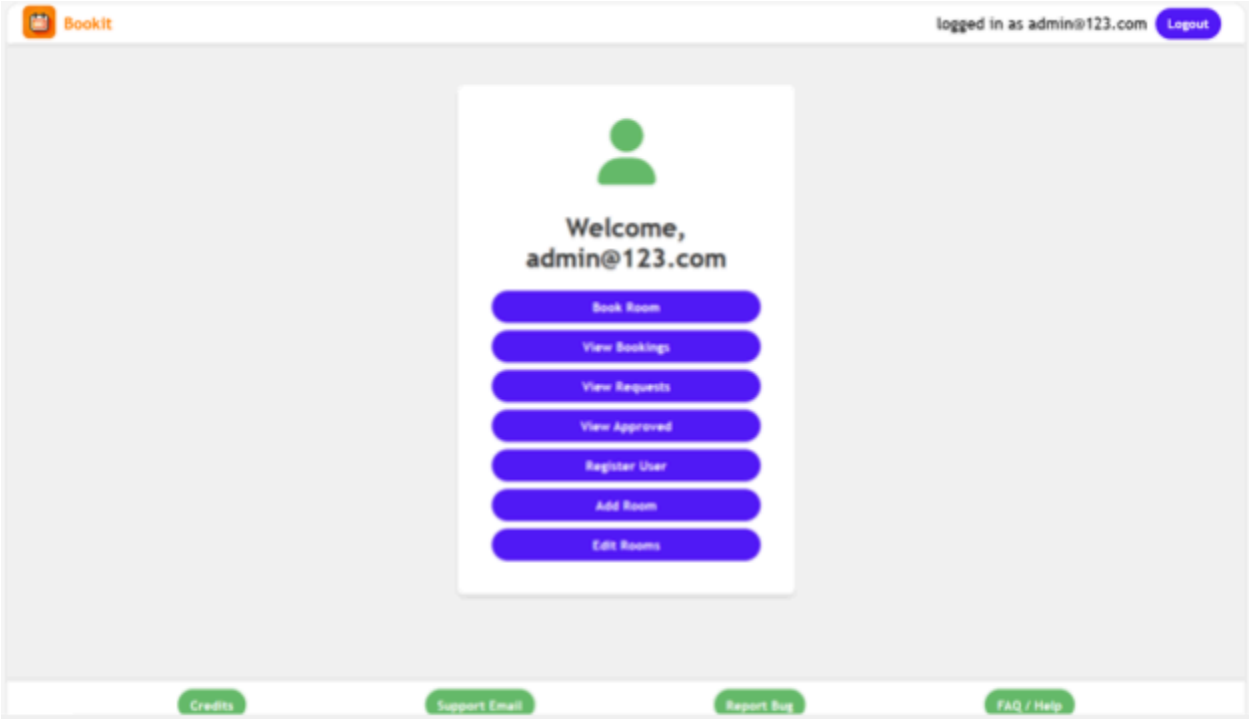


Figure 24: V0 UI main menu

We made it so the profile picture indicator for each of the separate user clearance levels was a distinct and visible colour.

V1

The main complaint we received from our V0 user testing was that the menu for navigation from the login success page was too long and crowded. To rectify this we split the menu into two columns - which our testers preferred unanimously.

Another concern that was raised in our user testing was that there was no way for users to tell when their risk assessments/bookings had been approved besides navigating to the booking on the bookings page and checking its status. We logged this issue in the backlog and considered adding a notifications panel to the login success page - but due to the time constraint of the project at the time we opted for an email-based notification system instead.

The updated UI for the login success screens in V1 can be seen below.

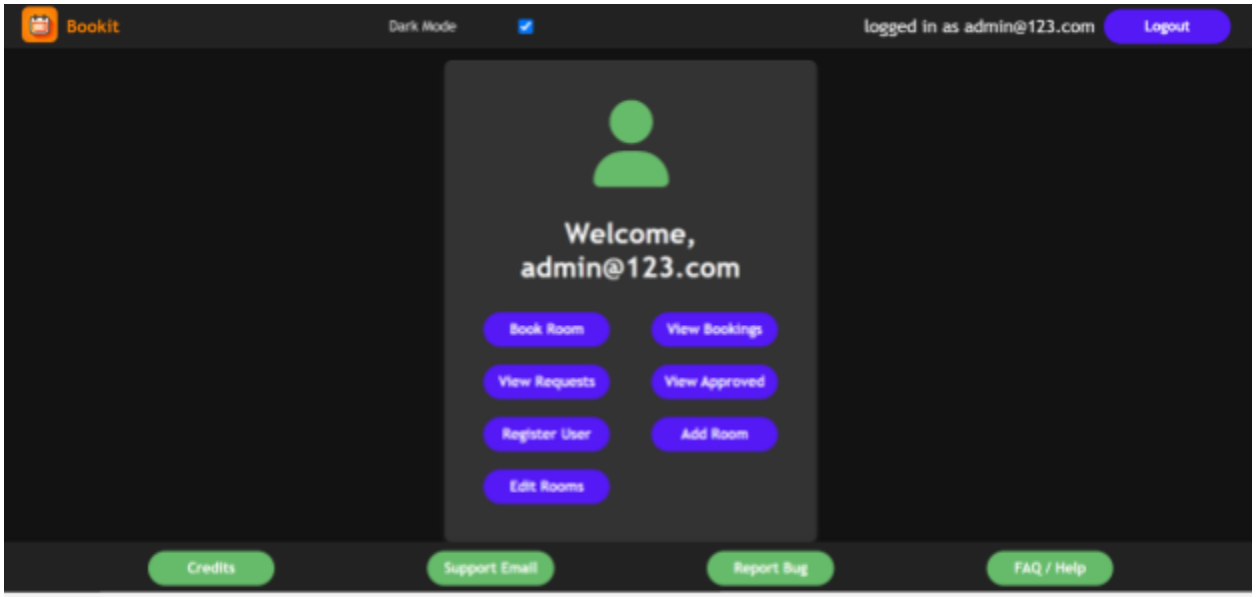


Figure 25: V1 UI main menu for admin

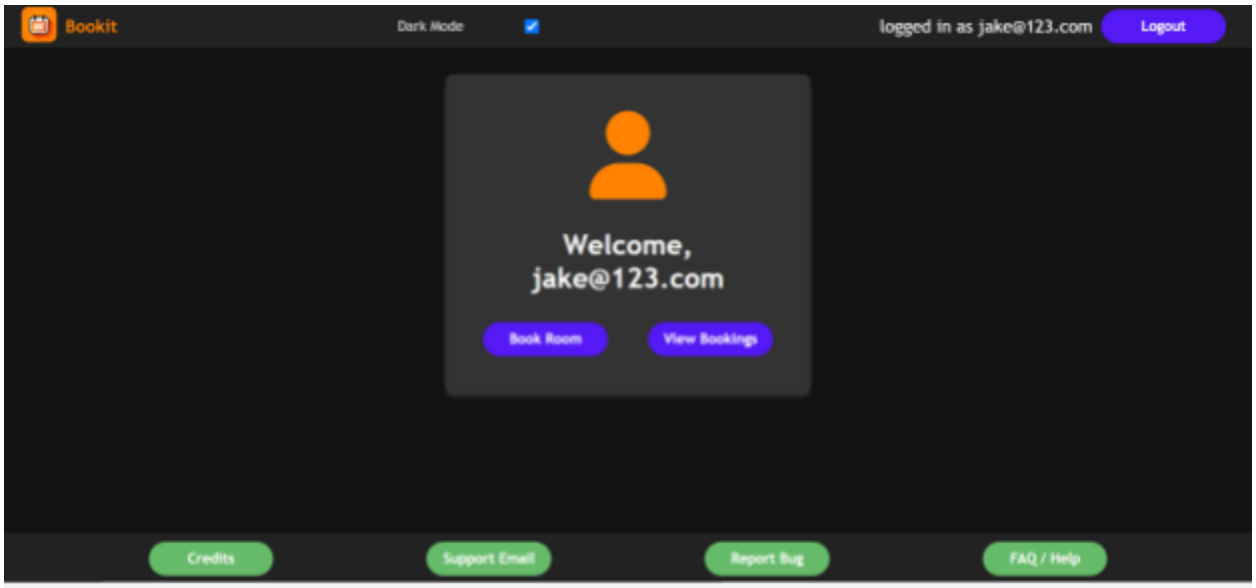


Figure 26: V1 UI main menu for user

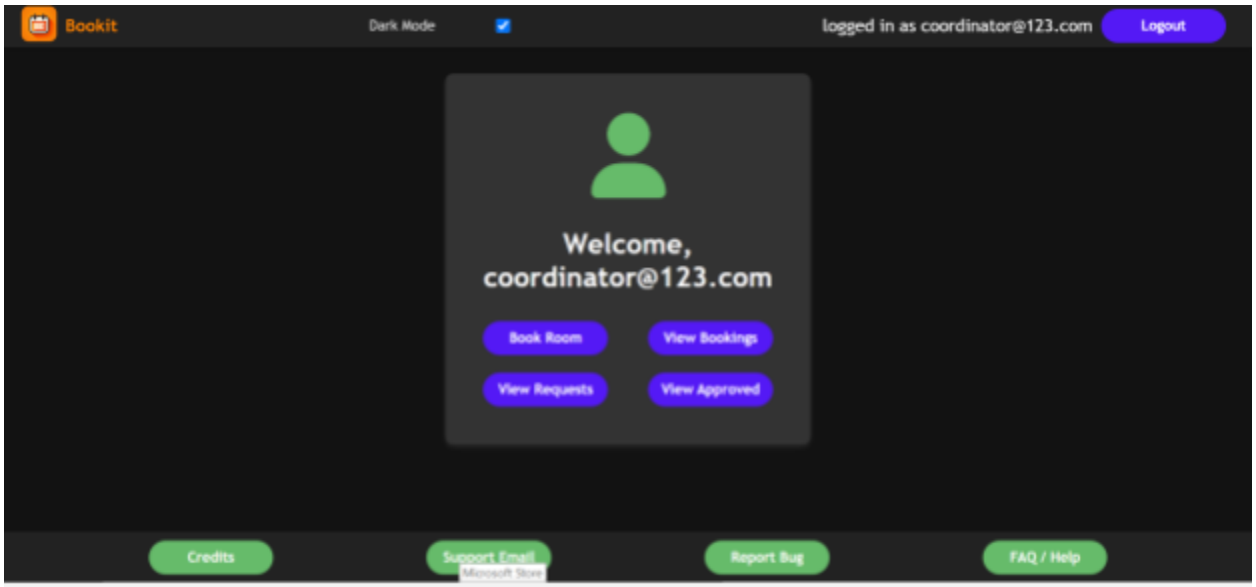


Figure 27: V1 UI main menu for coordinator

Going forward

As aforementioned - a built-in notifications panel on the login success screen would greatly improve the user experience to let users know when they have been approved/rejected for a booking.

If we were to implement user profiles to a greater extent - we could allow for a settings menu on this screen. This could include things such as customisable profile pictures and bios that users can set themselves so that coordinators don't have to just go by the user email to identify who has made a booking. Changes such as profile pictures would show on the login success page instead of the generic profile pictures with flat colours we included in the MVP.

Adding Two- Factor

After V0 had been completed a basic version of 2FA needed to be implemented which would in essence, change the format of the login page. The nature of the page is that it is embedded inside of the login process and is intended to be inaccessible through normal means. The based on V0 feedback, we used the basic template of our v1 login page as a foundation for this page including input box locations.

For the qr code we used this fact that

32. " The minimum physical size of a visible QR code needs to be at least 2 cm by 2 cm (0.8 inches by 0.8 inches) (excluding quiet zone) for it to be readable and scanned successfully." - QR Code Best Practices: Tips & Tricks To Create QR Codes That Engage And Convert"

Amanda Cox February 18, 2023

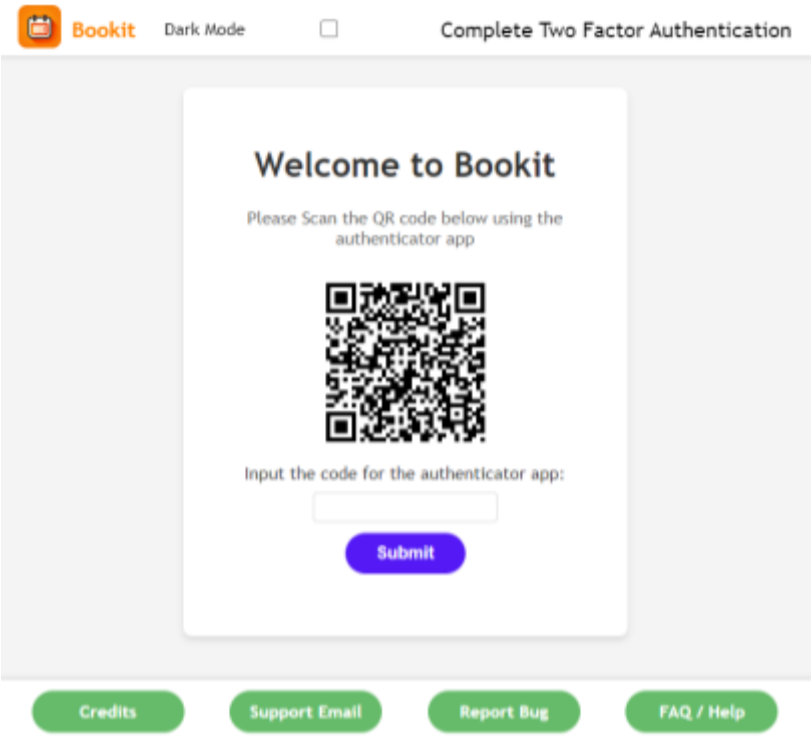


Figure 28: QR code screen

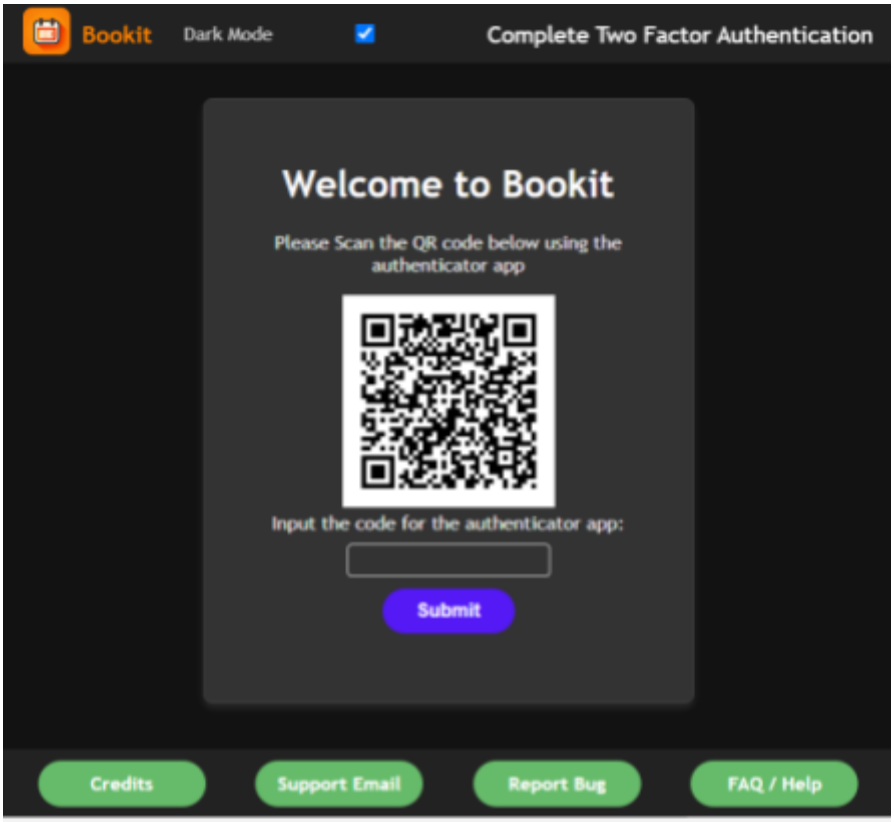


Figure 29: QR code screen in Dark Mode

Our QR code implementation far exceeds this requirement and numerous testing means that our QR codes are always scannable in V1. This is mainly due to the constraints on resizing the viewport.

Rooms List

Concept

Rooms list is the “room booking” page of Bookit. This is where users go to request the bookings they wish to make. It was our intention to ensure that the rooms are presented in a clear and easily understandable manner. This included individual cards for each of the rooms that had:

- A picture of the room
- The type of room and associated facilities
- Room seating capacity
- Room number
- The building the room is located in

We also wanted to make use of a generic filtration system for the rooms, where users could see what rooms were available given a certain date, time, seating capacity, duration and building.

The below high fidelity wireframe is the initial product of these considerations.

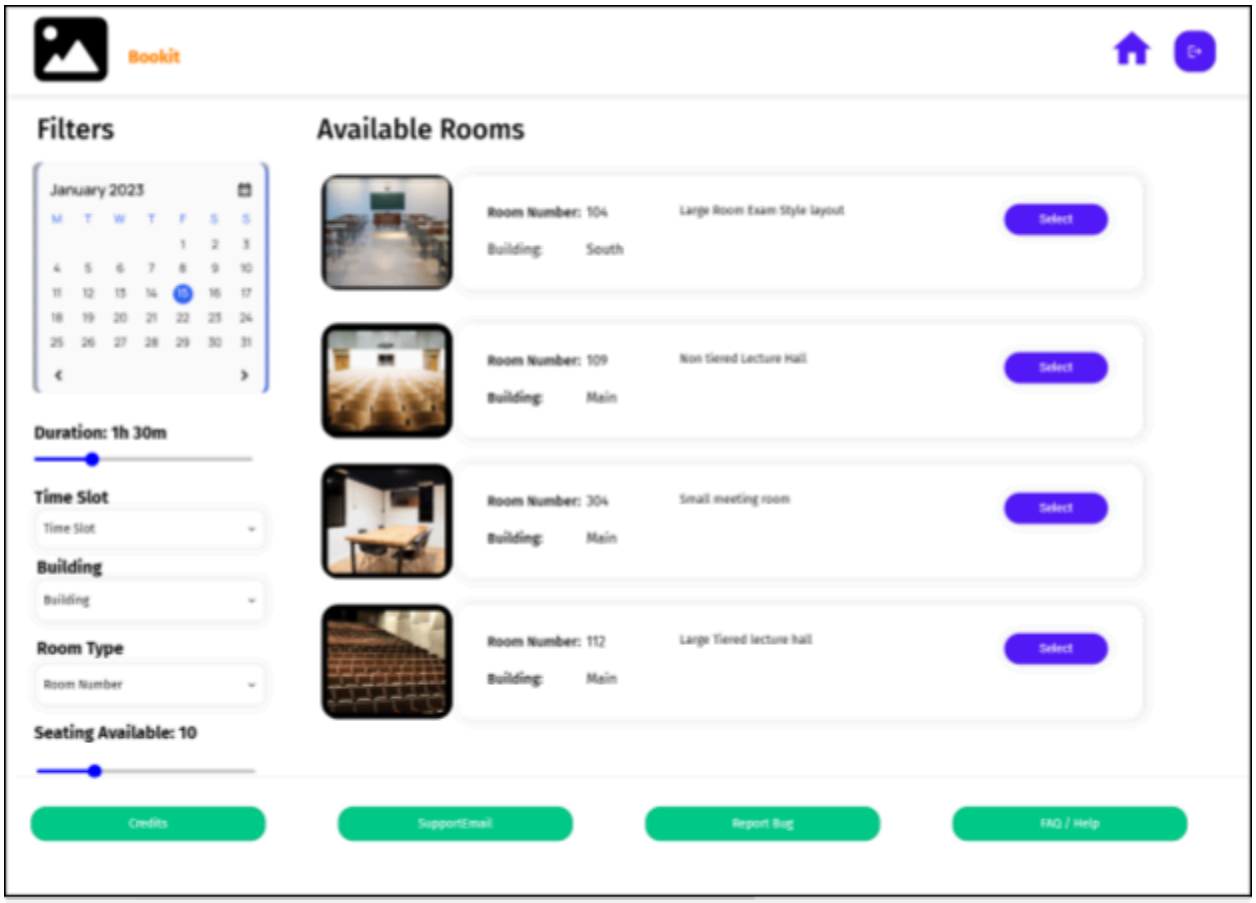


Figure 30: High fidelity UI for booking rooms

V0

We made a few minor text changes from the original wireframe to ensure clarity for the user: we changed the generic sentence describing the room to a series of features the room has pulled from the back end (E.g., Type: Lecture Theatre), and we also added buttons to order the selectable rooms - either by grouping them by the building they are in or ordering them from largest to smallest capacity.

A major change from the wireframe was that a date and time slot must be selected from the filter before a room can be selected as the V0 system has to make sure it is only presenting rooms that will actually be available at the desired time. To demonstrate this - we added a text element on each card as seen in the figure below.

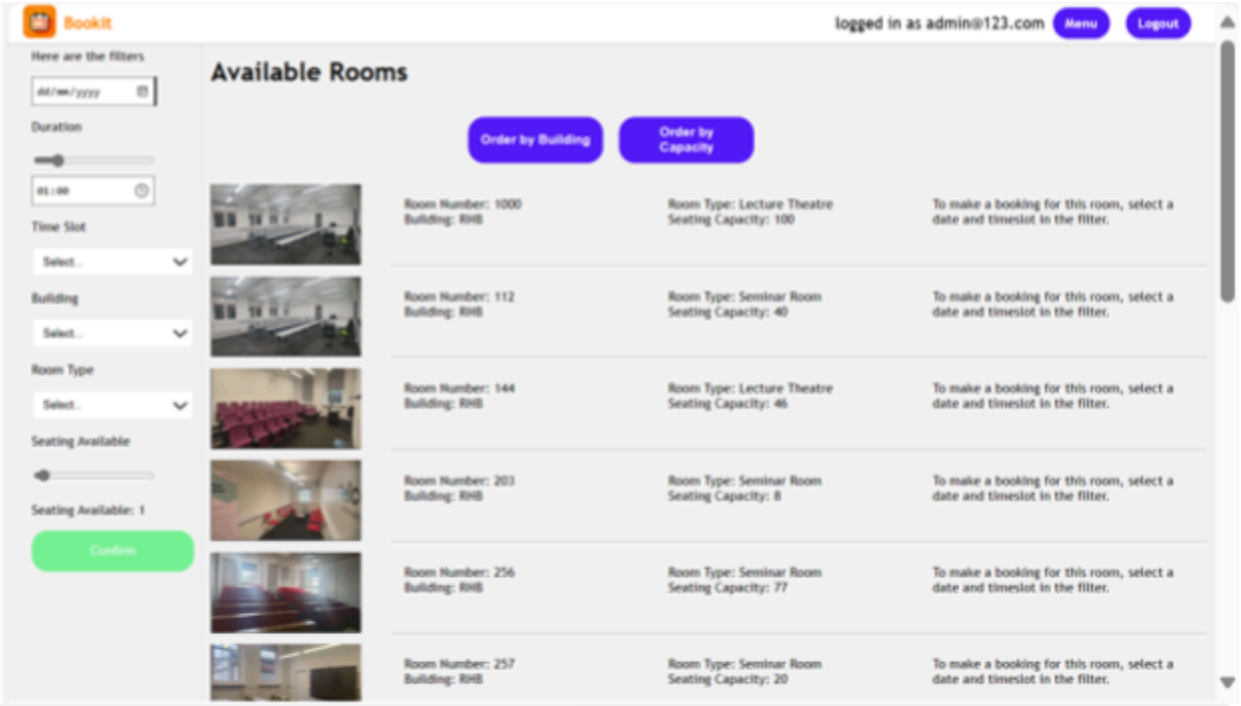


Figure 31: V0 UI for booking rooms

Once a time and date is selected, the page updates to display the available rooms and gives the user the option to pick which one they want by replacing this text element with the “select” button planned out in the wireframe; however this “select” text was changed to “book” - along with the date and time information. This is also to ensure clarity as seen below.

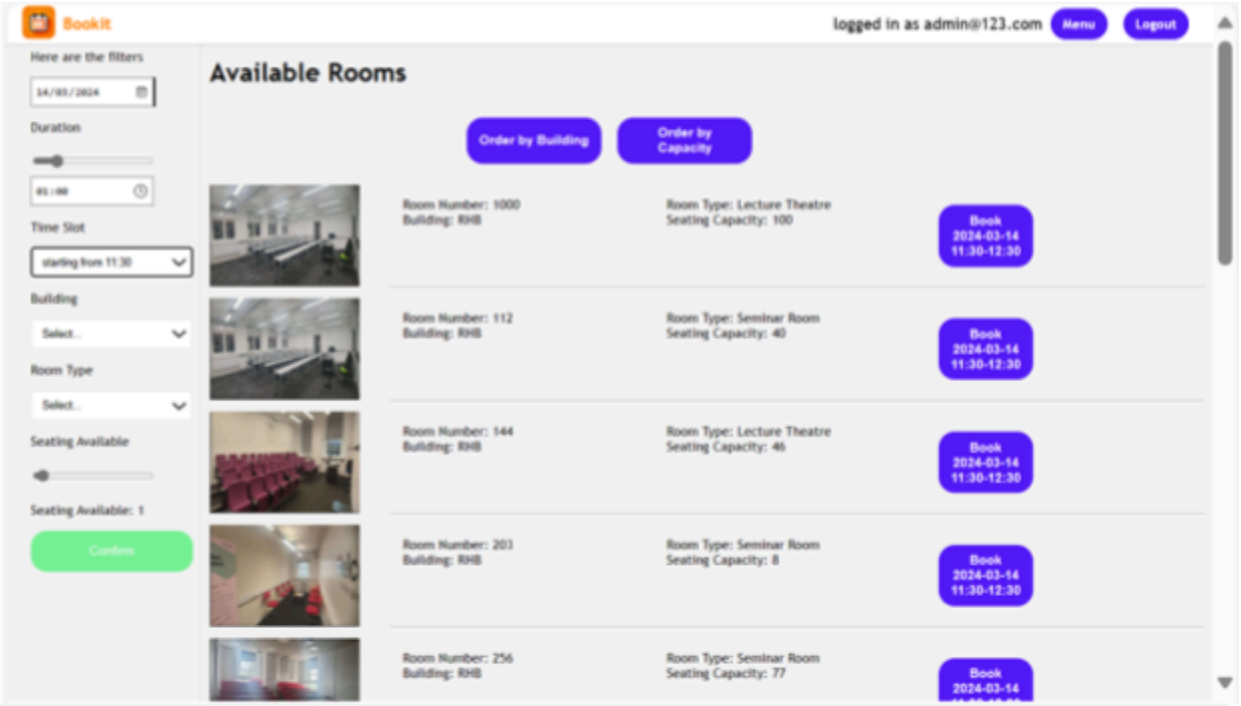


Figure 32: V0 filtered UI for booking rooms

V1

This page did not evolve much from the V0 version when it came to developing the V1 version - the main UI difference is an evolution based on feedback from our V0 user testing: in that users could book for dates in the past.

This was solved by greying out days in the past on this date and rendering them unclickable. A minor change that solved many issues on our back end as well as the SQL query we were using was displaying incorrect information due to bookings being made in the present being set for dates in the past.

Furthermore, we decided the “Order by” buttons were taking up a good chunk of space on the page while the header had lots of negative space - so we moved the “Order by” buttons to the header to make room for more room cards. We also applied this to all the room and booking list pages across Bookit as we felt it made the bookings and rooms feel “more available” to the user.

Another change from V0 to V1 was the introduction of AJAX requests to Bookit. This meant that for any pages (namely the booking list and rooms list pages) that used the generic filter system we created no longer needed the “confirm” button to update the page based on the selected filters, as the AJAX request would process the filter automatically. We therefore swapped out the “confirm” button in green for a “reset” button in blue so the user could reset the page to a default state should they choose to.

To see the visual effects of this change please see the below figures.

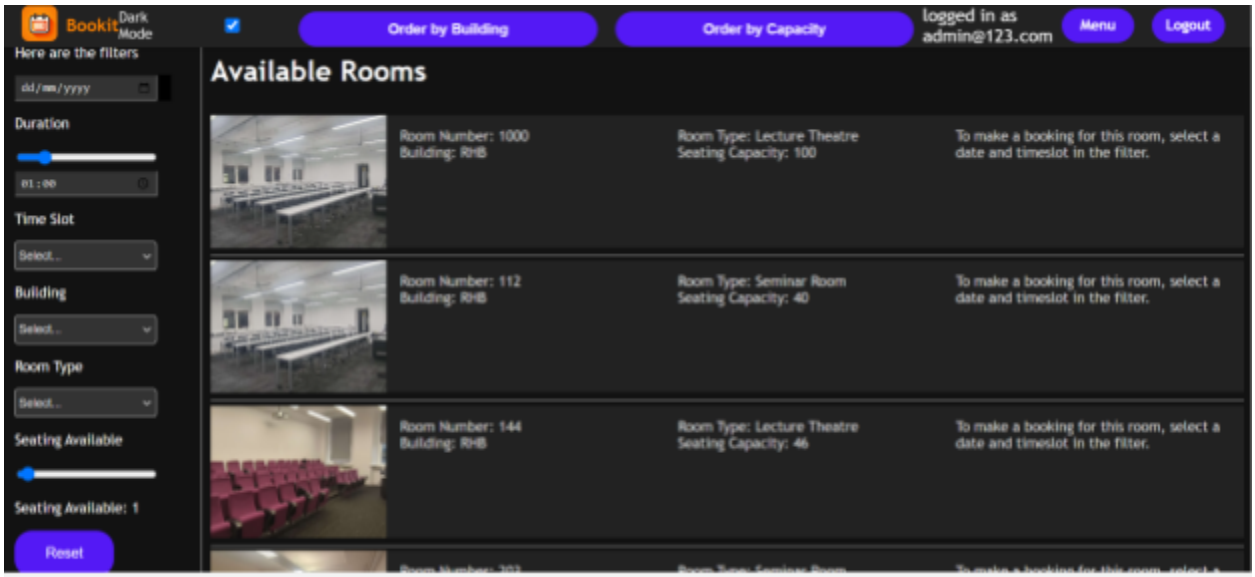


Figure 33: V1 UI for booking rooms

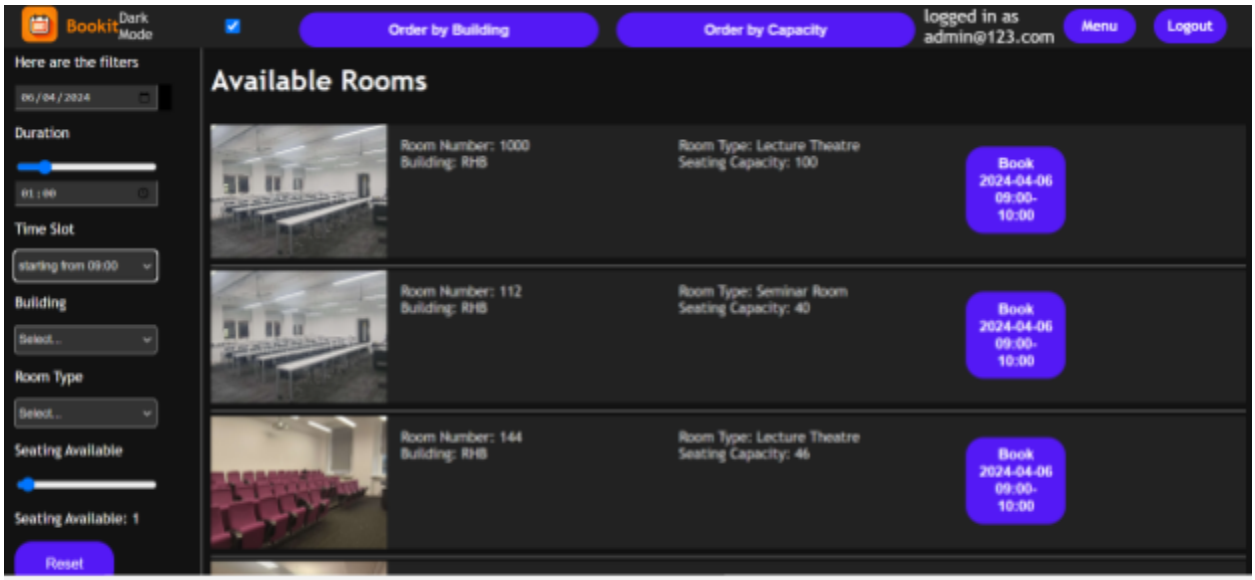


Figure 34: V1 filtered UI for booking rooms

Going forward

As Dr Sean McGrath - our module leader as mentioned previously - is technically a stakeholder in Bookit as an educator at Goldsmiths we had a brief discussion about possible developments for the rooms list page. He suggested that we could introduce a 10% seating leeway threshold for the capacity selector when booking rooms, as right now the numbers are exact and when you're booking for an event it is often hard to know just how many attendees you will have. We agree with this premise and have it in our considerations for future versions of Bookit.

We have also considered improving the UI by introducing headings for different buildings, or allowing for filters that only show one building at a time - as right now all the rooms are in one big list and it can be overwhelming. While this does not detract from the functionality of Bookit it could be considered harmful to the user experience and is therefore a worthwhile consideration for future updates to this page. This particular filter could also be applied to the other room and booking list pages as well should it be deemed necessary.

Add Booking

Concept

The idea for this page was to present the user with information regarding the room data, and also give them a space to fill in the given risk assessment template. The wireframe below has a very simple layout that was amended later to add more of the necessary complexities that adding the risk assessment had.

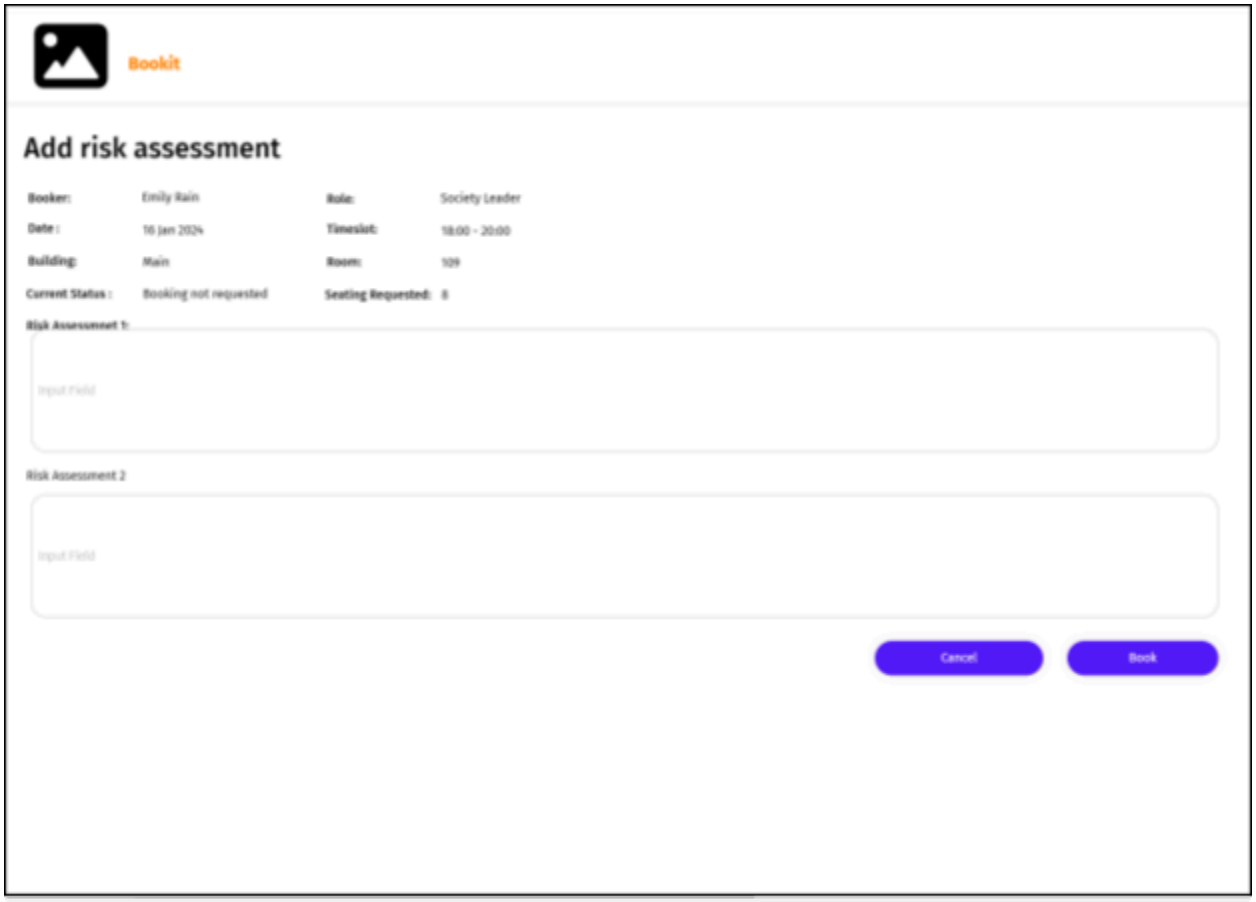


Figure 35: High fidelity UI for risk assessment and booking completion

The V0 version of the add booking page is identical to the wireframe. The only difference was that we did not get around to formatting the buttons on the page by the time we finalised V0. This did not end up affecting the functionality of the page but it was the main complaint from our V0 user testing for this page.

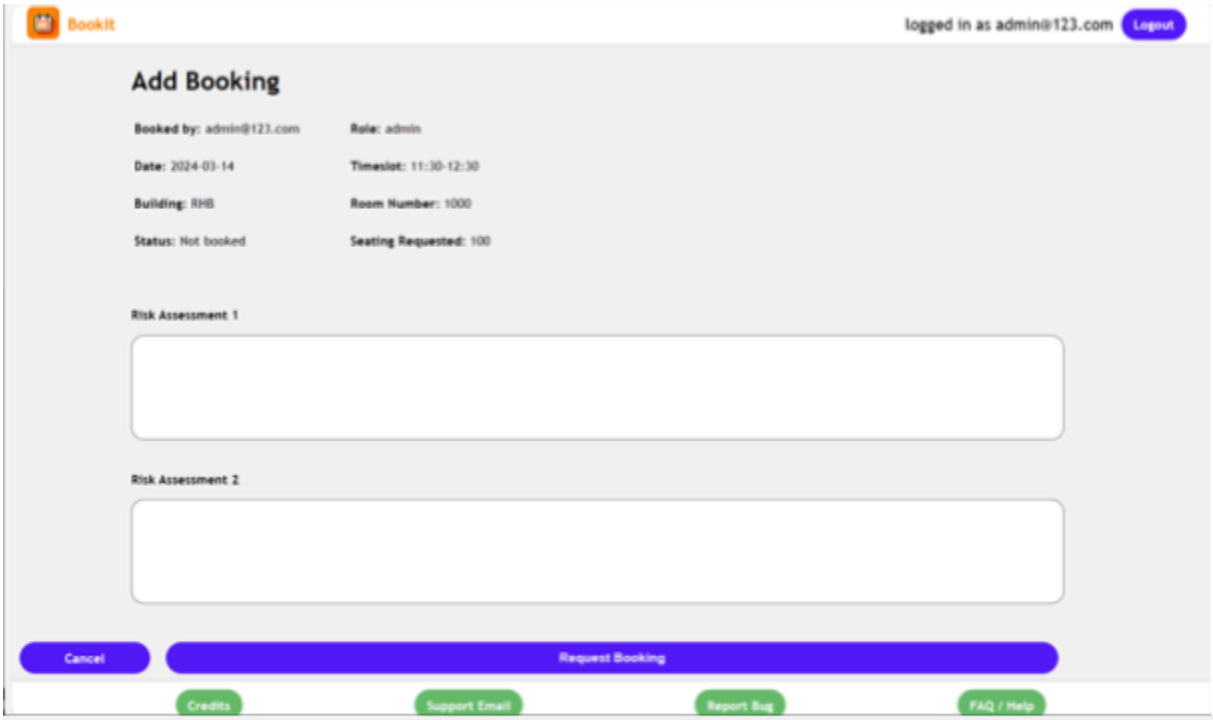


Figure 36: V0 UI for risk assessment and booking completion

V1

As well as reformatting the buttons at the base of the page - we also concluded after the production of V0 that the risk assessment structure we provided - just being two standard text entry fields was not adequate for our purposes. Not only did it not allow for enough information to be entered that is required by the Goldsmiths health and safety team due to the 200 character limit we imposed (originally to prevent too much data flooding the back end), but it also did not solve one of the main aspects of our problem statement in that the risk assessment process as it currently stands is slow and inefficient and requires too much back and forth between bookers and coordinators.

To rectify this - we considered creating a dedicated HTML form with all the required information that the user could enter and submit easily. However we felt this would be rather time consuming and would prevent us from doing vital bug fixes that were made apparent after the conclusion of our V0 user testing. Because of this - we decided on a compromise - by using the node module TinyMCE we baked the risk assessment document into the add booking page and allowed the user to edit it right there in Bookit. This allowed

for the current risk assessment template to be followed without the need for emailing forms back and forth and opening multiple applications - thus giving us a basic solution for our MVP.

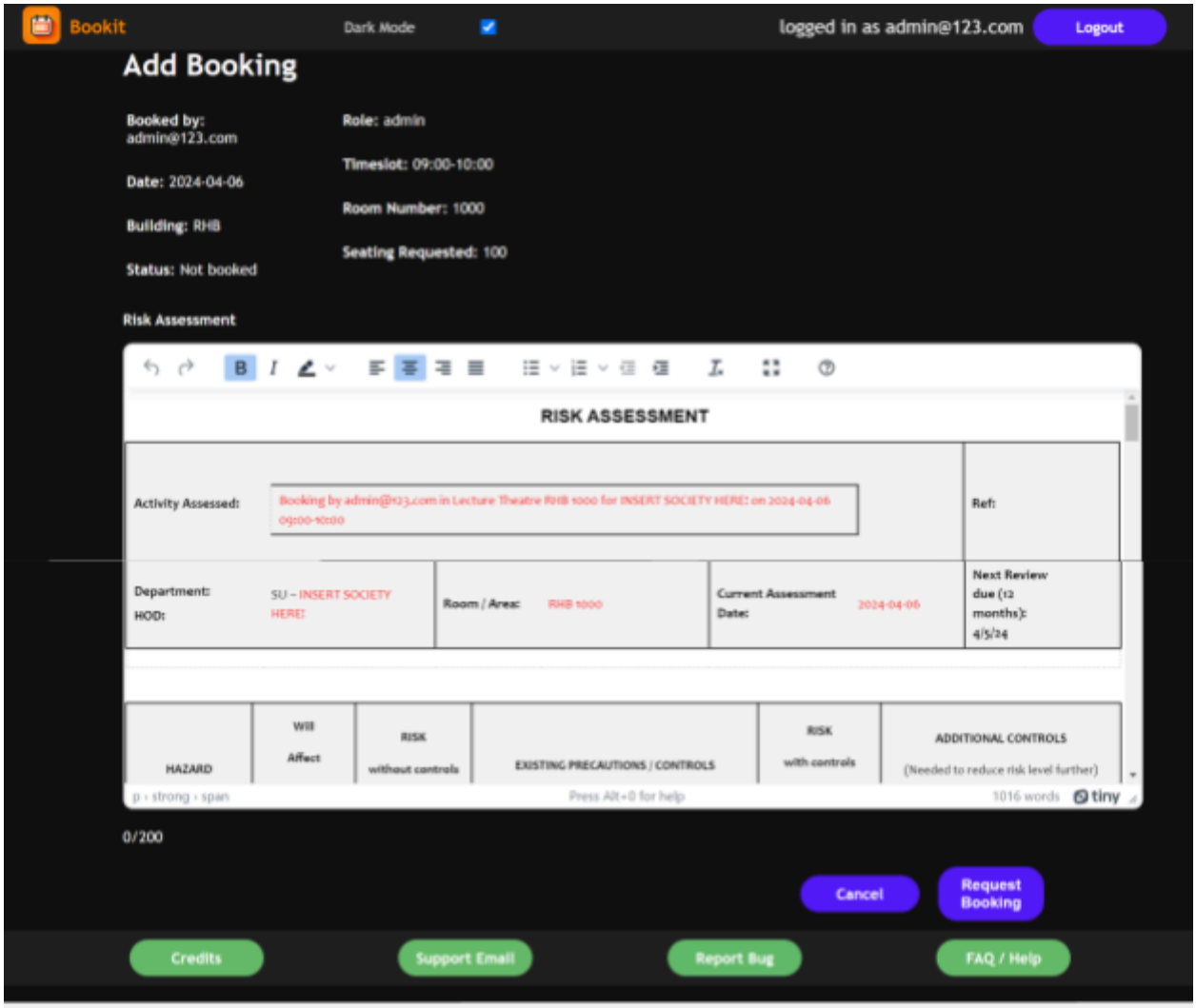


Figure 37: V1 UI for risk assessment and booking completion

Going forward

An argument could be made that our implementation of TinyMCE to template the risk assessments only minimally improves the efficiency of the booking process. An aspirational target for the future of Bookit is to create a dedicated HTML form for the risk assessment template that is both easier and faster to fill in and is more aesthetically pleasing than the document structure we implemented in V1.

Booking List

Concept

Booking list is the page that shows all the current bookings that have already been made. We wanted to show who the booking was made by and for what room, duration and time slot. We aimed to use a similar card aesthetic with text fields to represent this information, then the user could click on a button to view this detail more in depth (including the risk assessment), should they choose to. In short - the intention behind this page is to give the user a quick breakdown of the current bookings information so they know what rooms may be available at a prospective time frame - to more efficiently check this they can use the filter system presented on the left hand side.

We created the below wireframe with these considerations in mind.

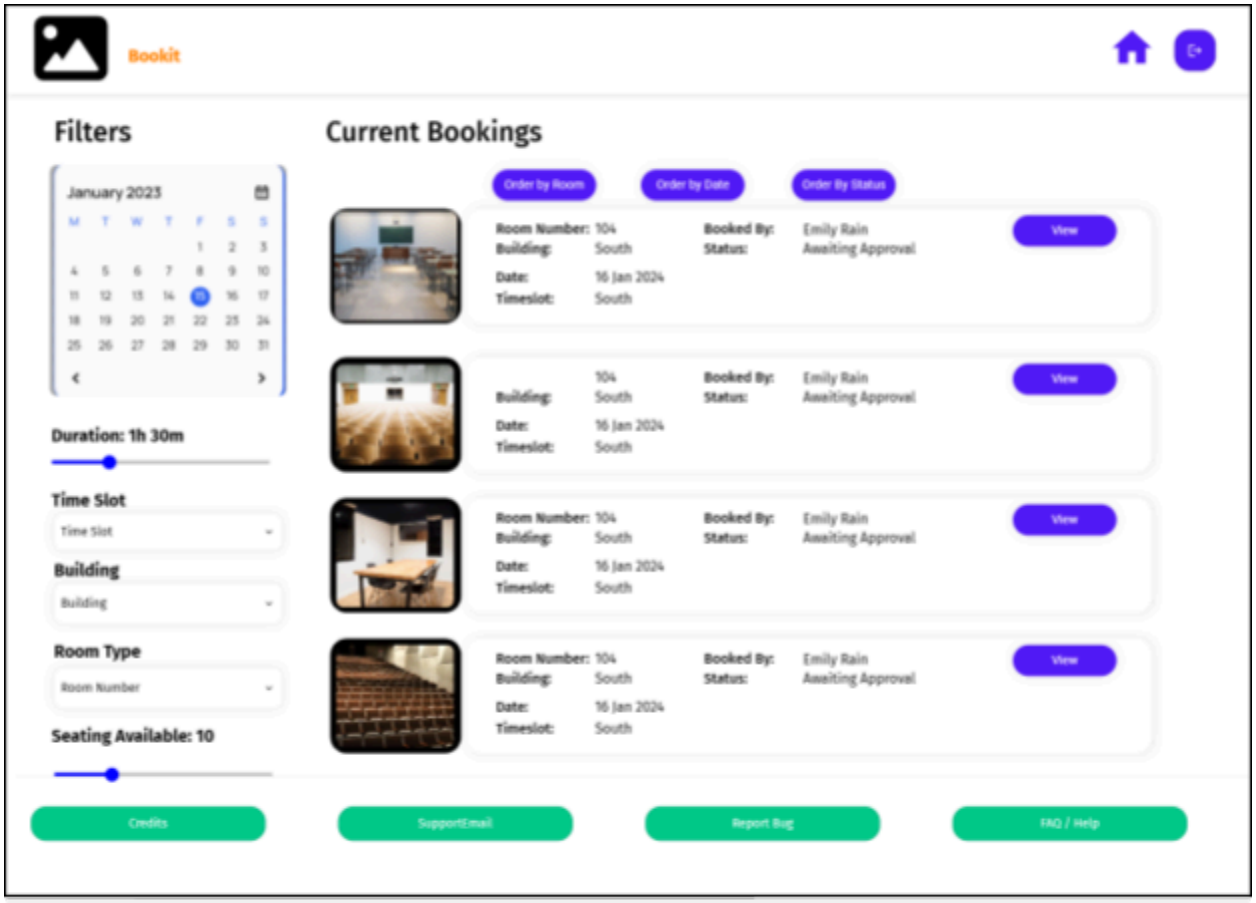


Figure 38: High fidelity UI for coordinators view on current bookings

V0

Aside from the aforementioned changes to the header and UI - there was very little evolution from the concept wireframe to the V0 version of the UI for the bookings list page. The only real difference being that we made the “Order by” buttons and text slightly bigger so it was easier to read and more accessible for visually impaired users.

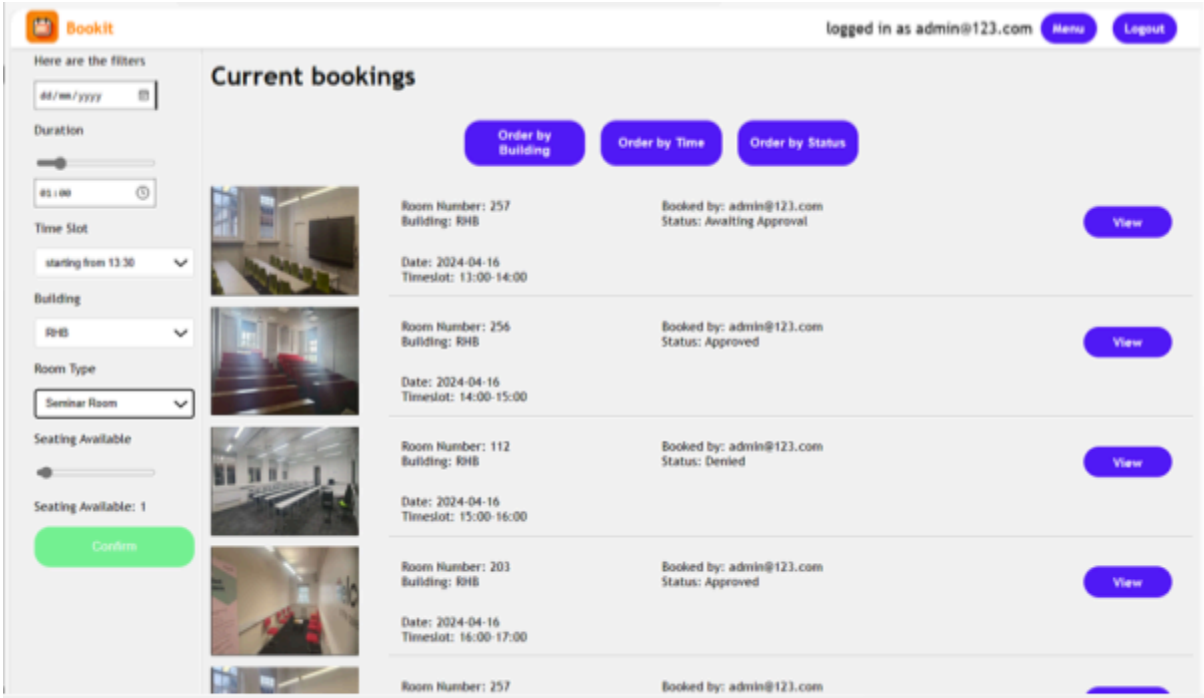


Figure 39: V0 UI for coordinators view on current bookings

V1

Aside from the aforementioned changes to the rooms list page and header UI that were also applied to this page, we added an edit button to the bookings made by the currently logged in user that redirected to the “edit booking” page.

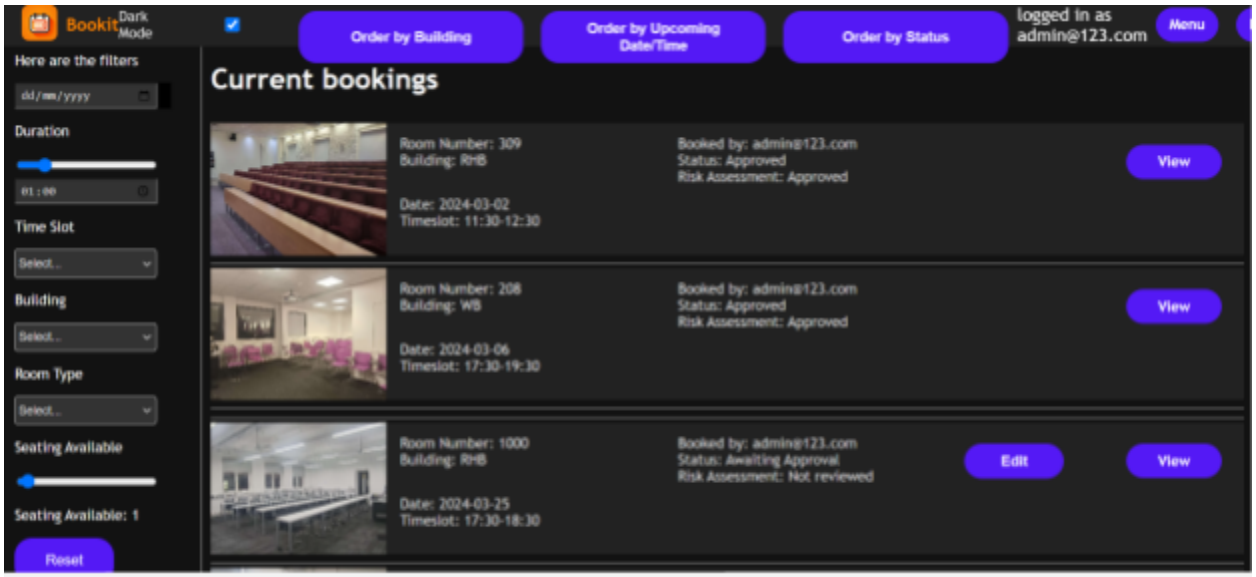


Figure 40: V1 UI for coordinators view on current bookings

Going forward

The above figure demonstrates that there are once again, spacing issues on the header. Reducing the size of the buttons and improving the CSS spacing will improve the user experience and make the page more visually appealing. It would also mean all elements are fully contained within the header and the logout button is no longer cut off to the side. This does not affect the overall functionality of the page, but it does affect the overall look and feel of Bookit which has a detrimental effect on the user experience.

View Booking

Concept

View booking is functionality available to all users while editing a booking is only available to coordinators and admins. But the functionality is similar just without the option to update the risk assessment data. The idea is to present the data clearly without distracting the user with additional buttons or functions like on the edit booking page - that is why the two pages are separate. This also displays the risk assessment. The concept has a far more simple risk assessment layout than the V1 product seen below for the aforementioned reasons (see “Add booking” above).

Figure 41: High fidelity UI for viewing booking detail screen

V0

The UI proposed in the concept wireframe is nearly identical to the one developed for V0.

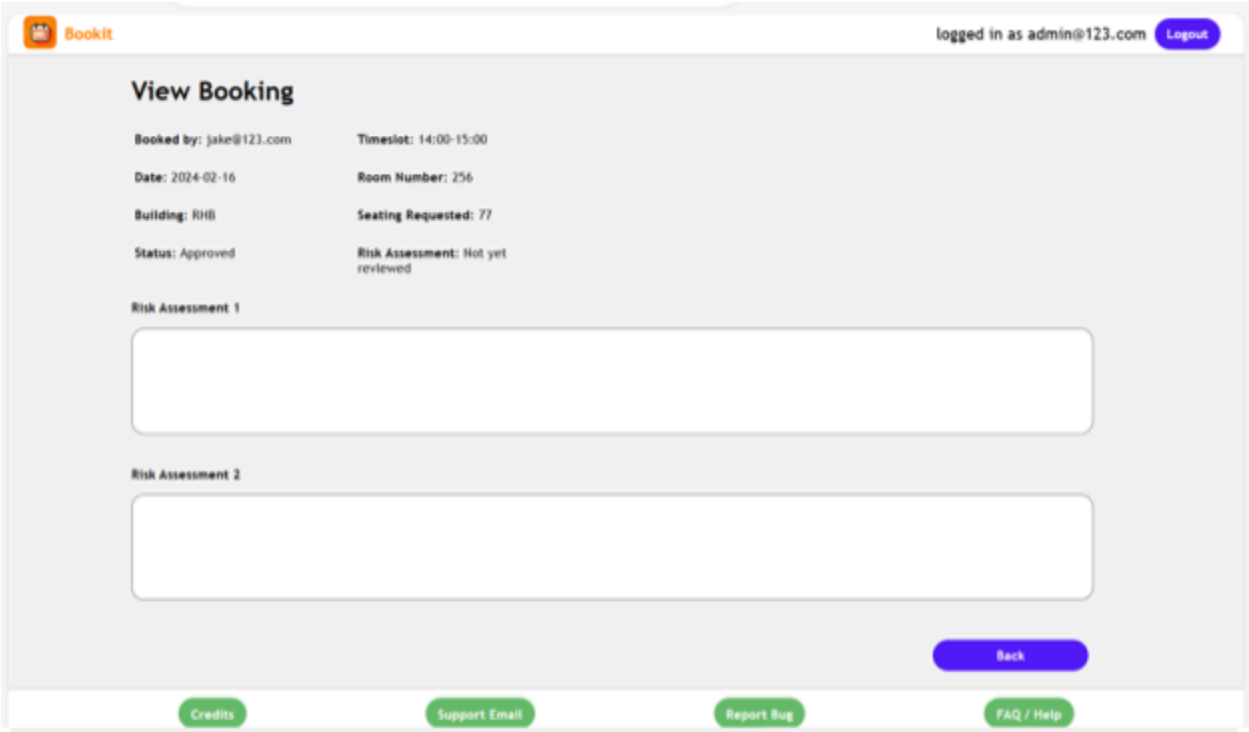


Figure 42: V0 UI for viewing booking detail screen

V1

We added some more useful functionality to this page in V1 which changed the UI. To start with, admins and coordinators now have the option to cancel bookings from this page, as seen below.

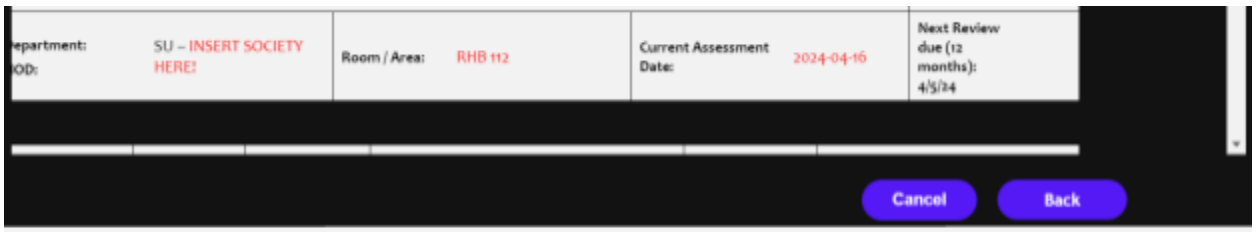


Figure 43: V1 UI for viewing booking detail screen with cancel and back

Furthermore, if you are viewing a booking that you made yourself, an edit button has been added to redirect you to your bookings edit page.

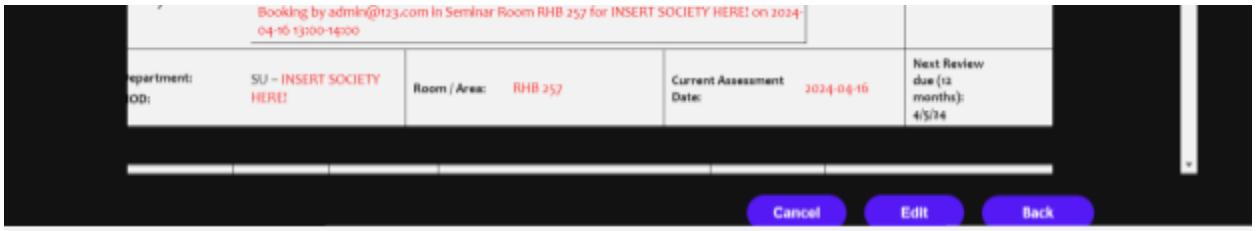


Figure 44: V1 UI for viewing booking detail screen with cancel, edit and back

Going forward

We do not plan any major overhauls for this page as we believe it fulfils its purpose to a reasonable degree. However there was some confusion during testing about the difference between “back” and “cancel” - with back taking you back to the bookings list and cancel deleting the booking. A few of our testers believed that both buttons would take you back to the booking page and got confused by this. Changing these buttons to “back” and “cancel booking” would help improve clarity as to what these buttons do.

Edit booking

Concept

The concept for the edit booking page is very similar to the above view booking page. However in this page the intention was that the user (either an admin or a coordinator) can make edits to the risk assessment and then update the booking database entry to reflect these changes. The below wireframe is the general concept for this page (note the “book” button - this was later changed to an “update” button; see below).

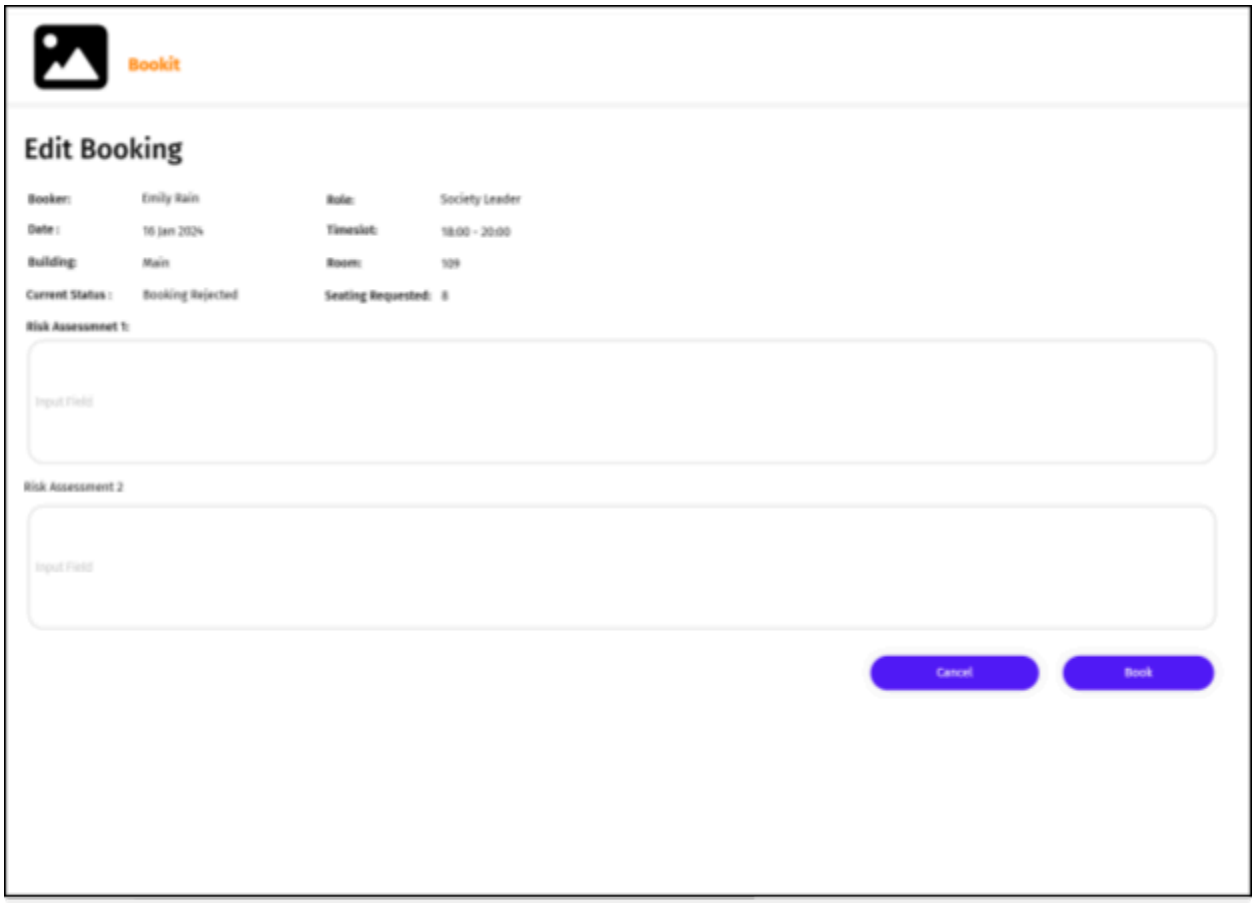


Figure 45: High fidelity UI for editing bookings

V0

Edit booking was not included in V0.

V1

As mentioned previously - the “Book” button in the concept was changed to “Update Booking”. Also - the changes made for the add booking page for the risk assessment were applied here as well.

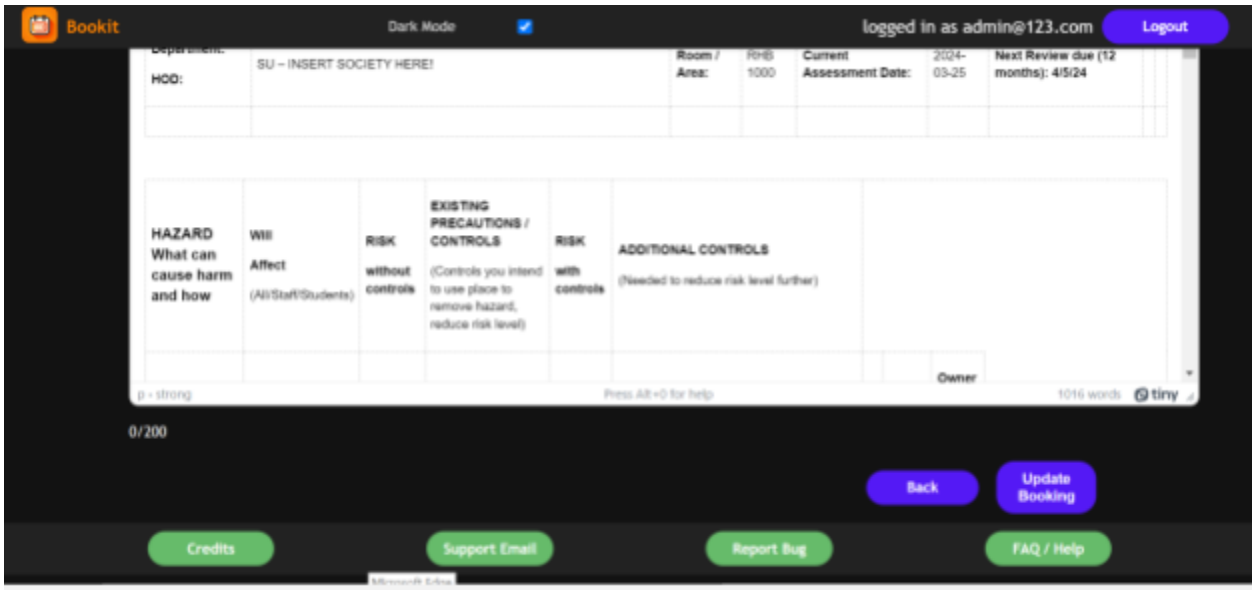


Figure 46: V1 UI for editing bookings

Going forward

Currently only the risk assessment is editable. There should be options to edit the time slot and duration of the booking as well for a given date.

Requests List

Concept

The requests list page is only available to coordinators and admins. It shows which bookings have been requested and are still up for review. We followed the same card aesthetic that we've been employing throughout the site - and we knew that we needed a quick way for coordinators to see the information on requested bookings. So for the concept we included the name of the person that made the booking, the date, timeslot, building and room number. Similarly with previously explained concepts - requests can be filtered as shown. Also, we knew we wanted a quick way for the coordinators to be able to review the risk assessments, but our concept did not account for the risk assessment approval process that takes place before the booking approval process - therefore we included the "Approve" and "Reject" buttons right there on the booking card. This was later changed as seen below in the figure for V0.

With these considerations in mind we produced the following wireframe.

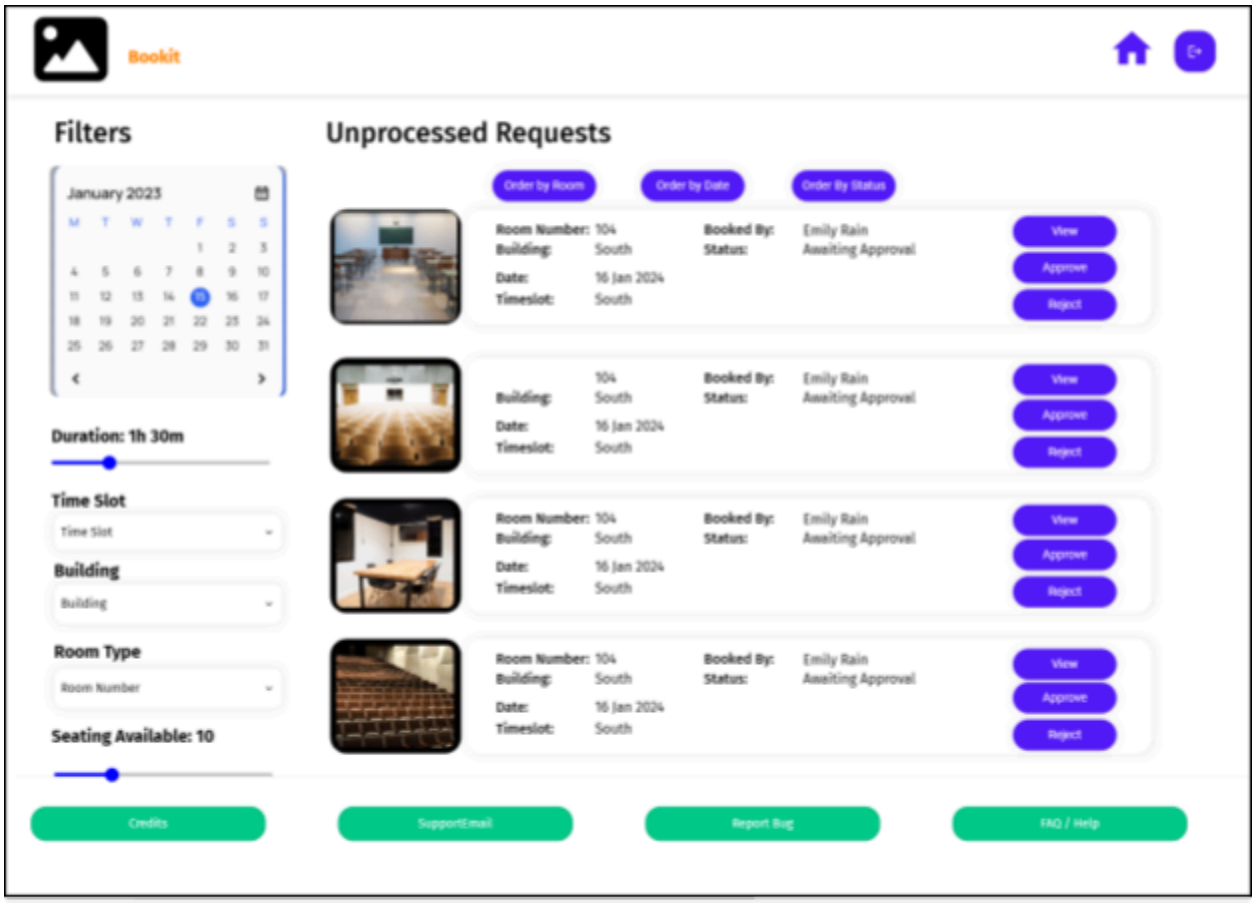


Figure 47: High fidelity UI for coordinator viewing booking requests

V0

For V0 we realised we didn't need an "Order by status" button as all the booking on this page would be "awaiting approval" anyway. Furthermore, we changed the "view" button to say "Review booking" to ensure clarity. As stated previously - we moved the options for approval and rejection of bookings to the booking review page after beginning production on V0.

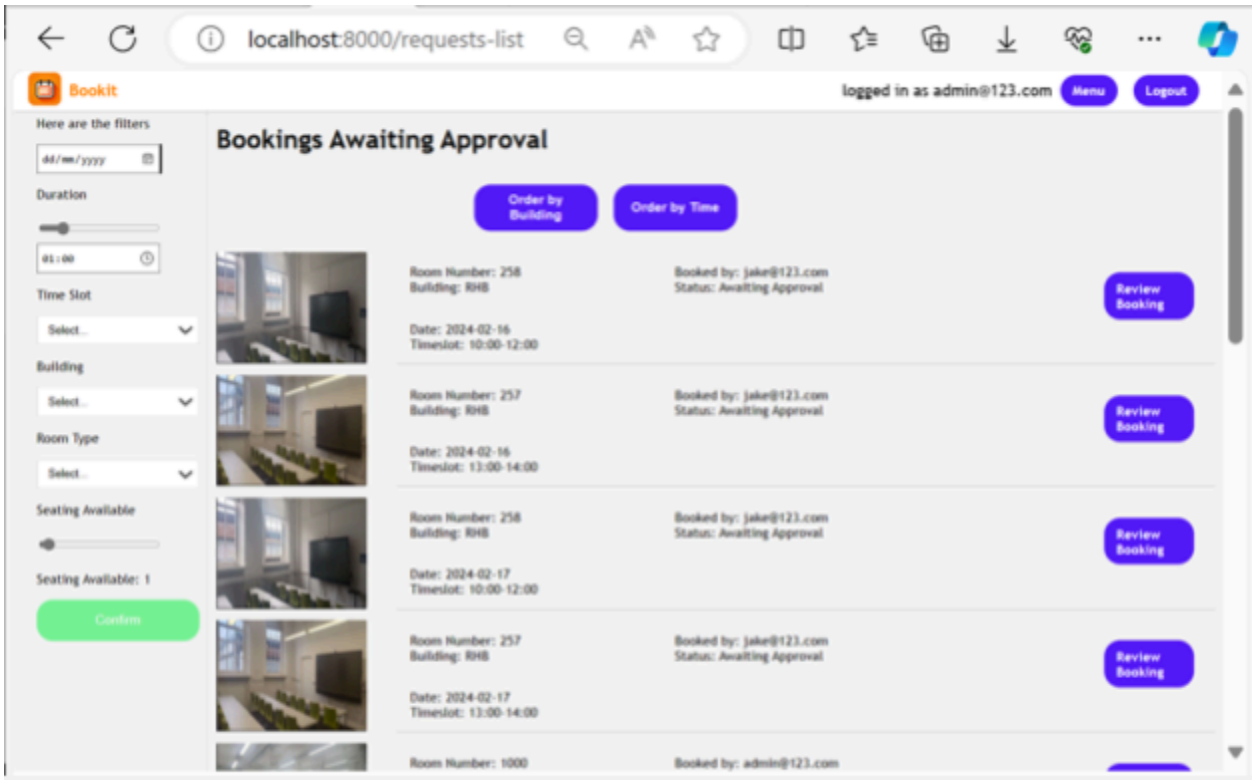


Figure 48: V0 UI for coordinator viewing booking requests

V1

Aside from the aforementioned changes to the header UI and filters, there was very little evolution on this page from V0 to V1.

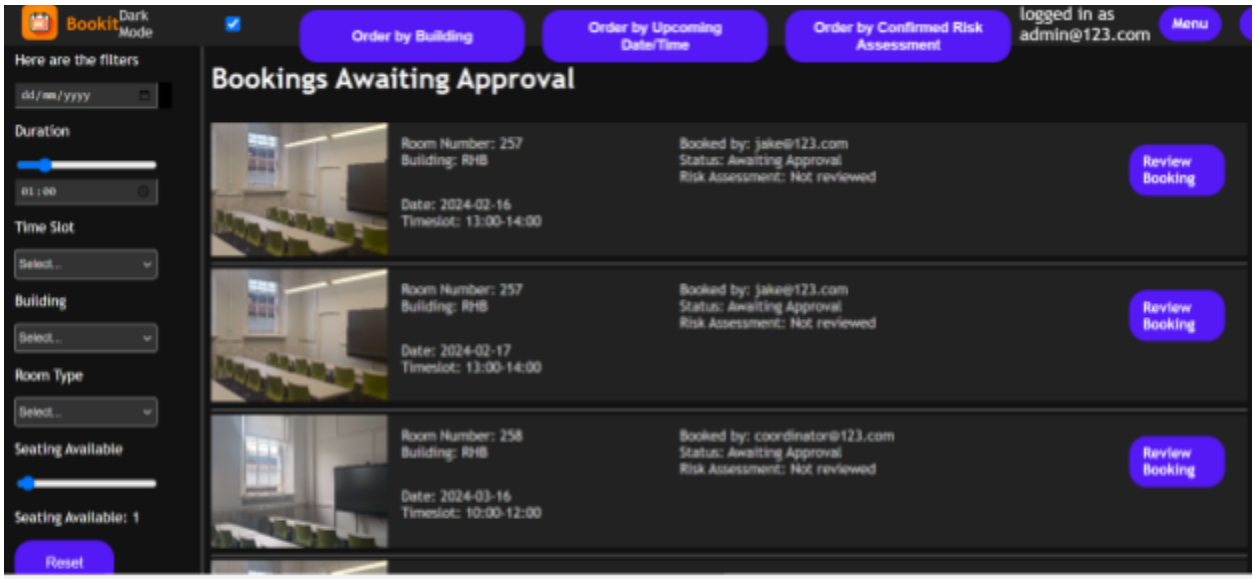


Figure 49: V1 UI for coordinator viewing booking requests

Going forward

Aside from the previously mentioned spacing and sizing fixes on the header - we do not plan any major updates for this page as we feel it fulfils its purpose.

Review Booking

Concept

We did not wireframe this page. As mentioned briefly previously - we originally intended the system to approve and reject bookings to be baked into the requests list page - but this did not account for the process of approving the risk assessment for a booking separately. Therefore, we created this new page in a similar style to the view booking page: it has options to confirm or deny risk assessments, then confirm or deny the entire booking depending on a confirmed risk assessment. This page is only available to coordinators and admins.

V0

This page was essentially the same as the view booking page but with options to approve and reject risk assessments. This can be seen in the figure below.

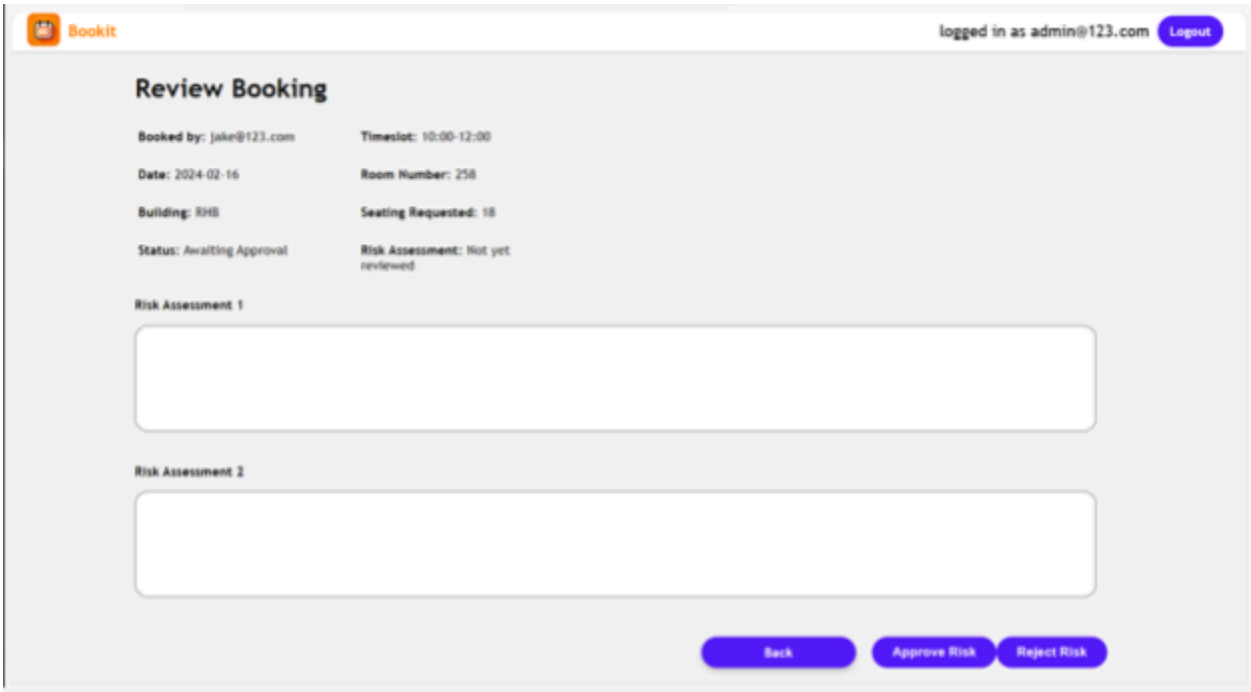


Figure 50: V0 UI for review booking

If the risk assessment is reviewed - the buttons change to “Approve Booking” and “Reject Booking” respectively.

V1

Aside from the previously documented changes to how the risk assessment is displayed there were no further changes to this page in V1.

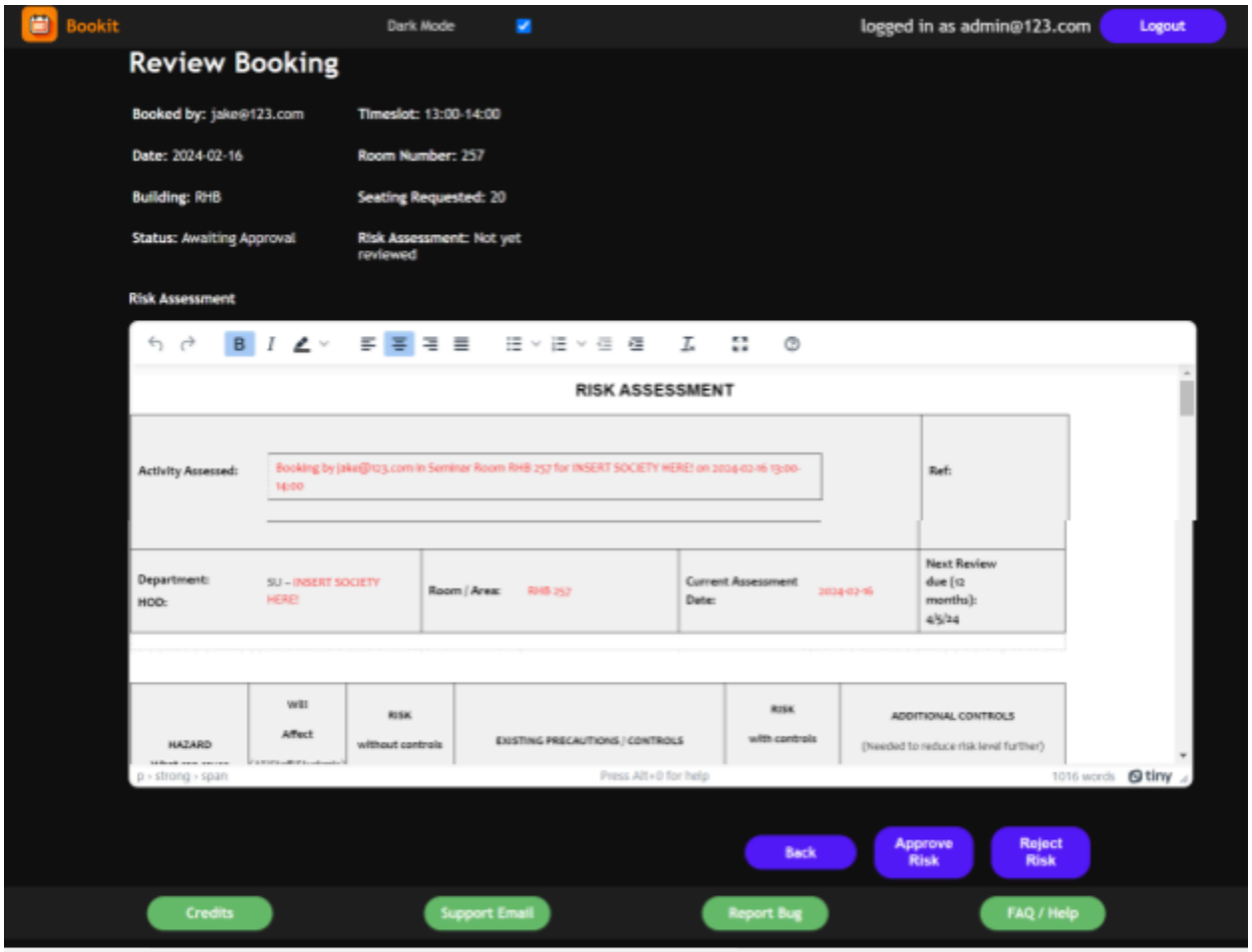


Figure 51: V1 UI for review booking

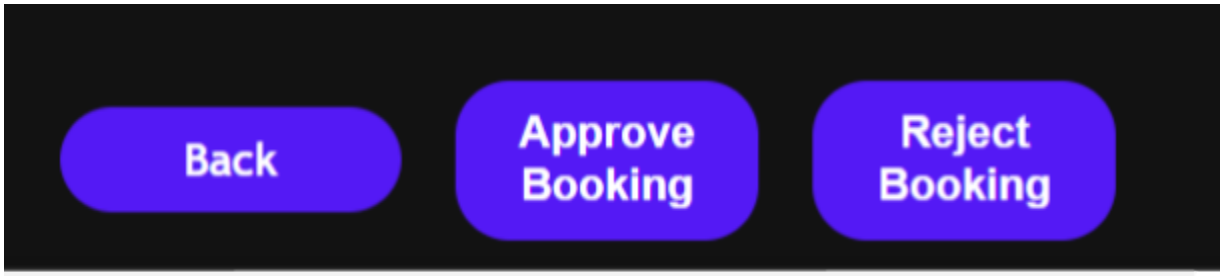


Figure 52: V1 buttons for review booking

Going forward

This page is mostly fit for purpose - but another button to confirm both the booking and the risk assessment at the same time might provide coordinators with a more efficient way of confirming bookings as opposed to loading the review page twice - therefore we include this in our considerations for this page when it comes to the future of Bookit.

Approved List

Concept

A page available only to coordinators and admins that shows all bookings that have been approved - both risk assessment and the booking itself. The page does this by filtering the SQL query to only show bookings with an “Approved” status. A similar process is followed for the requests list page and the “Awaiting approval” status.

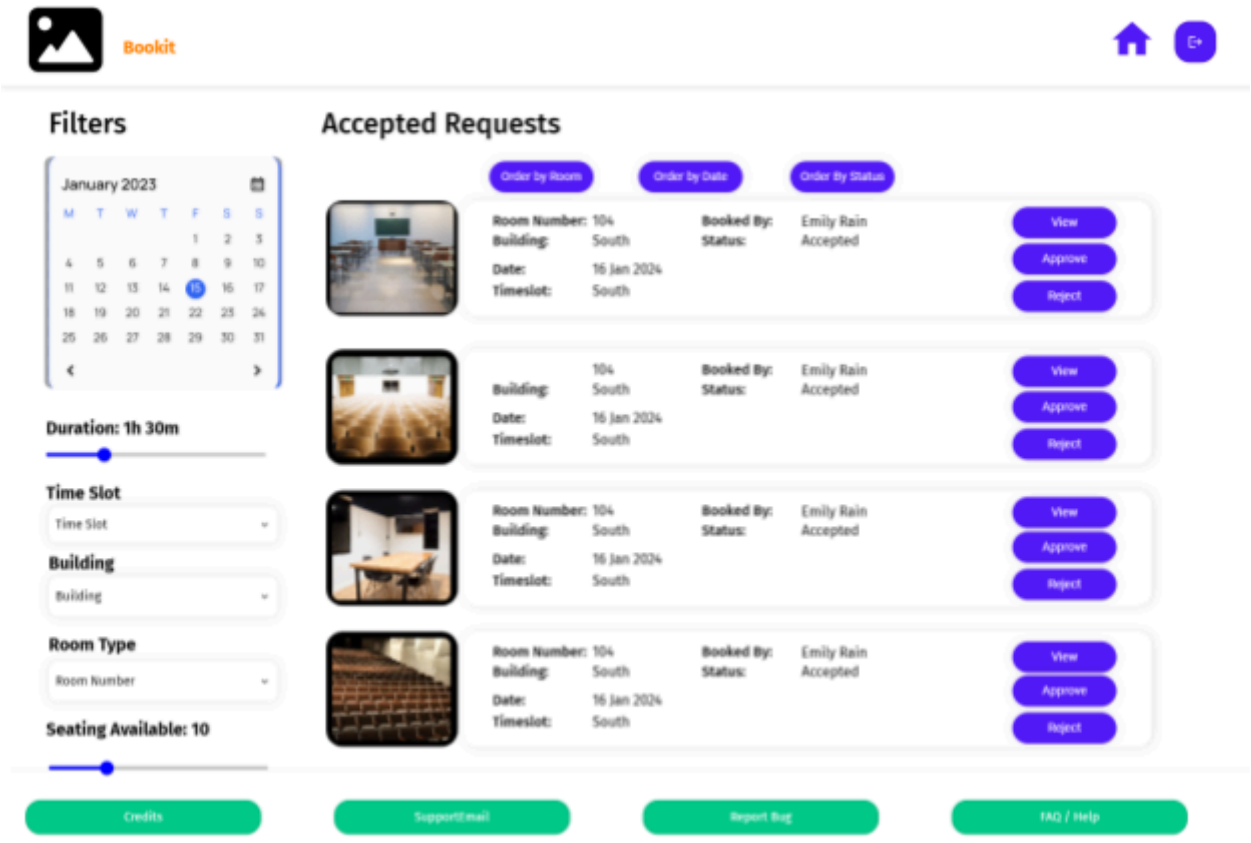


Figure 53: High fidelity UI for coordinator viewing accepted requests

When observing our concept wireframe we realised we didn't need the approval and rejection process to carry over to the approved list as this page was just supposed to show coordinators which bookings had already been approved. So we did not include these buttons/redirects for this page for V0.

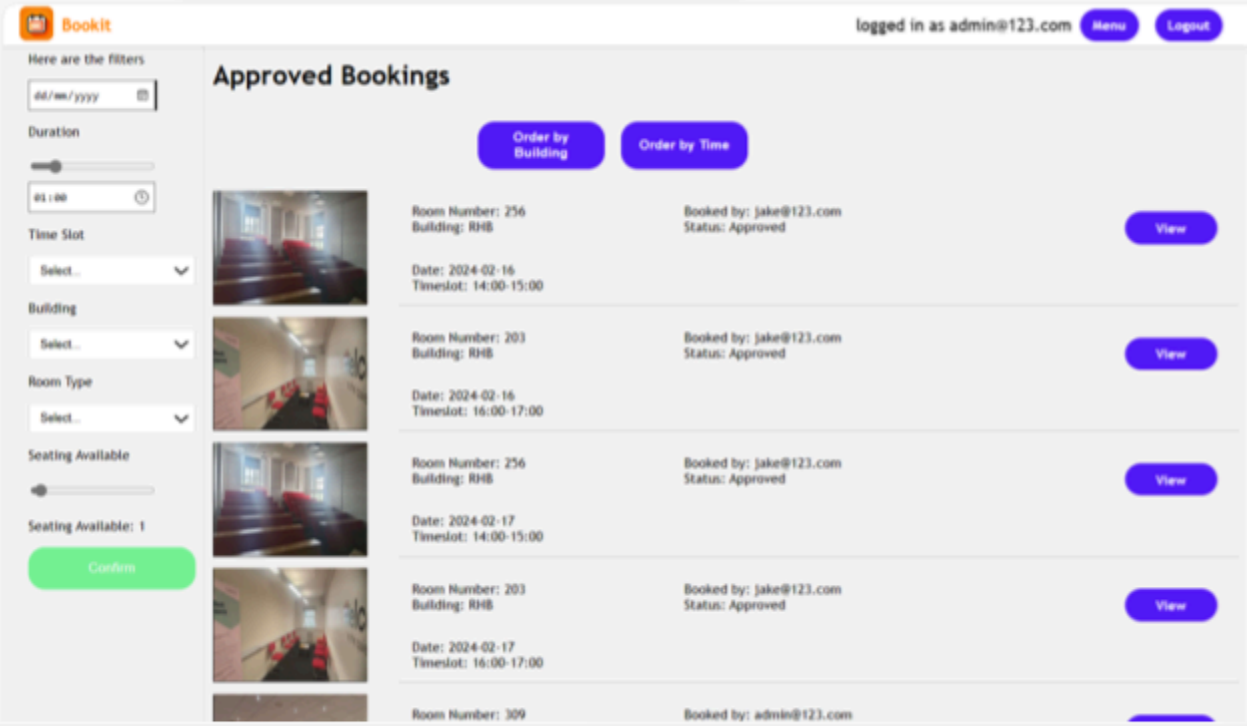


Figure 54: V0 UI for coordinator viewing accepted requests

V1

Our user tests for this page revealed a few things - one was that "Order by time" was unclear as to what the button actually did. So we renamed it to "Order by Upcoming Date/Time" to ensure clarity. We also added another "Order by" button for the bookings that were most recently approved as our testers told us that would be the most useful order the bookings could be in.

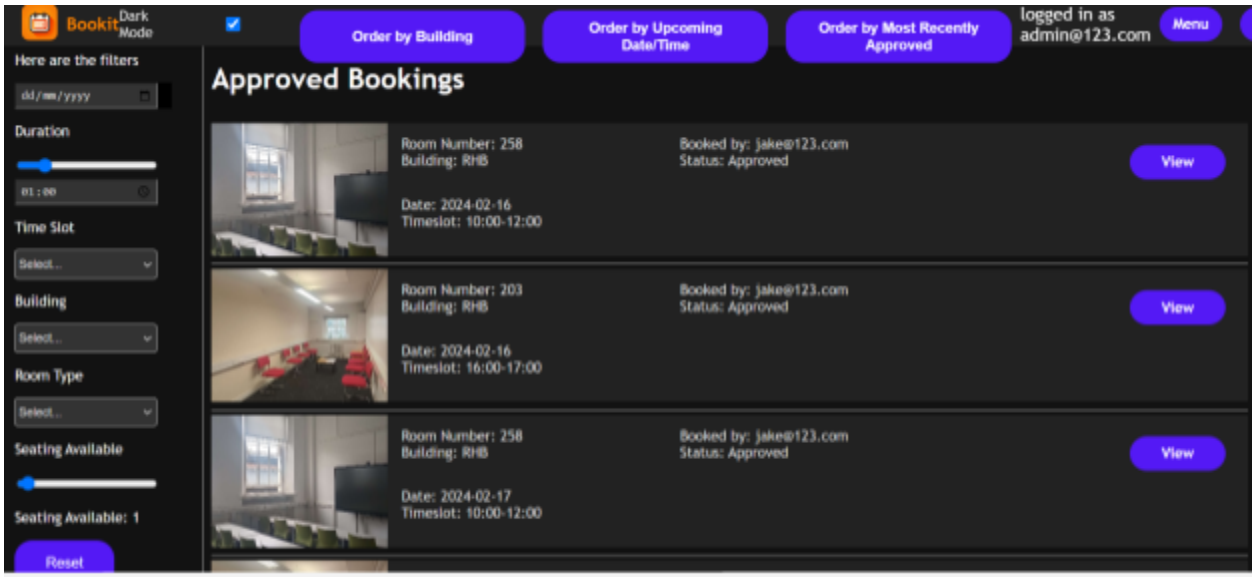


Figure 55: V1 UI for coordinator viewing accepted requests

Going forward

This page is mostly fit for purpose already - perhaps restricting the list to only upcoming bookings, as well as ordering by most recently approved by default could improve the user experience. Therefore we have included these in our considerations for developments for Bookit.

Add user

Concept

This page is for registering a new user as an admin. It allows for the entering of an email, password (with checks for complexity), and details about the user and their level of clearance. This was not a wireframes page either, as it was developed as more of an afterthought during the production of V0. However the layout and content was simple enough to the point where our lack of wireframe did not hinder us when creating this page.

V0

The basic premise for this page was to allow for a new email, password level of clearance, and society name/module (depending on whether the user is a society leader or lecturer respectively) to be registered through a single HTML form. The data is then fed through to the back end.

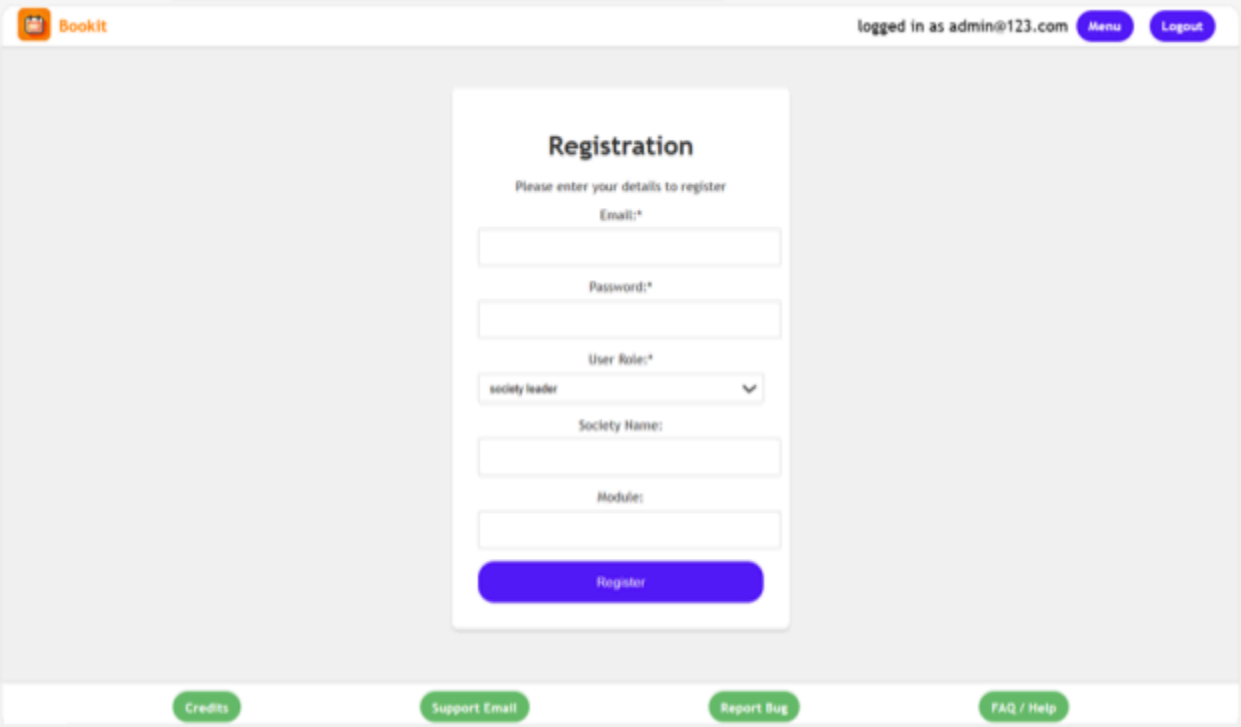


Figure 56: V0 UI for adding user

V1

The only change from V0 to V1 was the introduction of password complexity checks to ensure we maintained our level of security. Therefore we put a text element on the screen to show the admin what the password requirements are.

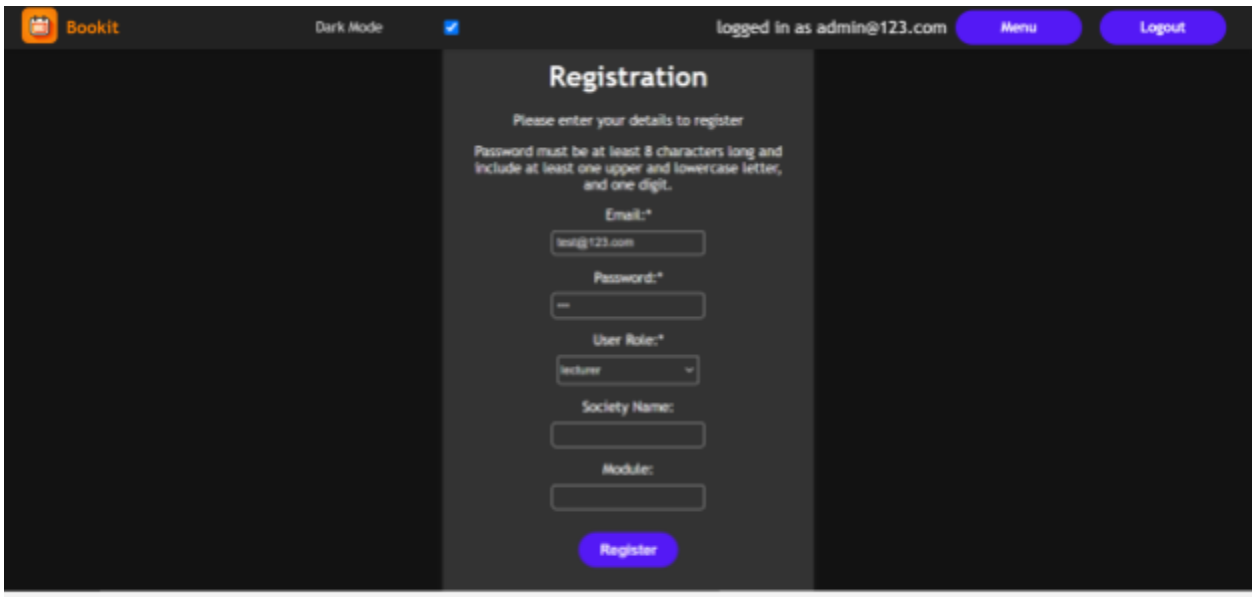


Figure 57: V1 UI for adding user

If the requirements are not met - the registration menu then changes to look like this to reflect the lack of a complex password.

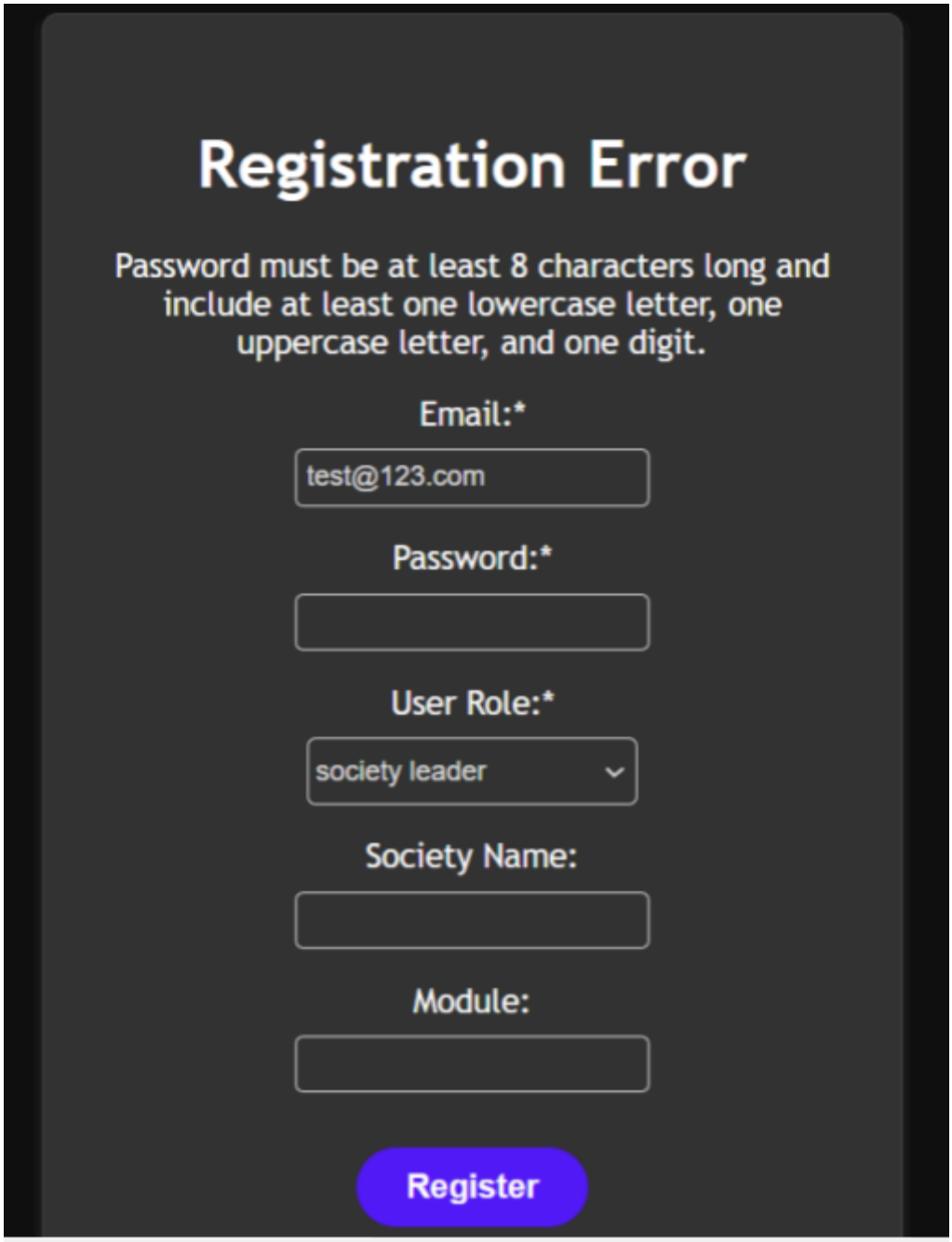


Figure 58: V1 UI for adding user error

Going forward

Currently both the module box and the society name box are available at all times regardless of the type of user that is being registered.

A good change for the future would be that the society name box would only be available if registering a society leader user, and the module box only available when registering a lecturer.

This would be a lower priority task - but would still benefit the user experience overall.

Add Room

Concept

A similar situation was had with the add room page as was had with the aforementioned add user page. The intention with this page is to have the user (an admin) enter details about the room, including the room number and capacity for the room - as well as selecting the building it is in from a drop down list and uploading a photograph from their system. We did not wireframe this page, as was the case with the add user page, as this also an afterthought when developing our considerations for admin functionality after the project proposal submission. However this page also did not hinder us during development of Bookit, even despite our lack of wireframe, thanks to the simplicity of the page and its “one form” structure.

V0

The V0 version of this page allowed for a room number, building name, capacity and picture URL to be entered for the room. You could also select the room type and whether or not the room was accepting bookings from a drop down list.

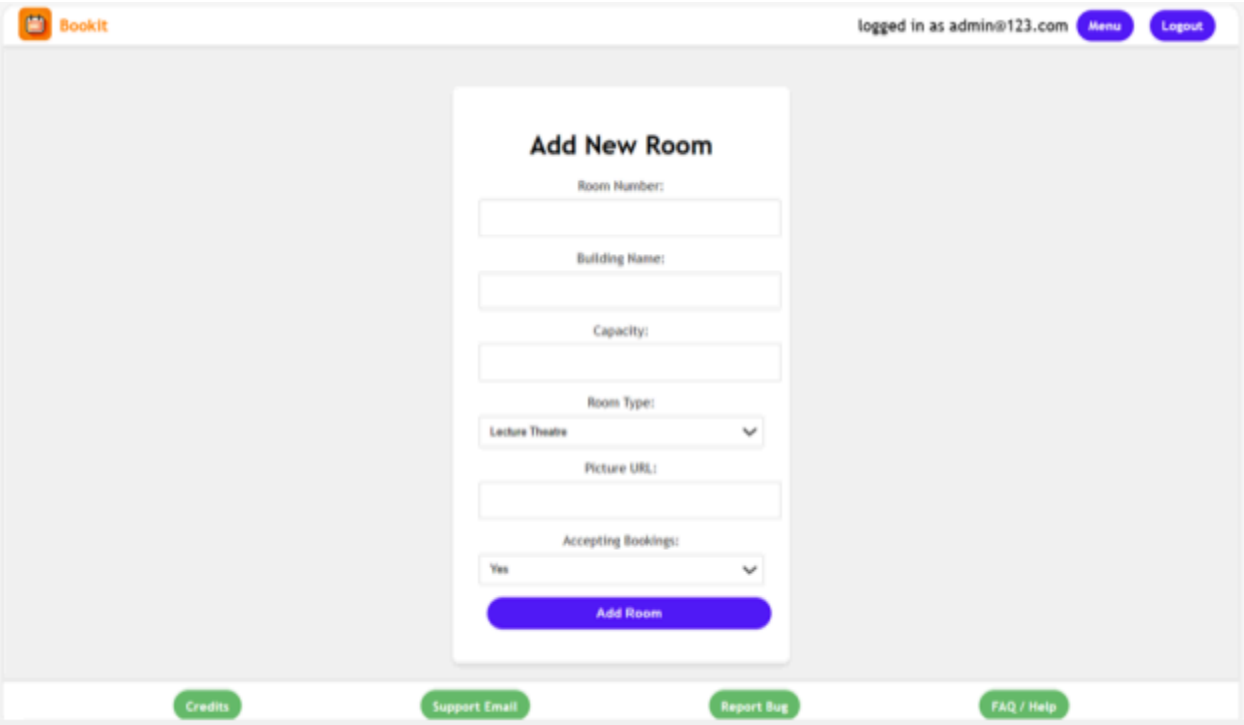


Figure 59: V0 UI for adding room

V1

One of the main complaints testers had from V0 was that uploading a picture to a separate hosting site and then pasting that URL into Bookit for this page was long and tedious - and it also went against the main philosophy of Bookit in that we wanted to make the booking process more efficient in all regards. So we added an option to upload a picture from the user's system as well as using a picture URL on an image hosting platform.

Testers also noted that it was possible to enter building names that didn't exist - so we changed it to a drop down list.

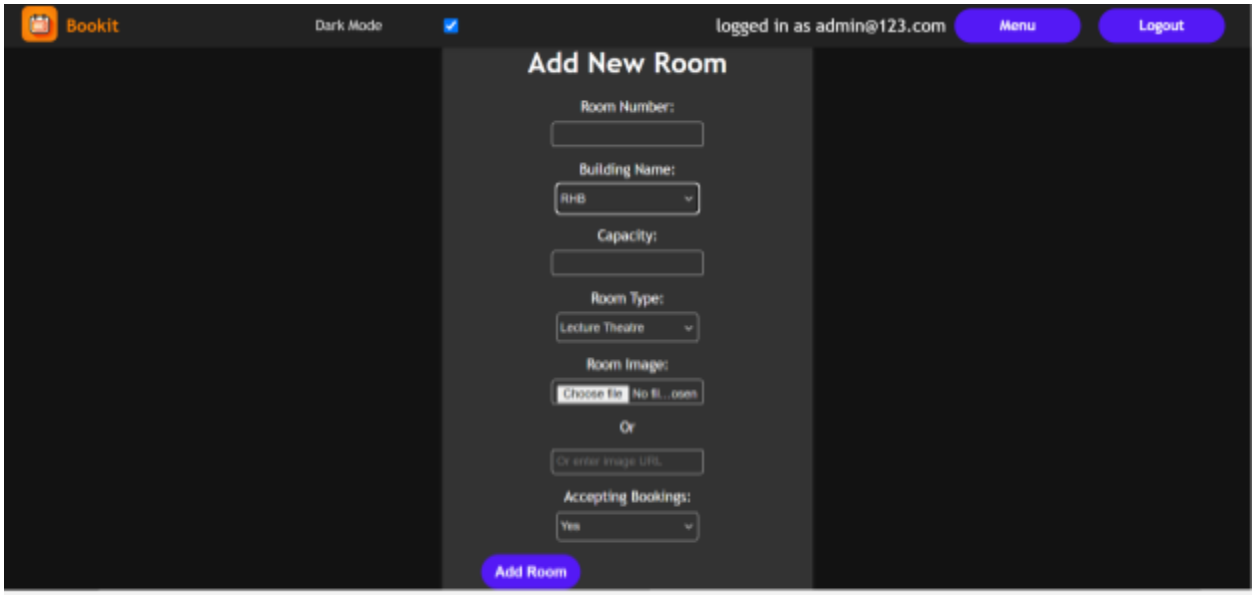


Figure 60: V1 UI for adding room

Going forward

It is possible to cause some errors by inputting a capacity that is not a number. Similarly, you can input a room number that actually is not a number - and while this will not cause any inherent problems in the system due to how we are handling the database data - it does break the continuity of the site.

Changing these fields to be sliders or “number only” input fields would rectify this problem, and we will consider this for future versions of Bookit.

Edit room list

Concept

This page is a variation on the rooms list page - but each card instead has an edit button that redirects to the edit page for that particular room entry in the back end. This page was not wireframed as it was conceptualised and developed during the production of V0 in response to further considerations to admin user functionality.

V0

For V0 - this page was not created - a route existed for it but the page content was exactly the same as the rooms list page, just without the added functionality for booking one of the rooms.

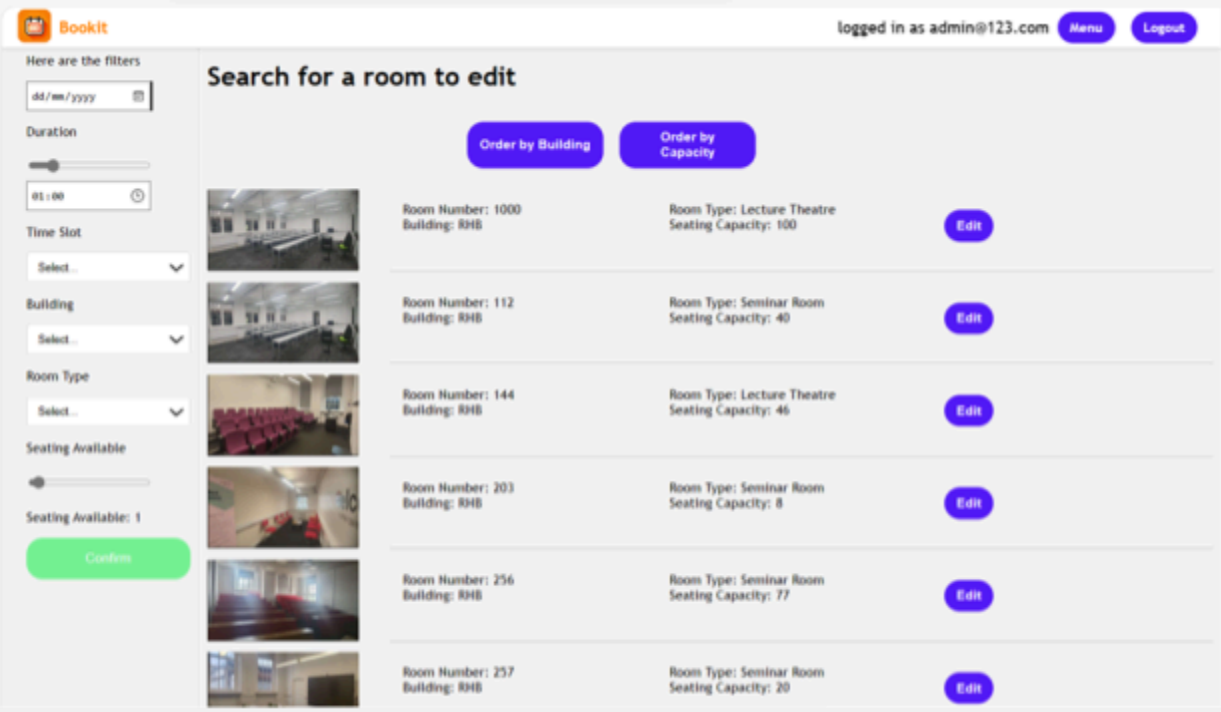


Figure 61: V0 UI for edit room list

V1

For V1 - we added a delete button along with the edit button so we could remove rooms from the list.

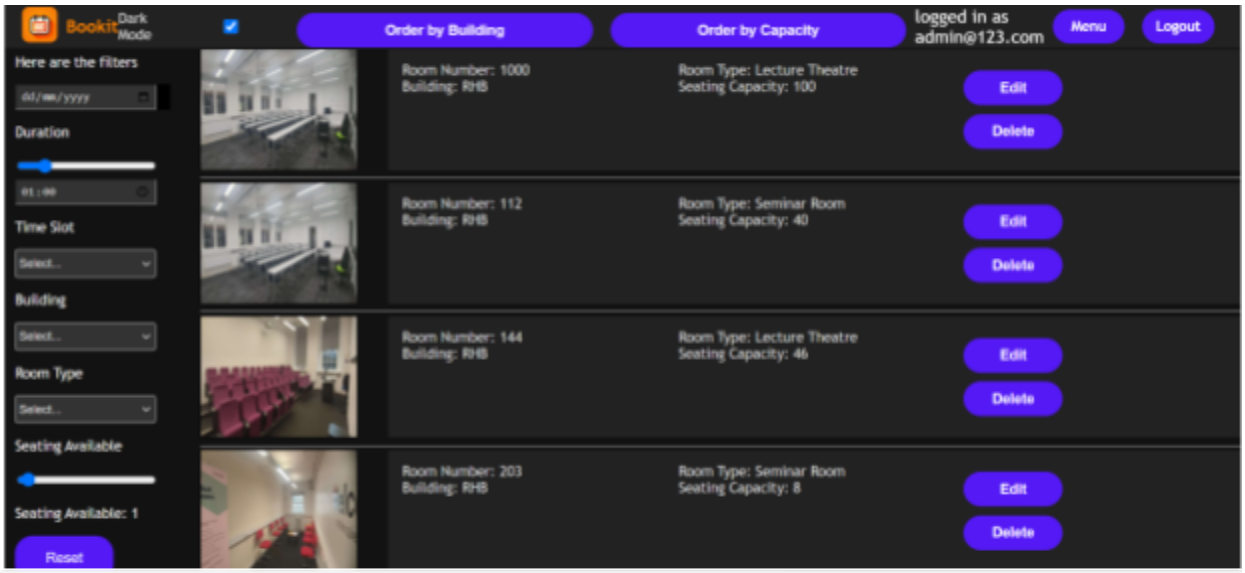


Figure 62: V1 UI for edit room list

Going forward

The filters for this page work - but most of them make no sense as they only apply to bookings (e.g., duration, time slot). Creating a separate filter just for the rooms list with a more limited scope of options would make more sense for this page.

Edit Room

Concept

This page is almost the same as add room - but instead of adding a new room entry to the back end it instead updates the entry for the particular room that is currently being edited (redirected from a specific room card on the "Edit room list" page). As was the case with the add room page, this page was not wireframed for the same reason.

V0

What was said about the add room page can also be said about this page for V0.

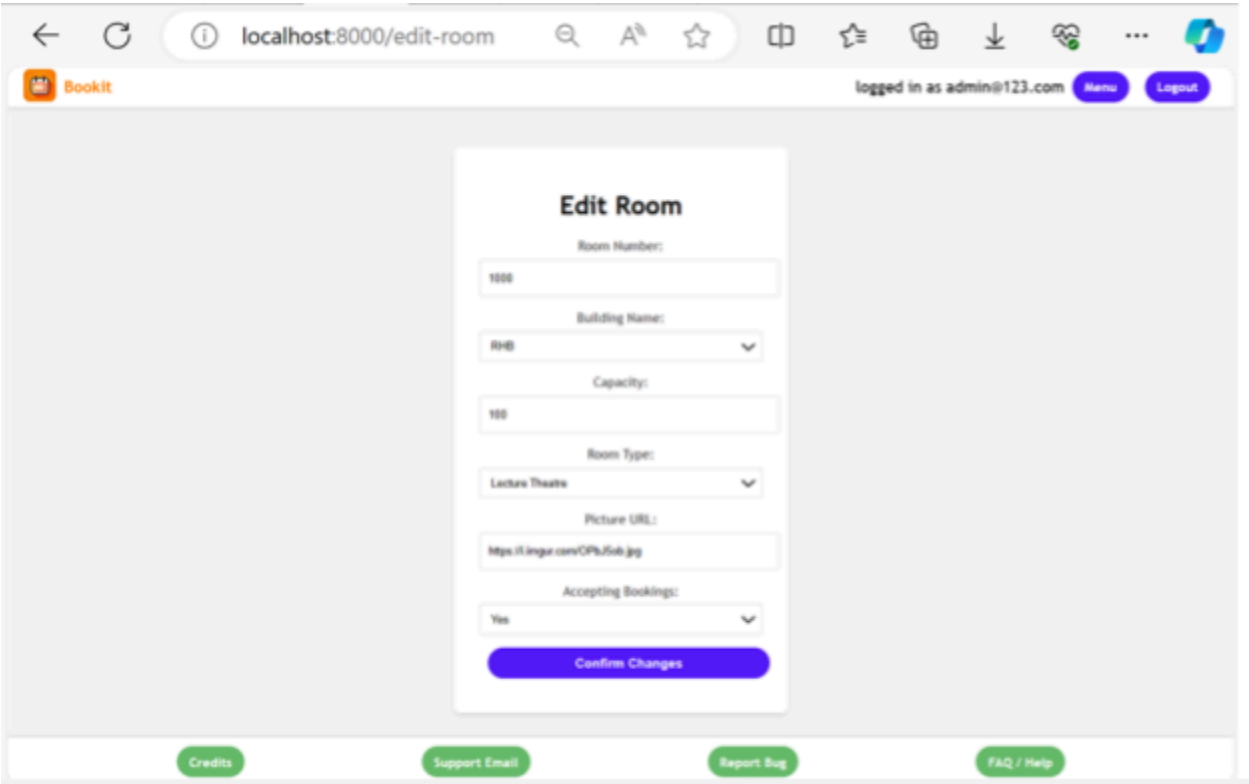


Figure 63: V0 UI for edit room

V1

The evolutions from V0 to V1 for edit room are exactly the same as they were for add room. The only difference is that "edit room" amends the database entries rather than adding them.

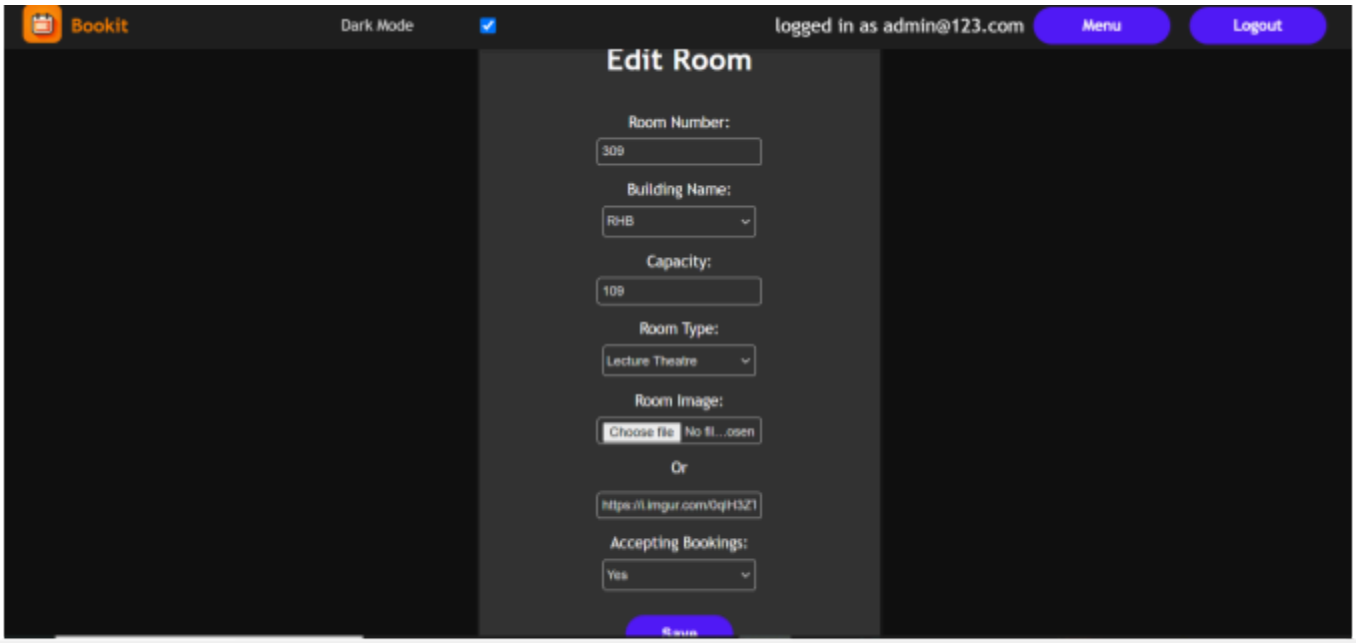


Figure 64: V1 UI for edit room

Going forward

See Add room - Going forward.

Credits Page

Concept

An easily legible list of all of our group members - those who were directly involved in the development of the source code.

V0

The V0 page was a simple unordered list of all the names.

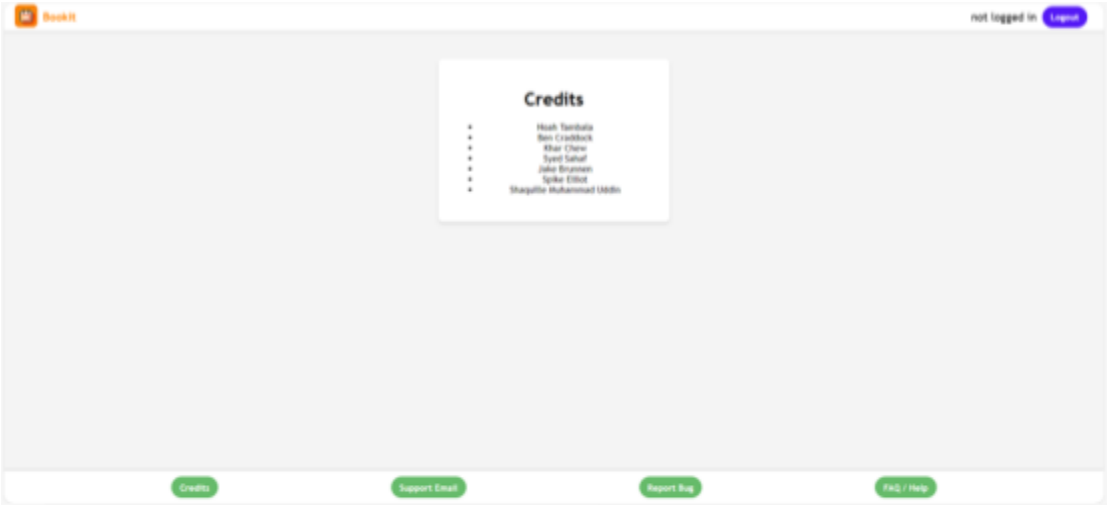


Figure 65: V0 UI for credits

V1

Our testers believed the format for the unordered list was ugly, so we redid the CSS for this page and removed the bullet points to their liking.

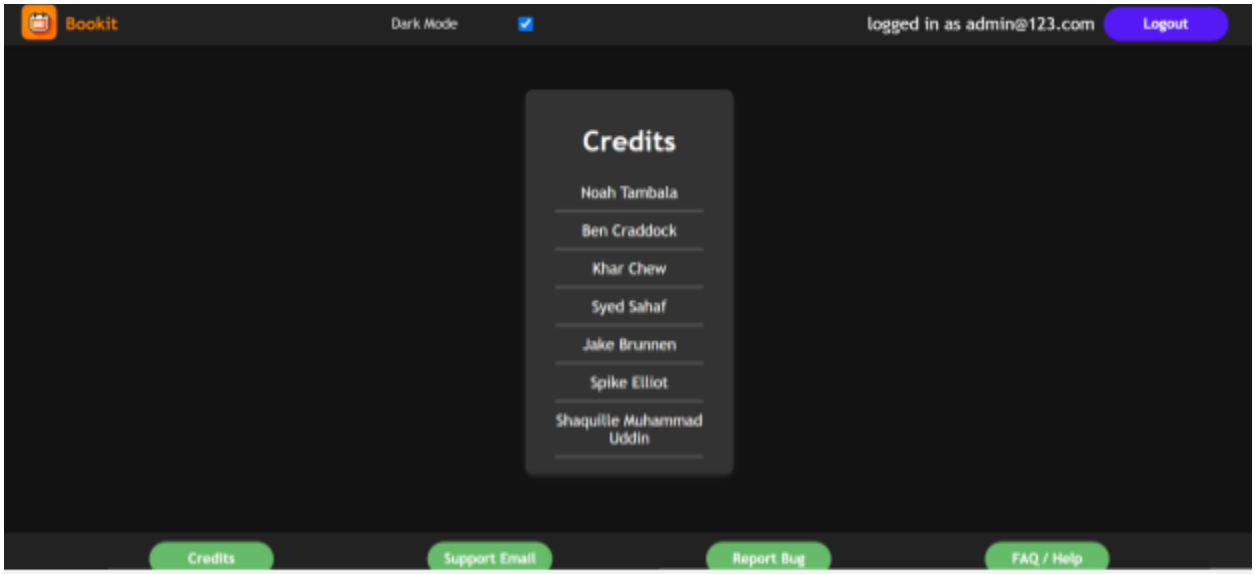


Figure 66: V1 UI for credits

Going forward

This page is complete.

FAQ / Help

Concept

A series of paragraphs with accompanying headings to provide somewhat of a basic user guide for users that are new to Bookit - pointing them in the right direction to help navigate the basic functionality of Bookit (e.g., “How do I book a room?”, etc.).

V0

In V0 the paragraphs were written but not styled.

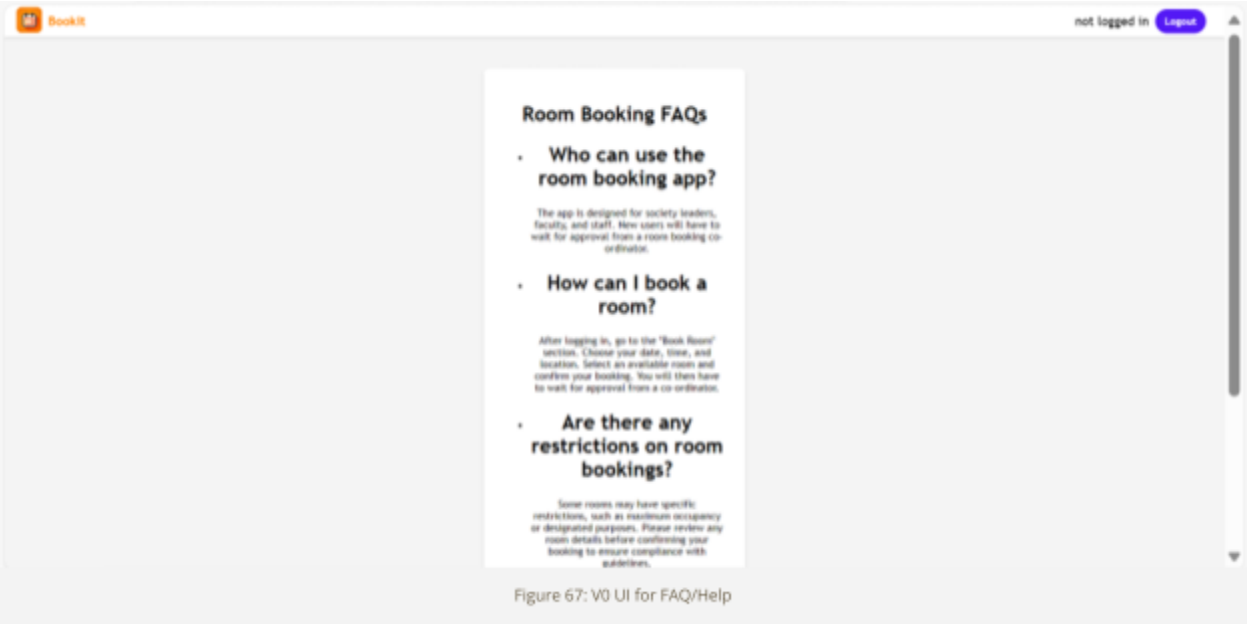


Figure 67: V0 UI for FAQ/Help

V1

For V1 we applied our basic CSS template and made the page look more in line with our card aesthetic.

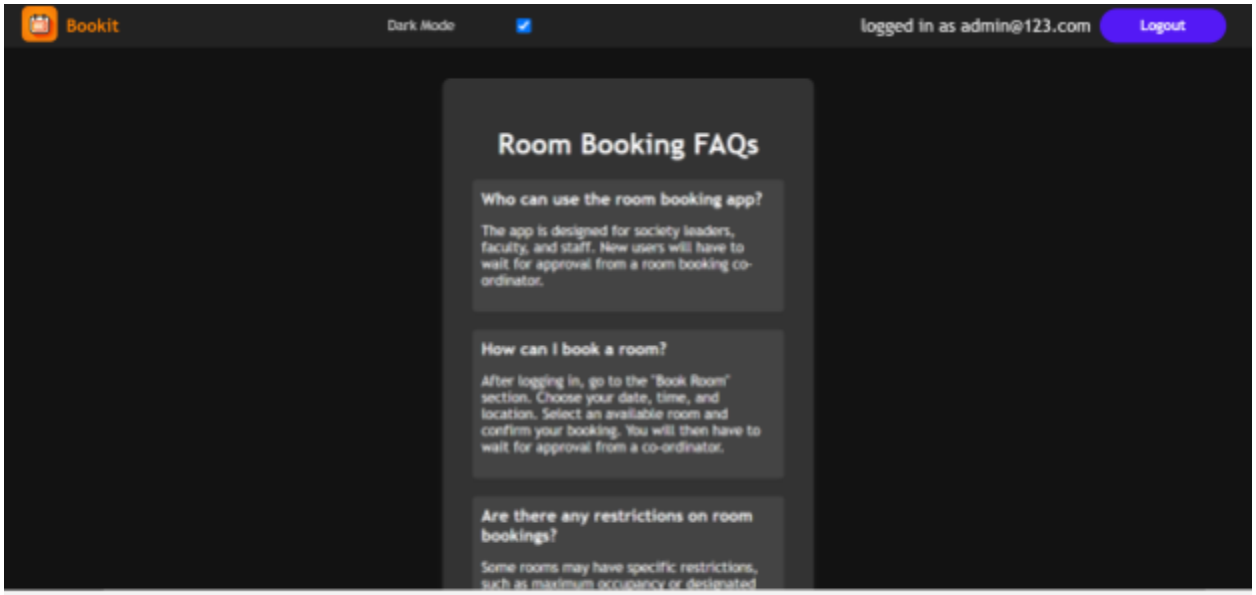


Figure 68: V1 UI for FAQ/Help

Going forward

The FAQs section is helpful for those just starting out with Bookit but does not provide comprehensive help (see “User guide”). Either adding more information or providing another page with a comprehensive user guide will help Bookit be more user friendly and accessible to those that may wish to use it.

Development

This section will detail how we developed the source code of bookit.

Source code

Github

Jake created a repository on Github called “roombooking” to develop the source code of Bookit. Every group member had access and we allocated tasks on a page-by-page basis so that we could split tasks effectively. This meant that there was minimal overlap between tasks and that changes made by one developer would be less likely to negatively impact previous changes made by another developer. Furthermore - this promoted modular development throughout our application.

For each development to the application a separate branch was created which was then committed to the main branch after completion - at which time the branches were then

merged to the main branch so the changes could be visible to all developers. Below is an example of this process with various changes to the main branch of the repo.

Figure 69: Examples of github commits

The repository contained the following structure:

- .vscode
    - Used for running the application directly from VSCode
- database-backups
    - Used for backups of the backend data
- dev-team-walkthroughs
    - Used for documentation on complicated changes to help other developers understand before progressing.
- media
    - Used for basic image files.
- public
    - Used to contain the communal css file used by all pages.
- rooms
    - Used for room image files
- security

    - Used for ZAP security reports (see security testing below)
- views
    - Used for ejs pages

The repository also contained the following vital files:

- .gitignore
    - Used to configure what parts of the project to ignore when committing changes
- LICENSE
    - Contains CC data for Bookit
- README.md
    - Important information for anyone installing Bookit locally
- app.js
    - Handles routing and middleware/logic
- create_db planetscale version.sql
    - Code to create the backend structure used for Bookit
- verify.js
    - Handles logic for the 2FA system

There is one key file that is not stored in the repository. This file is the .env file and it contains passwords and security configuration information that is not sensible to store on github where it is publicly accessible.

Production server .env file:

```
DATABASE_URL='mysql://0xg763coi21n6i5ig5u9:pscale_pw_Dm9yzopoMrN29dLvDWoXF650Mojtxa3tVNylxFCxxhi@aws.conne$
LOCAL_DB=true
LOCAL_HOST=localhost
LOCAL_USER=bookituser
LOCAL_PASSWORD=JuKi!s28;-24
LOCAL_DATABASE=roombooking
TWO_FACTOR=true
PRODUCTION_URL=https://www.doc.gold.ac.uk/usr/199/
```

Developer .env file:

```
DATABASE_URL='mysql://0xg763coi21n6i5ig5u9:pscale_pw_Dm9yzopoMrN29dLvDWoXF650Mojtxa3tVNylxFCxxhi@aws.connect.psdb.cloud/roombooking?ssl={"rejectUnauthorized":true}'
LOCAL_DB=false
TWO_FACTOR=false
```

Originally we handled bug documentation in our private discord server - where we also handled allocation of tasks. However this became hard to keep track of and led to an inefficient progression in development. We made this assessment after the production of V0 and therefore decided that we'd move to a Github project to log issues and allocate tasks. All bugs identified during unit tests and suggestions from user tests were logged as issues and then categorised as either "bugs", "enhancements", or "wontfix" - so we could prioritise them accordingly. Overall we opened 47 separate issues - which were labelled as "done" once complete. There are still some issues open to demonstrate further work that could be conducted on bookit that is not within the scope of the V1 MVP model - as well as further aggressive security tests that could be conducted in the event of a wider distribution of the Bookit application at Goldsmiths.

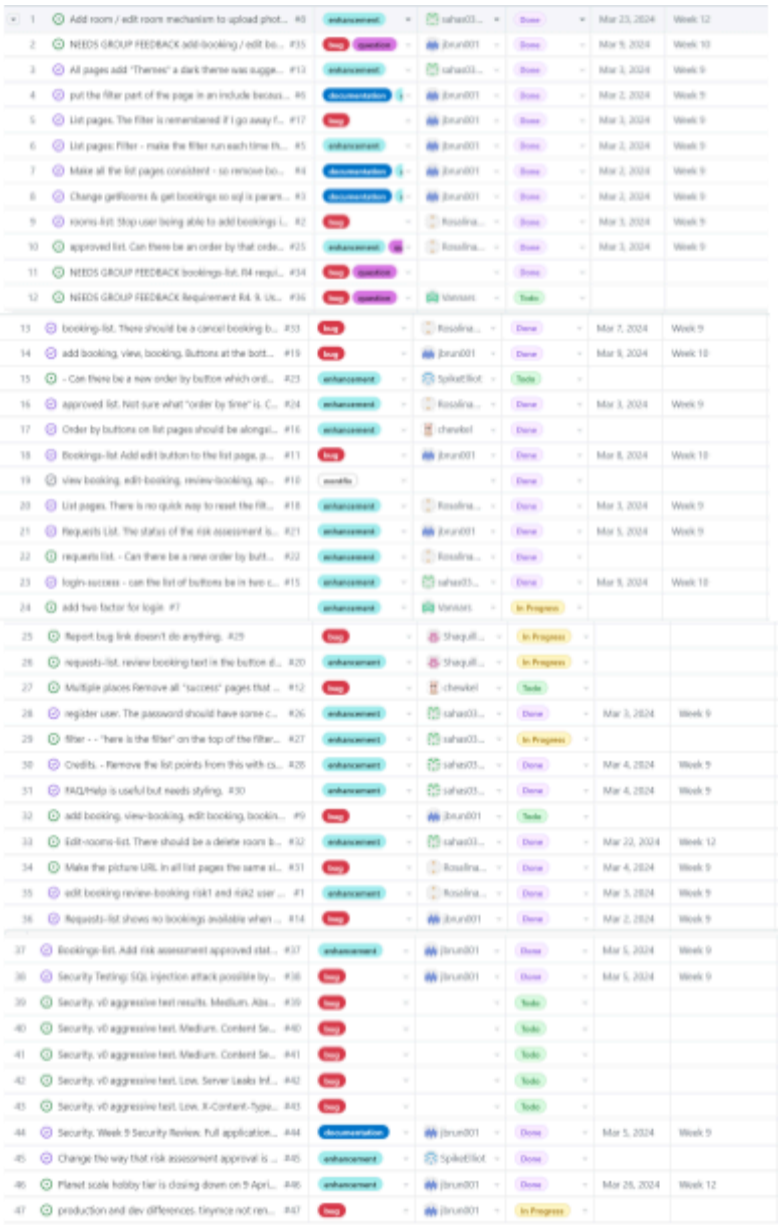


Figure 70: List of Github project log issues

To see a breakdown of development progress on a wee-by-week basis, please see the “Weekly sprints” section in the appendix.

Front end

When deciding how to approach our front end we decided on HTML paired with custom CSS as this was the method of front-end development our group was most familiar with - and seeing as we were going to need to allocate a lot of our time to learning new loading systems such as AJAX we decided that it was not worth allocating further time to applying a

new framework such as React or Bootstrap given that we were already proficient in producing custom CSS and HTML.

We handled all of our HTML pages using EJS templates. This meant we were able to use inline JavaScript alongside separate scripts. This allowed us to directly present the data from EJS requests in the middleware on the front end. An example of this is with how we handled providing the entries for the “Room type” drop down menu on the filter - as well as providing the inline logic for changing the text display for the selected seating capacity request in the same EJS template.

```
    <p>Room Type</p>
    <select placeholder="Field 2" id="roomType" name="roomType">
        <option value="">Select...</option>
        <% for (let i = 0; i < roomTypes.length; i++) { %>
            <option value="<%= roomTypes[i].room_type %>"><%= roomTypes[i].room_type %></option>
        <% } %>
    </select>
    <p>Seating Available</p>
    <input id="seatSlider" type="range" min="1" max="100" value="1" class="slider" name="seating" oninput="updateSeatValue(this.value)">
    <p id="seatValue">Seating Available: 1</p>
    <input id="filterReset" type="reset">
</form>
<script>
    function updateTimeDisplay(val) {
        var stringVal = "0" + Math.floor(val / 60) + ":" + ("0" + (val % 60)).slice(-2);
        document.querySelector("#timeDisplay").value = stringVal;
    }
    function updateTimeSlider(val) {
        var hoursMinutes = val.split(":");
        console.log(hoursMinutes);
        var totalTime = parseInt(hoursMinutes[0]) * 60 + parseInt(hoursMinutes[1]);
        document.querySelector("#timeSlider").value = totalTime;
        if (document.querySelector("#timeDisplay").value > "04:00") {
            document.querySelector("#timeDisplay").value = "04:00";
        }
        if (document.querySelector("#timeDisplay").value < "00:30") {
            document.querySelector("#timeDisplay").value = "00:30";
        }
    }
    function updateSeatValue(val) {
        document.getElementById('seatValue').innerText = 'Seating Available: ' + val;
    }
```

Figure 71: Room Type EJS example

These techniques are used frequently all throughout Bookit. Using inline JS meant we were able to directly manipulate the HTML of the page within the EJS file itself as opposed to handling it from a separate script file, giving us a more readable solution.

For page components that were used across multiple pages - such as the footer, we created a separate EJS template and included it using an inline script to ensure we reduced code repetition.

```
<body>
    <%- include('header-nomenu-nologout') %>
    <div class="content">
        <div class="welcome-container">
            <h1>Welcome to Bookit</h1>
            <p>Please Log in</p>
            <form action="/login-check" class="input-group-admin-pages" method="POST">
                <div class="input-group-admin-pages">
                    <label for="email">Email:</label>
                    <input type="email" id="email" name="email" required>
                </div>
                <div class="input-group-admin-pages">
                    <label for="password">Password:</label>
                    <input type="password" id="password" name="password" required>
                </div>
                <button type="submit" class="button-link">Log in</button>
            </form>
        </div>
    </div>
    <%- include('footer') %>
```

Figure 72: EJS footer template

While these front-end techniques were effective in building up a skeleton for Bookit - we realised that for transferring between pages with updated data from the middleware we were making far too heavy use of loading new pages. To rectify this, we employed the use of AJAX - to asynchronously update the page with fresh data from the middleware should that be necessary. An example of this is with how we updated the page data after filtering a list of bookings - we attached an eventlistener using a jQuery function to the document object of the filters component template and then used an AJAX request like the one below to update the page from the data fetched by the middleware.

```
// Attach the event listener to the form
$(document).ready(function() {
    // v1 submit the form every time the data change
    $('#filterForm').find('input, select, range').on('change', updateListData);
    $('#filterForm').on('reset', function() {
        updateSeatValue(1);
        // 5ms delay to allow the form to reset before handling defaults:
        setTimeout(function() {
            // update list with default values:
            updateListData();
        }, 5);

    });
    function updateListData() {
        //$('#filterForm').submit(function(e) {
        //    e.preventDefault(); // Prevent default form submission
        $.ajax({
            // Send the form data to the server
            type: "POST",
            // url: $(this).attr('action'),
            url: '/' + window.location.pathname.split('/').pop() + '-filtered',
            // data: $(this).serialize(), // Serialize form data for submission
            data: $('#filterForm').serialize(),
            success: function(response) {
                // Update the bookings list with the response
                $('.data-list').html($(response).find('.data-list').html());
                // remove the new data animation if it has already been added
                $('.data-list').removeClass('newDataLoadedEffect');
                // force the animation to run again
                void $('.data-list')[0].offsetWidth;
                // re-add the animation on the new data
                $('.data-list').addClass('newDataLoadedEffect');
            }
        });
```

Figure 73: jQuery event listener for updating pagec

While we were initially unfamiliar with AJAX and jQuery as a group - the level of documentation and online guides surrounding these technologies made learning how to implement them efficient and easy to understand, and therefore our unfamiliarity had little effect on our development progress as these frameworks are heavily used in the web development scene.

Middleware

Middleware is the figurative “central party” in ensuring that requests to the back end return meaningful data so that it can be presented to the user on the front end. Our middleware is handled through express - or EJS - written in JavaScript. EJS handles traversal throughout the application by executing certain code when certain URLs are received by the browser -

via GET requests as seen below - the EJS application is defined in the variable “app” which handles all routing throughout Bookit:

```
require("dotenv").config();
const speakeasy = require("speakeasy");
const qrcode = require("qrcode");
const express = require("express");
var path = require('path');
const app = express();
const port = process.env.PORT || 8000;
var session = require("express-session"); // used for creating sessions so data can persist between pages like if a user is logged in
const mysql = require("mysql2");
var ejs = require("ejs");
const sanitiseHtml = require("sanitize-html"); // used to make sure there are no cross site scripting issues with the input
const bodyParser = require("body-parser");
const bcrypt = require("bcrypt"); // used for storing and comparing password hashes
const bcryptSaltRounds = 10; // the higher this is the more processing time - 20 is unworkable takes too long so changed to 10
const multer = require('multer');
```

```
/**
 * requests-list route
 * this route is used to display the bookings that are awaiting approval by the co-ordinator
 * requests-list-filtered route is used by this rendered page to filter the list when
 * the filter is applied
 *  */
app.get("/requests-list", isLoggedIn, (req, res) => {
```

Figure 74: Examples of middleware

EJS also handles POST requests - which are used to execute code upon a specific redirect. This is useful for bookit when the user submits a form and this redirects to a separate link - we can make an AJAX call to render the new data fetched by that redirect - such as when the requests list gets filtered:

```
/**
 * requests-list-filtered route
 * this route is the target for the filter form, and the order by form in the requests-lists page
 */
app.post("/requests-list-filtered", isLoggedIn, function (req, res) {
```

Figure 75: Example of POST used for filtering requests list

As the EJS middleware acts as a “bridge” between the front end and the backend - there needs to be a way to fetch data from the back end from this JavaScript logic. You may have noticed in the above Figure that we are using many required node_modules on our NodeJS server. One of these is mysql12. Seeing as during the project proposal stage of this module we decided to go for a relational database system - we use SQL queries in the middleware

to fetch data from the back end. An example of how this process works is with the “Order by” buttons on the list pages for rooms and bookings. As seen in the figure below.

Figure 76: “Order by” buttons

Clicking one of these buttons submits the below HTML form:

```
<div class="order-by-button-wrapper">
<form id = "orderForm" class="card-wrapper" action="/rooms-list-filtered" method="POST">
    <button class="button-link-order" name="orderSelection" type="submit" value="o_b_Building">Order by Building</button>
    <button class="button-link-order" name="orderSelection" type="submit" value="o_b_Capacity">Order by Capacity</button>
</form>
```

Figure 77: “Order by” buttons HTML form

Which is then passed to the middleware via a POST request to /rooms-list-filtered - which then checks if a value for a clicked “Order by” button exists (this post request handles both the filters for the rooms list page as well as the “Order by” buttons):

```
// route for the filter and ordering in the rooms-list page
// this route is re-used for filtering AND ordering, if there no req.body.orderSelection
// then we are processing filtering, otherwise we are processing ordering
app.post("/rooms-list-filtered", isLoggedIn, function (req, res) {
  loggedInMessage = getLoggedInUser(req);
  var userrole = req.session.user_role;
  var userId = req.session.userid;
  var email = req.session.email;
  var filters = {};
  // work out if we are posting to filter the list or order the list
  // if the POST doesn't contain req.body.orderSelection then we are have a post that is filtering
  if (!req.body.orderSelection) {
    var startingTimeslot = req.body.timeslot.replace("starting from ", "");
    var bookingDuration = req.body.durationRange;
    var bookingHours = Math.floor(bookingDuration / 60);
    var bookingMinutes = bookingDuration % 60;
    var startingTimeSplit = startingTimeslot.split(":");
    var endingTimeslot =
      String(parseInt(startingTimeSplit[0]) + bookingHours).padStart(2, "0") +
      ":" +
      String(parseInt(startingTimeSplit[1]) + bookingMinutes).padStart(2, "0");
    filters = {
      date: req.body.date,
      timeslot: startingTimeslot + "-" + endingTimeslot,
      building: req.body.building,
      roomType: req.body.roomType,
      minSeats: req.body.seating,
      duration: req.body.durationRange,
    };
    // save the current filters in the user session, this is so they can be re-used when there is no filter POSTED
    req.session.roomListFilters = filters;
  } else {
    filters = req.session.roomListFilters;
    // save the orderSelection
    req.session.roomListOrder = req.body.orderSelection;
```

Figure 78: checks if a value for a clicked “Order by” button exists

The value from the HTML form is then handled by this EJS POST request and converted into an SQL query component that changes how the data is fetched from the back end:

```
// get the current list order from the session and if not present initialise the order
// if (!req.session.roomListOrder) req.session.roomListOrder = ""
// manage the ordering of the data - if the user has re-ordered the list
var listOrder = "";
//console.log("session.roomListOrder: " + req.session.roomListOrder);
switch (req.session.roomListOrder) {
  case "o_b_Capacity":
    listOrder = " ORDER BY r.capacity DESC";
    break;
  case "o_b_Building":
    listOrder = " ORDER BY r.building_name, r.room_number";
    break;
  default:
    listOrder = " ORDER BY r.building_name, r.room_number";
    break;
}
var selectedTimeslot = "";
var selectedDate = "";
if (filters.timeslot != "-NaN:NaN" && filters.date != "") {
  selectedTimeslot = filters.timeslot;
  selectedDate = filters.date;
}
console.log(filters);
```

Figure 79: SQL query component that changes how the data is fetched from the back end

This string is then concatenated to the overall query to fetch the data - this code snippet shows how this “Order by” string is passed to the getRooms() method to make a call to the database:

```
    // add the "order by" string
    sqlquery = sqlquery + " " + listOrder;

    // execute sql query
    db.query(sqlquery, sqlParameters, (err, results) => {
      if (err) {
        console.error(err.message);
        reject(err); // if there is an error reject the Promise
      } else {
        resolve(results); // the Promise is resolved with the result of the query
      }
    });
```

Figure 80: database giving a response to the middleware

This also works when the database gives a response to the middleware as seen above - the database will either respond with an error or a meaningful result - at which point the middleware will process the response and then render the page with that relevant data. The “Order by” form also demonstrates this with how it resolves a promise based on the results returned by the SQL query.

```
Promise.all([
  getRoomTypes("rooms-list", userId), // Promise.all[0]
  getBuildingNames("rooms-list", userId), // Promise.all[1]
  getRooms("rooms-list",filters, listOrder,""), // Promise.all[2]
])
  .then(([roomTypes, buildingNames, rooms]) => {
    // if you had more data calls above you name it here, the first variable is the result of promise.all[0] etc.
    console.log(filters);
    res.render("rooms-list.ejs", {
      loggedInMessage,
      userrole,
      email,
      rooms,
      roomTypes,
      buildingNames,
      selectedTimeslot,
      selectedDate,
    });
  })
  .catch((error) => {
    console.log(
      "Error getting data from database calls or in the code above"
    );
  });
);
```

Figure 81: "Order by" resolving a promise

In addition to this - the middleware also handles specific data for the instance Bookit gets loaded. The below code snippet shows a couple of examples of this - creating a unique and persistent session that a given user can log into to access the main functionality of the app

- of which will persist across all browser/page traversal. This session will contain important data such as the session key and the cookies for that user to ensure we stay in line with our defined security plan. The below code snippet also shows how we apply certain node modules to the express application like body_parser which is vital for Bookit as it allows for the handling of HTML form data like in the “Order by” example above.

```
// session middleware - used for login, used to create a unique persistent session
app.use(
  session({
    secret: "secretforumkey",
    resave: false,
    saveUninitialized: true,
    cookie: {
      sameSite: "strict",
    },
  })
);

// this allows processing of form data
// extended the limit to to 500kb so the post of the risk assessment is managed
// the risk assessment template is 223kb, so this allows for just over double the size
app.use(bodyParser.urlencoded({ limit: '500kb', extended: true }));
```

Figure 82: applying node modules to the express application body_parser

When it comes to exception handling - Bookit employs several techniques to avoid crashes. One of these is standard try/catch blocks to ensure that errors for small sections of code such as null pointer exceptions do not halt the entire application. An example of this is how we used a try/catch block to ensure a valid hash is generated when using the node module bcrypt to encrypt our plain text passwords. This is once again to ensure we maintain a good level of security for Bookit. Returning null if there is some error ensures we can check for a valid hash later on.

```
// function to hash a plain text password
function hashPassword(password) {
  try {
    var hash = bcrypt.hashSync(password, bcryptSaltRounds);
    return hash;
  } catch (error) {
    return null;
  }
}
```

Figure 83: hashing password

Another way Bookit handles exceptions has to do with SQL queries. If there is some error in how the query is generated then the query won't be able to execute. To solve this, we handle the error by rejecting the given promise should the query not execute for whatever reason - which ensures the app does not crash despite not reaching the desired outcome. An example of this is when Bookit has to update a risk assessment on a previously made booking, as seen in the code snippet below.

```
// execute sql query
console.log("insertUpdateRiskAssessment: " + sqlquery);
db.query(sqlquery, changedDataFields, (err, results) => {
  if (err) {
    console.error(err.message);
    reject(err); // if there is an error reject the Promise
  } else {
    resolve(results); // the Promise is resolved with the result of the query
  }
});
);
```

Figure 84: Errors for updating risk assessmentc

A similar process is followed for when a list of rooms or bookings is filtered - however the syntax varies significantly. In this situation, we had to make sure we caught the error should the promise for resolving the filters not be fulfilled for whatever reason - using a .catch statement as opposed to rejecting the promise all together. This can be seen in the code snippet below.

```
const filters = {};
Promise.all([
  getRoomTypes("edit-room", userId),                  // Promise.all[0]
  getBuildingNames("edit-room", userId),              // Promise.all[1]
  getRooms("edit-room",filters, listOrder,roomId),          // Promise.all[2]
])
  .then(([roomTypes, buildingNames, rooms]) => {
    // if you had more data calls above you name it here, the first variable is the result of promise.all[0] etc.
    console.log(filters);
    res.render("edit-room", {
      loggedInMessage,
      userrole,
      email,
      rooms,
      roomTypes,
      buildingNames
    });
  })
  .catch((error) => {
    console.log(
      "Error getting data from database calls or in the code above"
    );
  });
);
```

Figure 85: .catch statement

A further example of exception handling is with our use of the node module sanitise-html. This ensures that any forms that are submitted by the user follow the defined structure to prevent injection and other errors. An example of this can be seen from the app.js snippet below to show how the room data is displayed when a room is successfully edited by an admin.

```
app.post("/edit-room-success", isLoggedIn, upload.single('roomImageFile'), (req, res) => {
  //sql changes to room table
  var userrole = req.session.user_role;
  var email = req.session.email;
  var userId = req.session.userid;
  const roomId = parseInt(sanitiseHtml(req.body.roomId));
  const roomNumber = sanitiseHtml(req.body.roomNumber);
  const buildingName = sanitiseHtml(req.body.buildingName);
  const roomType = sanitiseHtml(req.body.roomType);
  const capacity = sanitiseHtml(req.body.capacity);
  let pictureURL = req.body.pictureURL ? sanitiseHtml(req.body.pictureURL) : null;
  const isAcceptingBookings = sanitiseHtml(req.body.isAcceptingBookings);
  console.log("edit room sucess: " + roomId);
```

Figure 86: app.js snippet for room display

We also use our middleware to handle uploading files for creating/editing room entries through the node module “multer”. Further exception handling is required here, we use a regex expression to ensure the file uploaded includes a common image file extension,

otherwise the image file for the room is not processed. The below code snippet demonstrates this.

```
const storage = multer.diskStorage({
  destination: function (req, file, cb) {
    cb(null, 'media/rooms/')
  },
  filename: function (req, file, cb) {
    const roomNumber = req.body.roomNumber;
    const buildingName = req.body.buildingName;
    const filename = `${roomNumber}-${buildingName}.jpeg`;
    cb(null, filename);
  }
});

// Accept image files only
const imageFilter = function(req, file, cb) {
  if (!file.originalname.match(/\.(jpg|jpeg|png)$/i)) {
      return cb(new Error('Only image files are allowed'), false);
  }
  cb(null, true);
};

const upload = multer({ storage: storage, fileFilter: imageFilter });
```

Figure 87: multer middleware

Furthermore, we use middleware for security purposes using Speakeasy for Two-Factor Authentication.For instance speakeasy has inbuilt functions for the generation and verification of secret-tkeys from parameterized input as is seen in the code snippets above/below.

```
//Two Factor Authentication
//THIS VARIABLE ACTIVATES TWO FACTOR ACROSS THE ENTIRE APP <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
let activateTwoFactor;
activateTwoFactor = true;

// override this value if there is a TWO_FACTOR=TRUE or TWO_FACTOR=FALSE in the .env file
if (process.env.TWO_FACTOR) {
  let mfa = process.env.TWO_FACTOR.toLowerCase();
  // have to do this as mfa is a string not a boolean
  if (mfa == "true") activateTwoFactor = true
  else if (mfa == "false") activateTwoFactor = false
  else console.log("INFO: invalid values for TWO_FACTOR in .env - TRUE and FALSE are the only valid values")
}

console.log("Two factor authentication activation is " + activateTwoFactor);

// 1) Make a secret key const generatedSecret using generateSecretKey() which returns secret, which generates a key based on the argument (user email)
// 2) Create a QR code URL using the qrToURL function which takes a secret key as its argument. This will be used to diplay the qr code as an image in the page
// 3) User scans the qr code using google authenticator - a user should input the code generated into a post box on the log in page,  which updates token in a POST method
// 4) Run verified on the ascii value of generatedSecret against the userinput taken from the GET method

function generateSecretKey(email){
  this.email = String(email);
  var secret = speakeasy.generateSecret({
    name: String(this.email)
    });
    return secret;
}
```

Figure 88: speakeasy middleware

```
// This function takes two arguments, user input and user secret. These will be session based and defined in the GET method and used in the Post method to verify the user.
//speakeasy does the heavy lefting here - we are using ascii encoding for our keys
function verifyKey(token , userSecret){
  var verified = speakeasy.totp.verify({
    secret: String(userSecret),
    encoding: 'ascii',
    token: token
  })
  return verified;
}
```

Figure 89:verification of secret-tkeys

We use these to verify the content taken from the form data in the login page allowing us to have 2fa for all login attempts. In the snippet below for instance you can see the post response for /login-tfa takes session data to generate the unique QR codes using the middleware functions secretKey and qrCodeUrl.

```
app.post("/login-2fa", async (req, res) =>{
  qrCodeUrl = req.session.qrCodeUrl // references the session qr 
  secretKey = req.session.generatedSecret.ascii // references the
  const userToken = req.body.twoFactorCode; // references user in

  // Verify the two-factor authentication code
  const isTokenValid = verifyKey(userToken, secretKey); // the de

  if (isTokenValid) { // login if true
    req.session.isLoggedIn = true; // Set the session variable to
    loggedInMessage = getLoggedInUser(req);
    res.redirect("login-success");
  } else {
    loggedInMessage = "Two-factor authentication failed, try agai
          // render the original 2fa page again with the orinal s
    res.render("login-2fa.ejs", { loggedInMessage, qrCodeUrl });
  }
});

app.get("/login-error", function (req, res) {
  loggedInMessage = getLoggedInUser(req);
  res.render("login-error.ejs", { loggedInMessage });
});

app.get("/logout", function (req, res) {
  req.session.destroy((err) => {
    if (err) {
      console.error("Error destroying session:", err);
      res.sendStatus(500); // Internal Server Error
    } else {
      loggedInMessage = "not logged in";
      res.render("login.ejs", { loggedInMessage });
    }
  });
});
```

Figure 90: post response for /login-tfa

Through these middleware techniques and exception handling - along with our use of AJAX on the front end - we ensure a fast, robust and secure experience for Bookit users.

Back end

As detailed in our project proposal as well as previously - we used MySql for our backend database as it was the backend framework with which we were most familiar as a group. As there were so many aspects to bookings, rooms and users alike - we decided to create a multi-table cross-parameterized database. Even though our prototype version doesn't include all the rooms/buildings available for booking at Goldsmiths we wanted to "future-proof" our back end - we therefore normalised our database to 3NF form and extracted different data to separate tables. This removes data duplication.

Overall - our database has 8 tables.

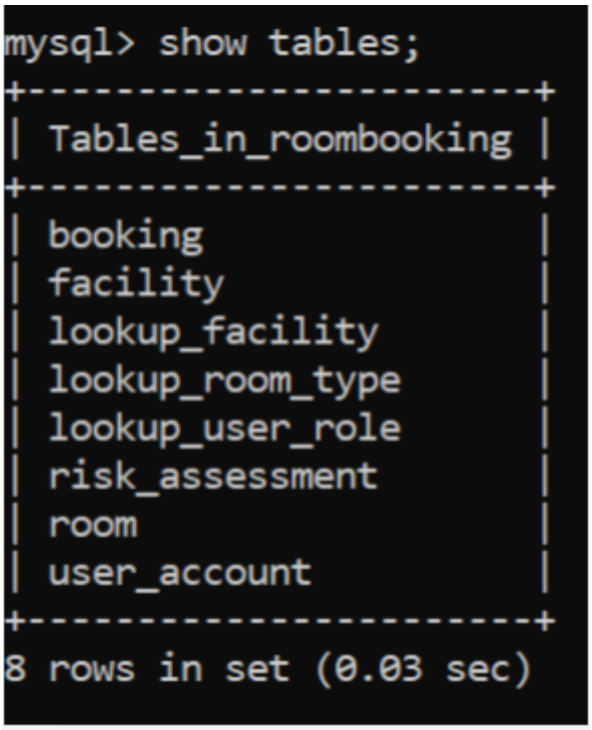


Figure 91: database with list of tables

The below figure demonstrates the normalisation of our database with the "booking" table. The primary key of the booking table is the "id" field. The booking table has two foreign key relationships with the "user" and "room" tables with their respective id fields. This means that we do not need to repeat the room and user data when it comes to referencing a booking that a user made for a particular room - as we can just analyse the id of the relevant user and room within the booking table entry - thus eliminating data repetition.

```
mysql> DESCRIBE booking;
+--------------------------------+-------------+------+-----+-------------+----------------+
| Field                          | Type        | Null | Key | Default     | Extra          |
+--------------------------------+-------------+------+-----+-------------+----------------+
| id                             | int         | NO   | PRI | NULL        | auto_increment |
| booking_start                  | datetime    | YES  |     | NULL        |                |
| booking_end                    | datetime    | YES  |     | NULL        |                |
| booking_reason                 | varchar(50) | YES  |     | NULL        |                |
| booking_status                 | varchar(50) | YES  |     | NULL        |                |
| risk_assessment_approval_status | varchar(50) | YES  |     | Not Reviewed |                |
| confirmed_on                   | datetime    | YES  |     | NULL        |                |
| cancelled_on                   | datetime    | YES  |     | NULL        |                |
| user_id                        | int         | YES  |     | NULL        |                |
| room_id                        | int         | YES  |     | NULL        |                |
+--------------------------------+-------------+------+-----+-------------+----------------+
10 rows in set (0.04 sec)
```

Figure 92:normalisation of database with the "booking" table

Testing

As part of our agile development techniques we decided to place most of our testing focus on user testing. According to Cedric Bach, user testing is the most efficient method for ensuring you get a user experience for your application that caters to the needs of your stakeholders and users - finding it to be more effective than document-based inspection and expert inspection when performed vigorously on an entire digital system.[23] With this in mind - we conducted two sets of user tests. One after the production of V0 to see what we should change for V1, and one after the production of V1 to see what should be included/altered for future versions of Bookit.

User tests

Version 0 user testing and feedback

[23] Bach, Cedric, and Dominique L. Scapin. "Comparing inspections and user testing for the evaluation of virtual environments." Intl. Journal of human–computer interaction 26.8 (2010): 786-824. (Abstract)

Login

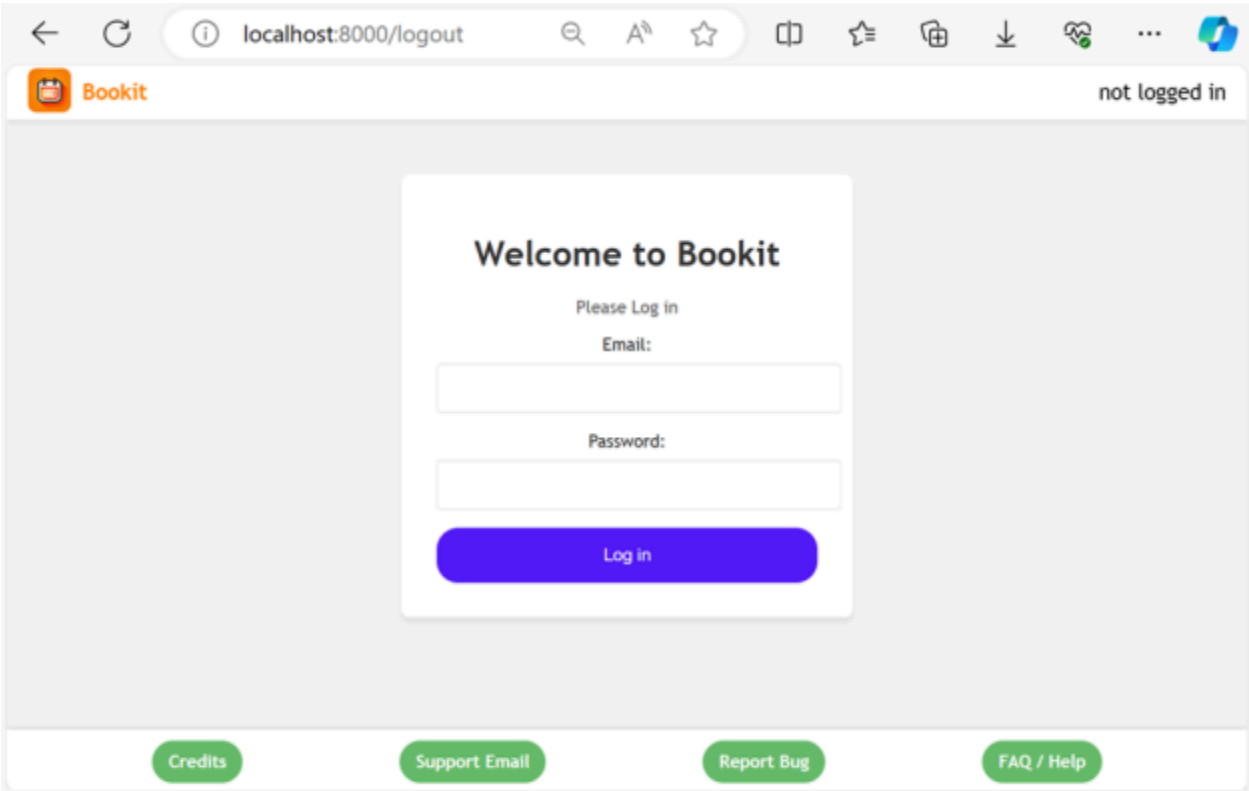


Figure 93: User Test V0 login

User testing feedback

- Can there be a dark mode theme? Logged as backlog issue 13.
- Can there be two factors for login? Logged as backlog issue 7.

Login Success

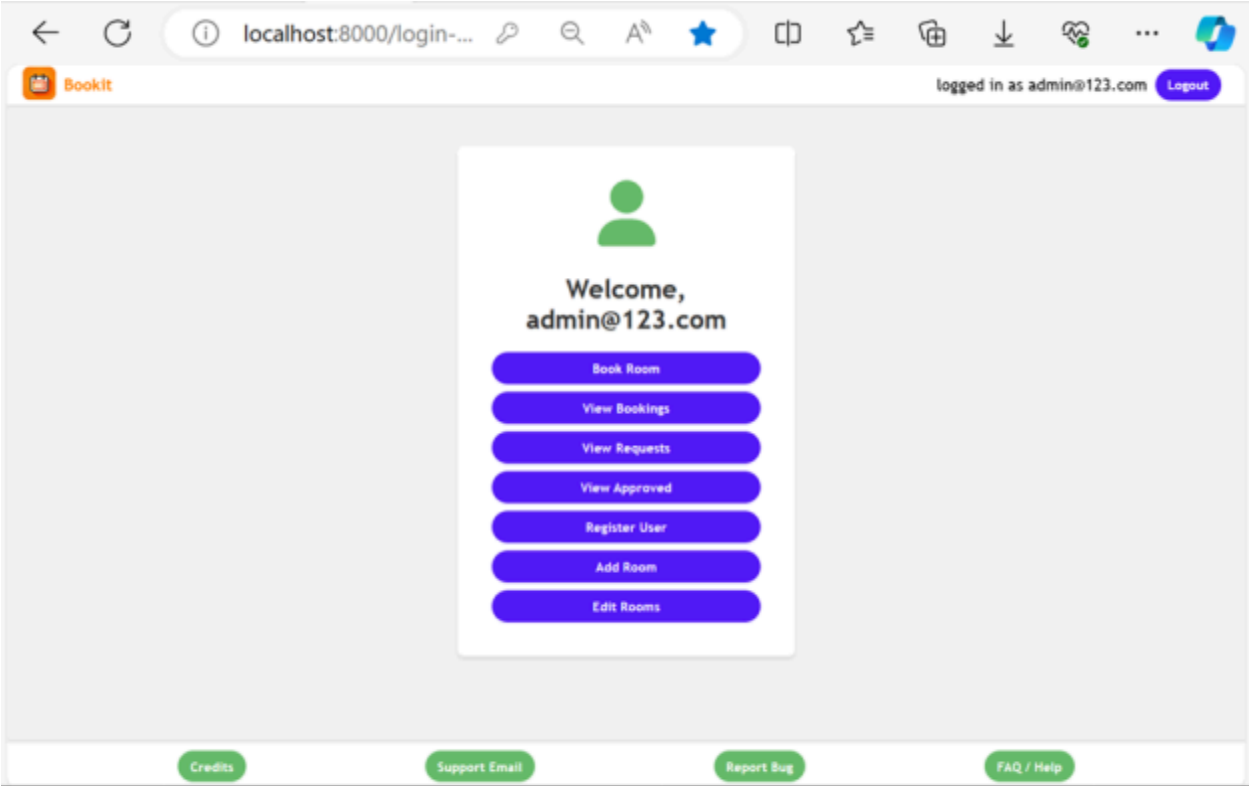


Figure 94: User Test V0 login success

User testing feedback:

- This menu is too long down the page so there are two columns so I don't need to scroll. Logged as backlog issue 15
- How are users notified of successful bookings? Logged in backlog as issue 36.

Rooms List

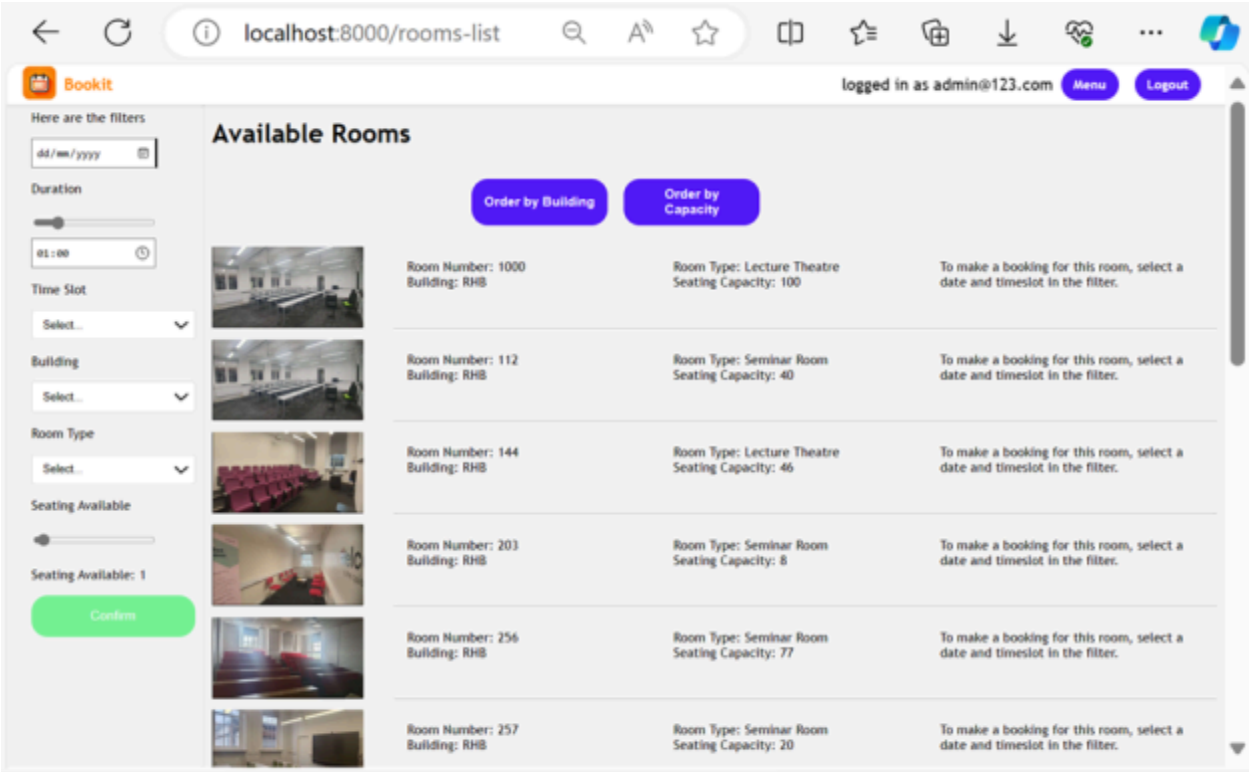


Figure 95: User Test V0 room list

User testing Feedback

- Can you make the list change when the fields in the filter change – I don't want to have to keep clicking confirm. Logged as backlog issue 5.
- Move the buttons to sort the data alongside the heading, they take up too much space at the moment. Logged as backlog issue 16.
- The filter is remembered if I go away from the page and come back, but the fields of the filter always start reset. This is confusing. Either reset the filter each time the user opens the page or populate the filter fields with the filter data. Logged as backlog issue 17.
- There is no quick way to reset the filter – can there be a button to reset the filter, so all list items show. Logged as backlog issue 18.
- "Here is the filter" on the top of the filter doesn't seem professional. Logged as backlog issue 27.
- One room picture, room 309, is longer than the others. Can all the room pictures be the same size but also scale if I change my browser settings for zoom. Logged as backlog issue 31.

Rooms list - with booking about to be requested

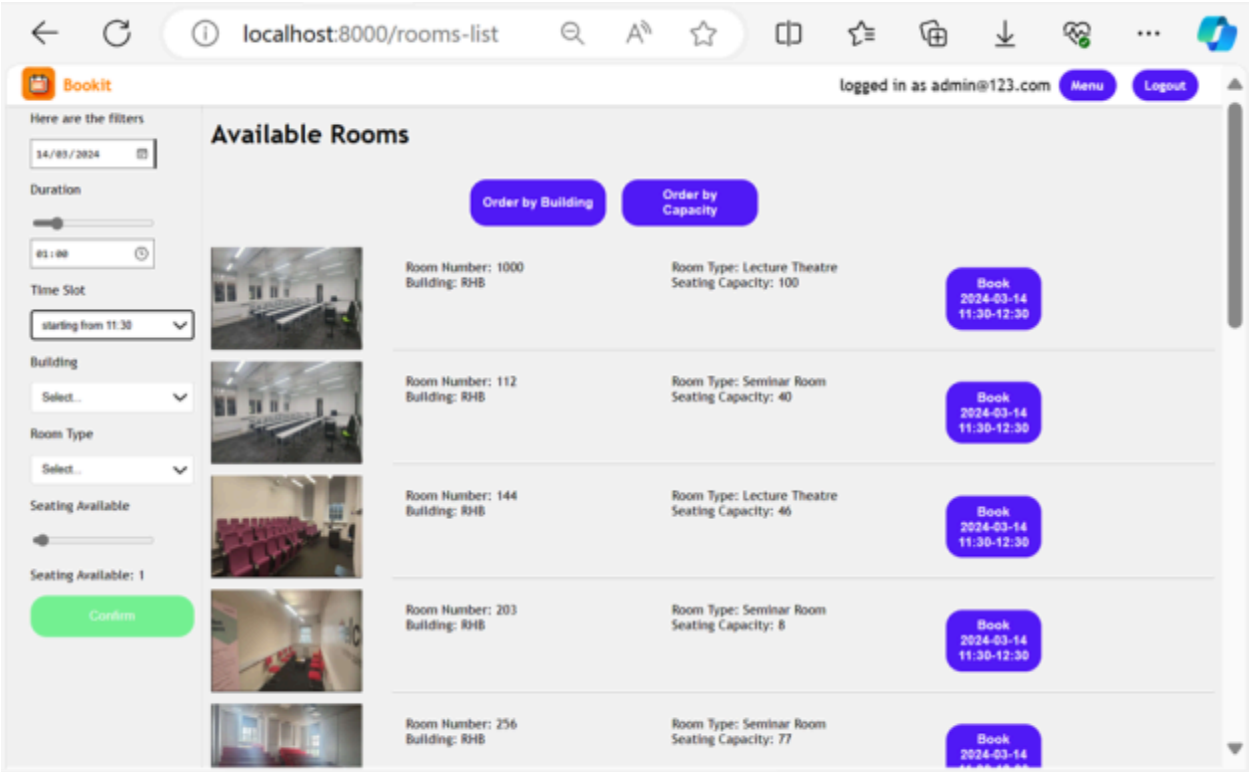


Figure 96: User Test V0 room list request

User testing feedback

- I can book a room for dates in the past. This shouldn't be allowed. Logged as backlog issue 2.

Add Booking

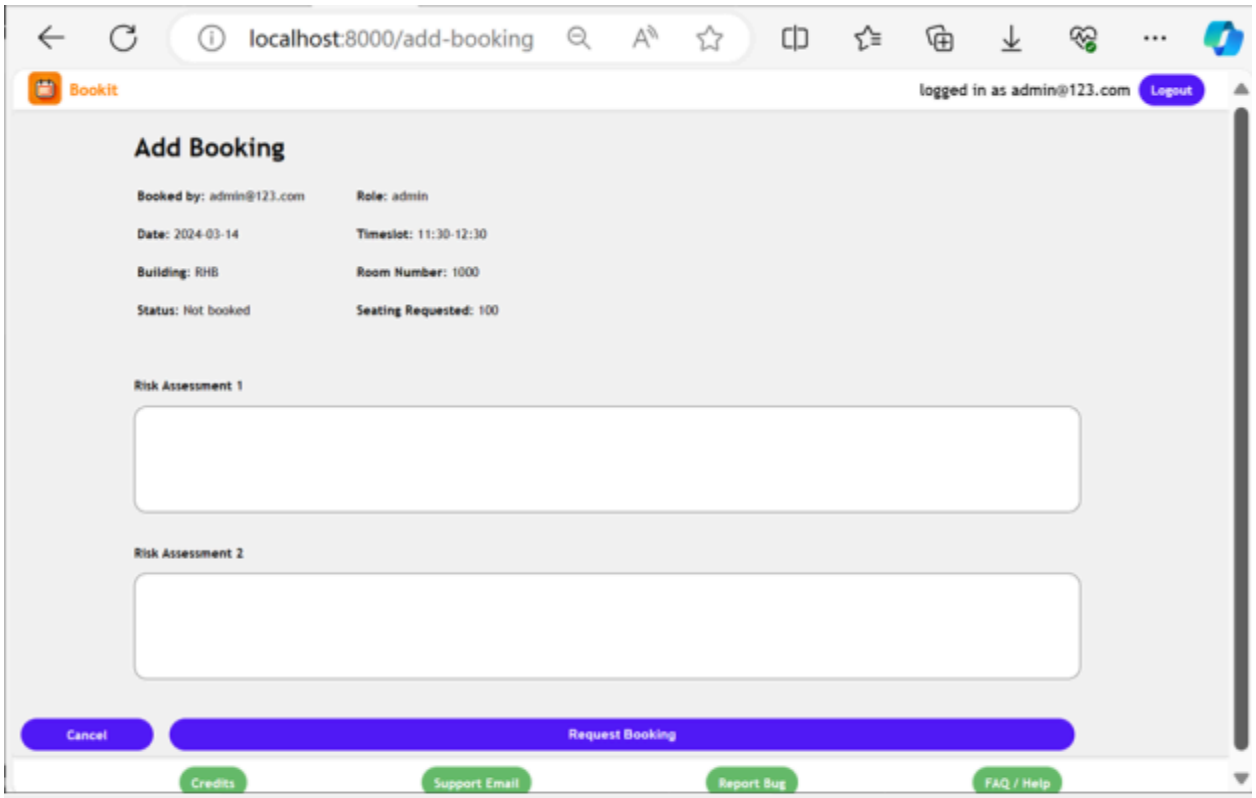


Figure 97: User Test V0 add booking

User testing feedback

- Buttons at the bottom of the page aren't formatted correctly. Logged as backlog issue 19
- The seats that I request are not stored in the booking. It looks like the seats that are shown here are the maximum seats in the room, not the seats I requested. This is important so the coordinator can review the booking. Logged as backlog issue 9.
- The risk assessment doesn't have any indicator to say how many characters are left to type. Logged as backlog issue 1
- add-booking. requirement R4 6. Template risk assessments for society leaders. common options should be given as choices and drop downs/tick boxes. Logged as issue 35

Booking List

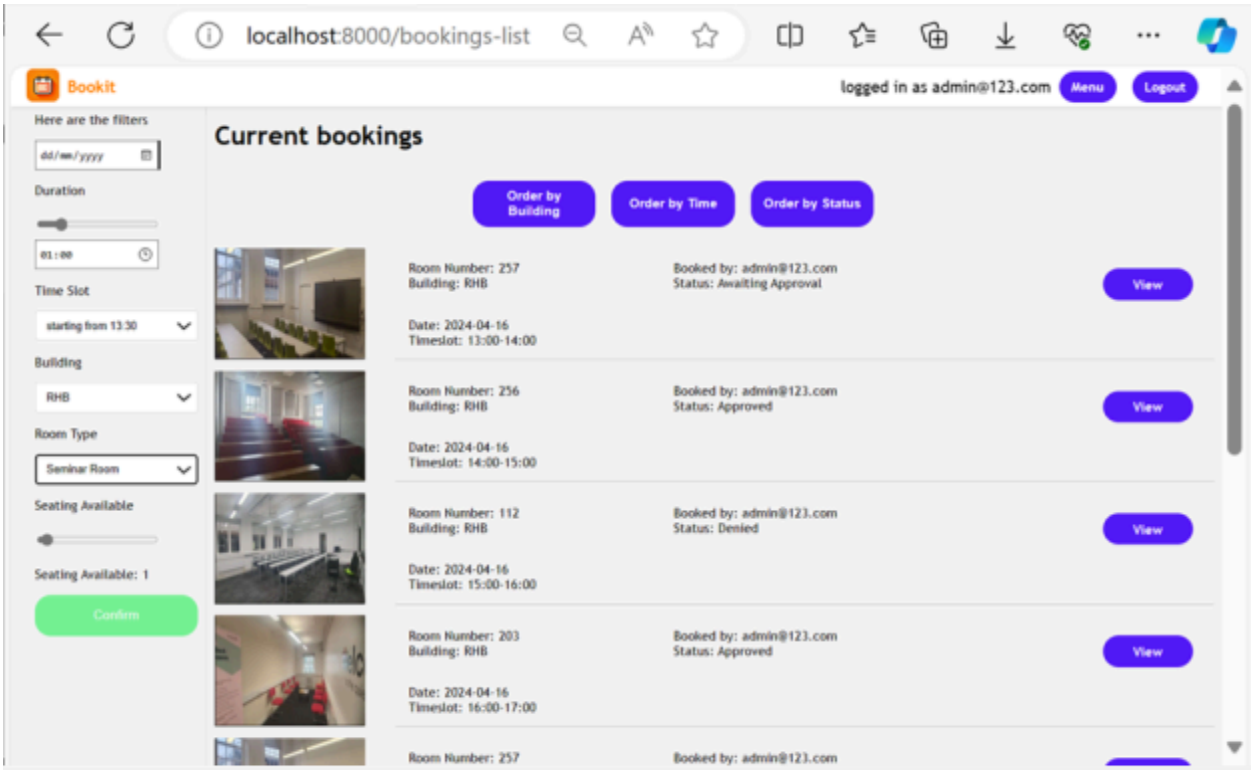


Figure 98: User Test V0 booking list

User testing feedback

- Add to the list item text to show that the risk assessment has been approved, rejected or not reviewed. Also an edit button would be good (it should only appear if the risk assessment hasn't yet been approved or rejected. Logged as backlog item 11.
- There should be a cancel booking button. Logged in backlog as issue 33.

View Booking

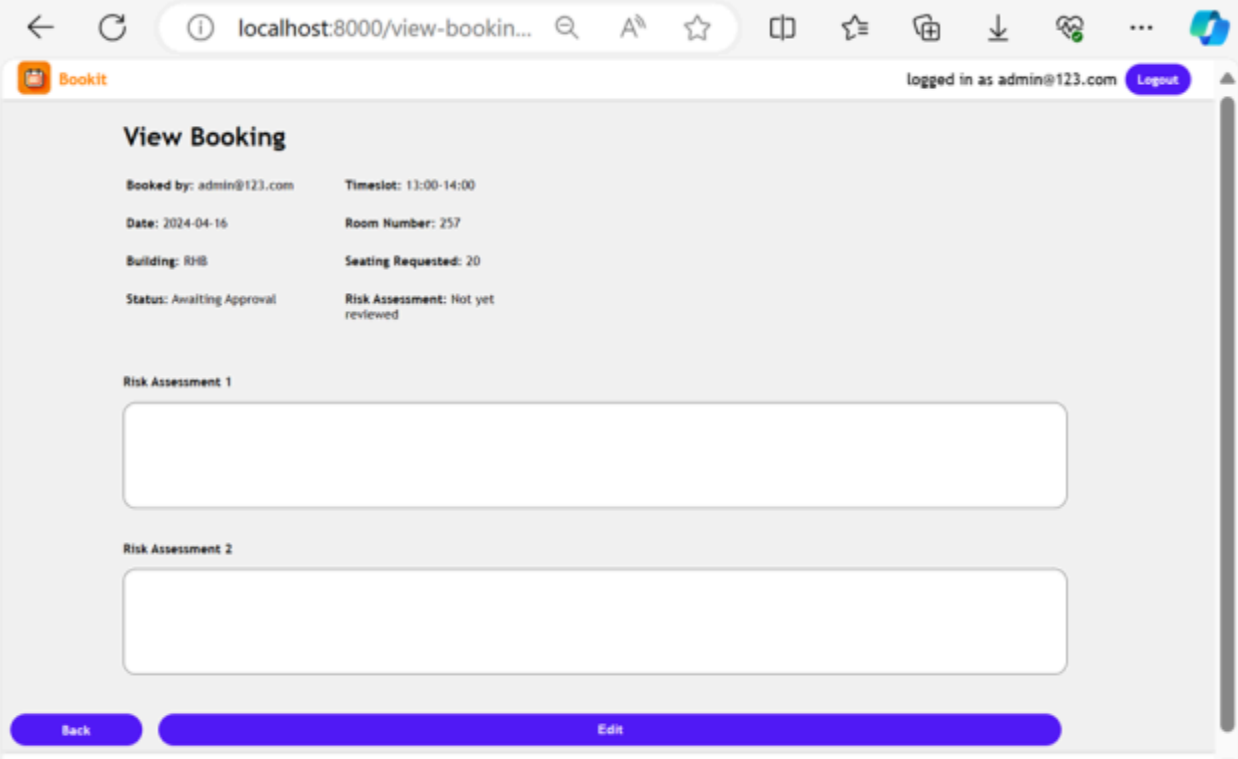


Figure 99: User Test V0 view booking

User testing feedback

- Buttons at the bottom aren't formatted correctly. Logged as backlog item 19.

Requests List

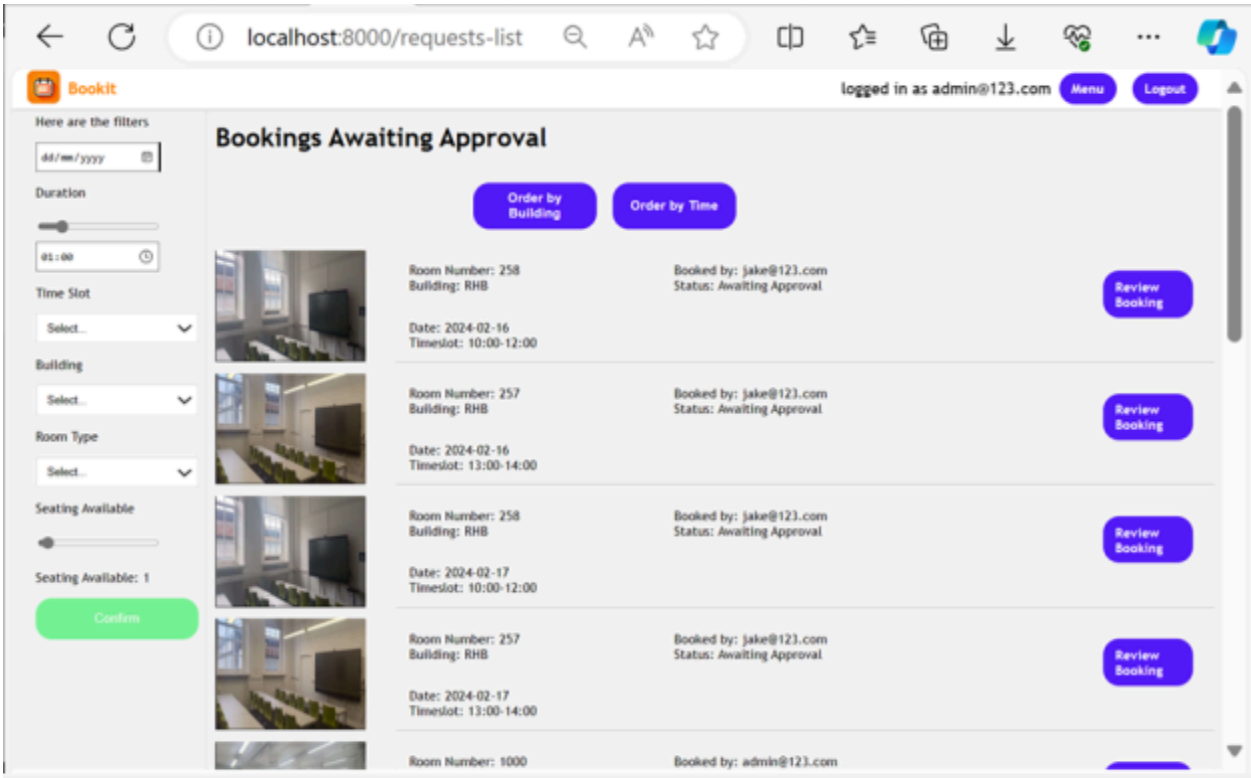


Figure 100: User Test V0 request list

User testing feedback

- Review booking button doesn't look centred. Logged as backlog item 20
- The status of the risk assessment is not shown in the item of data in the list. Add this so the coordinator can quickly see which have already been risk assessed. Logged as backlog item 21.
- Can there be a new order by button which orders by those bookings that have had a risk assessment completed and then the oldest bookings first. The coordinators would use this. Logged as backlog item 22
- Can there be a new order by button which orders by those bookings that have not been reviewed for risk assessment, with the oldest bookings first. This would be useful for users reviewing risk assessments. Logged as backlog issue 23.
- Users can edit bookings that have been made, so they can be re-assessed by a coordinator. Logged as backlog issue 34.

Review Booking

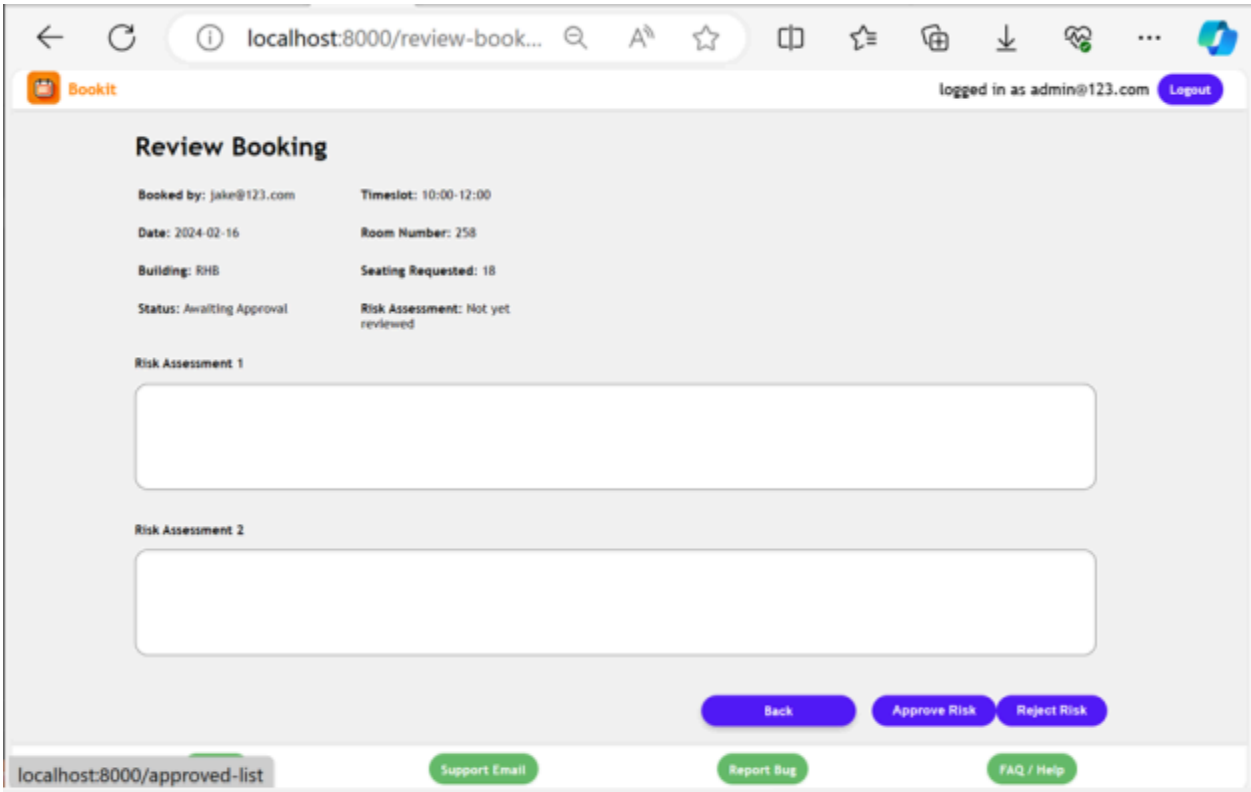


Figure 101: User Test V0 review booking

User testing feedback.

- This page changes when the risk assessment has been completed to have approve or reject booking buttons. I think it's meant to do that. This issue is expected functionality. Not added to the backlog.

Approved List

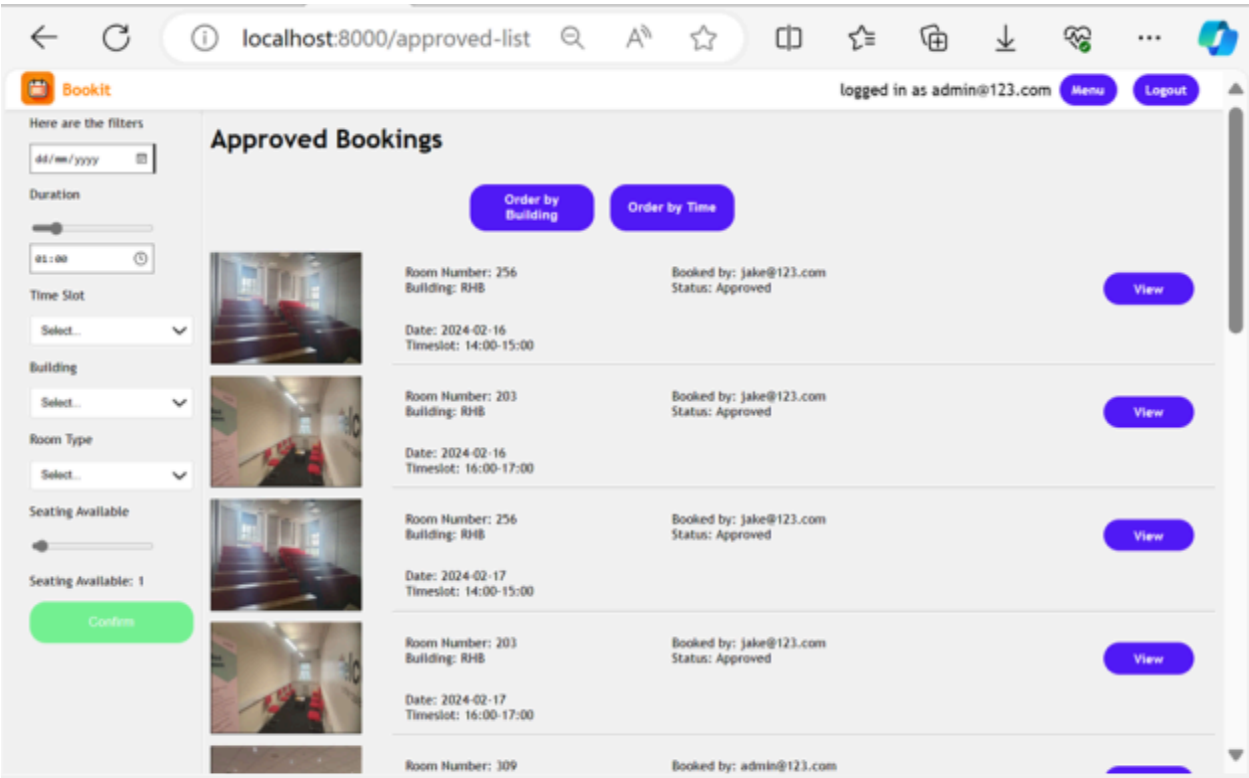


Figure 102: User Test V0 approved list

User testing feedback

- Not sure what order by time is. Can this button be named more clearly? Logged as backlog request 24.
- Can there be an “order by” that orders this list by the most recently approved booking first. This would be the most useful order. Logged as backlog request 25.

View booking

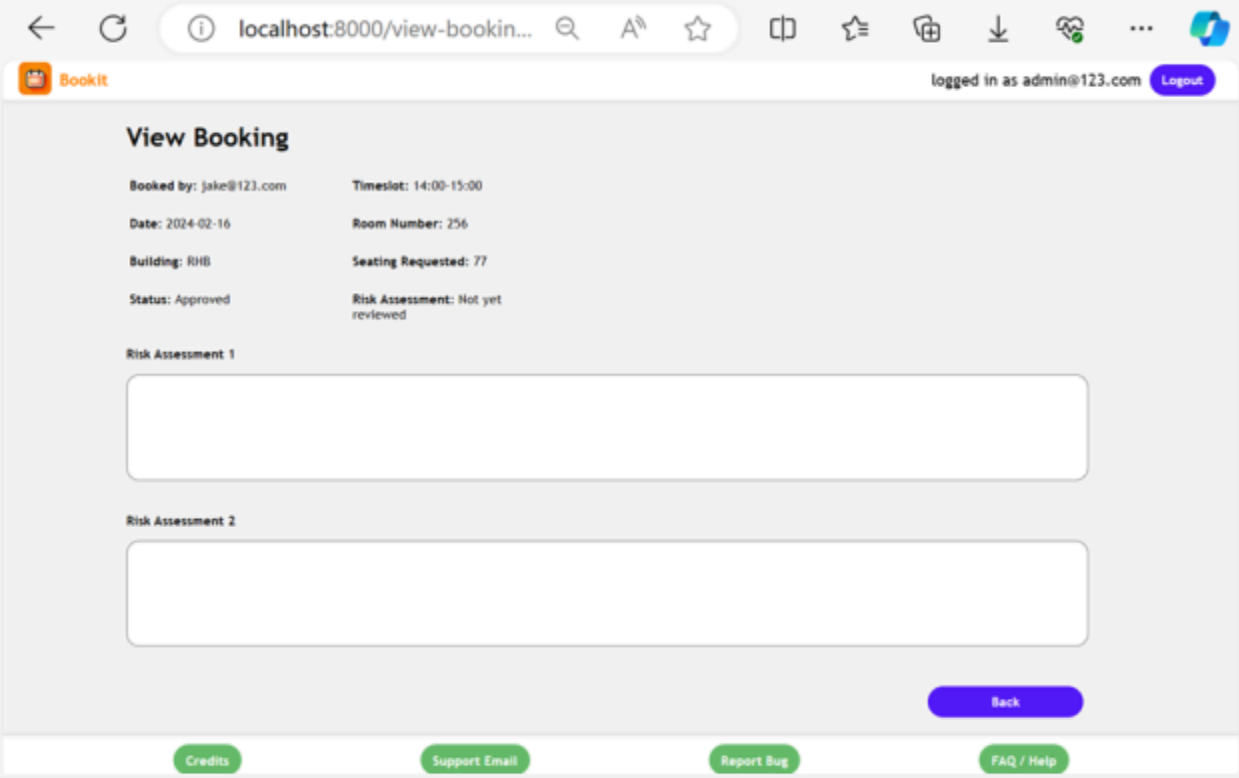


Figure 103: User Test V0 view booking

User testing feedback

- See comments on bookings list view booking, same page.

Add user

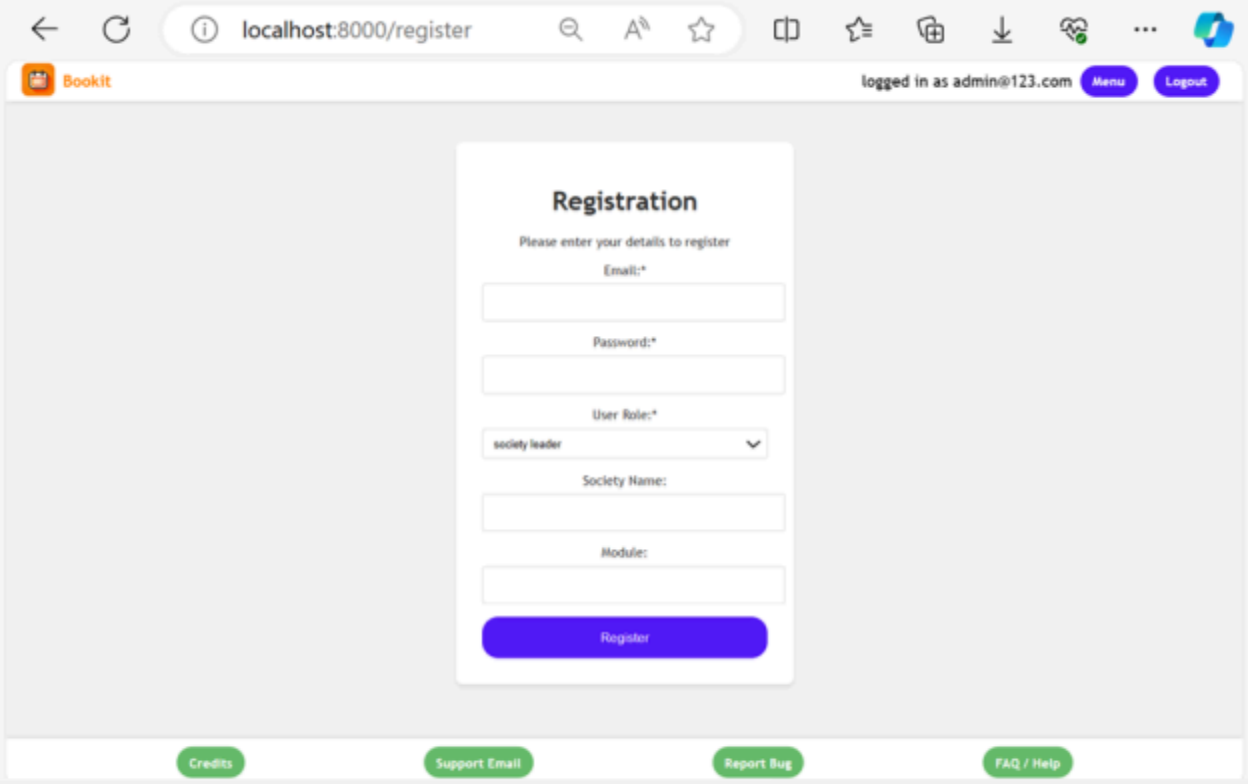


Figure 104: User Test V0 add user

User testing feedback

- The password should have some complexity checks on it, at the moment any password can be entered. Logged as backlog item 26.

Add Room

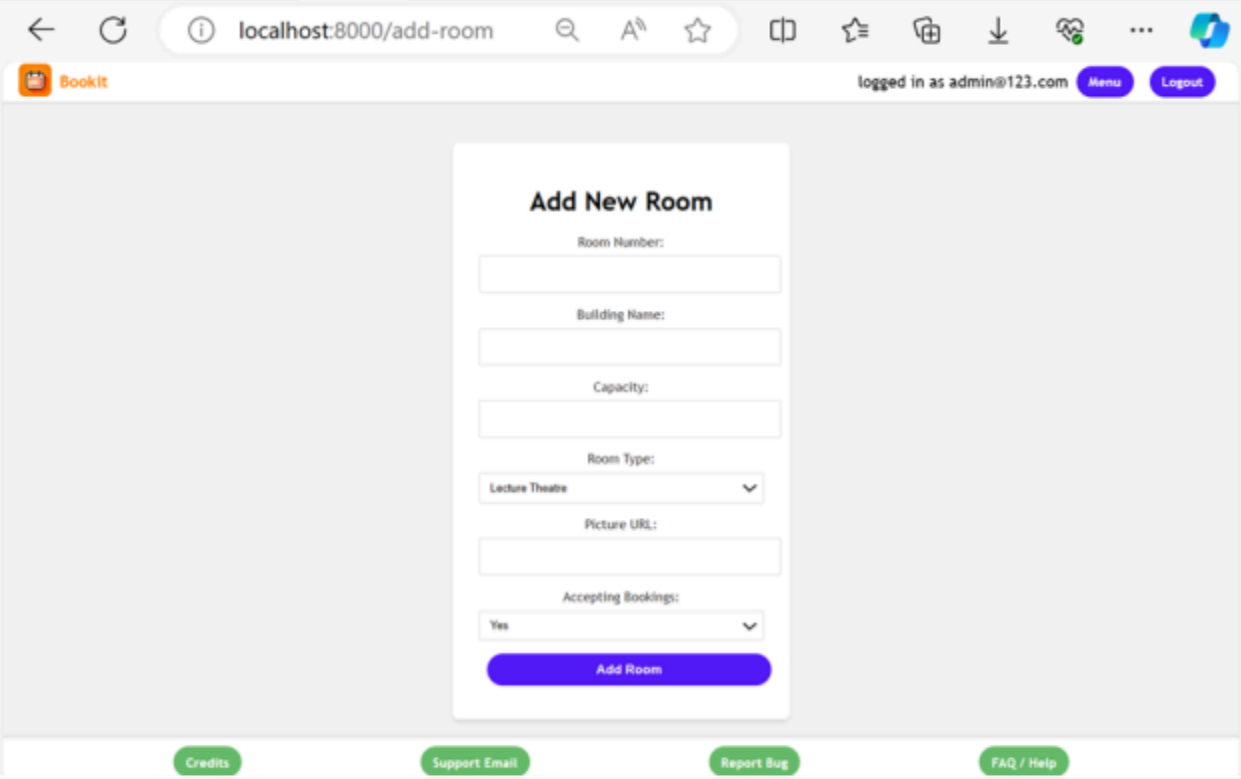


Figure 105: User Test V0 add room

User testing feedback

- How do I get a URL for a picture of a room? Is there a way of uploading a photo from this page? Logged as backlog item 8.

Edit room list

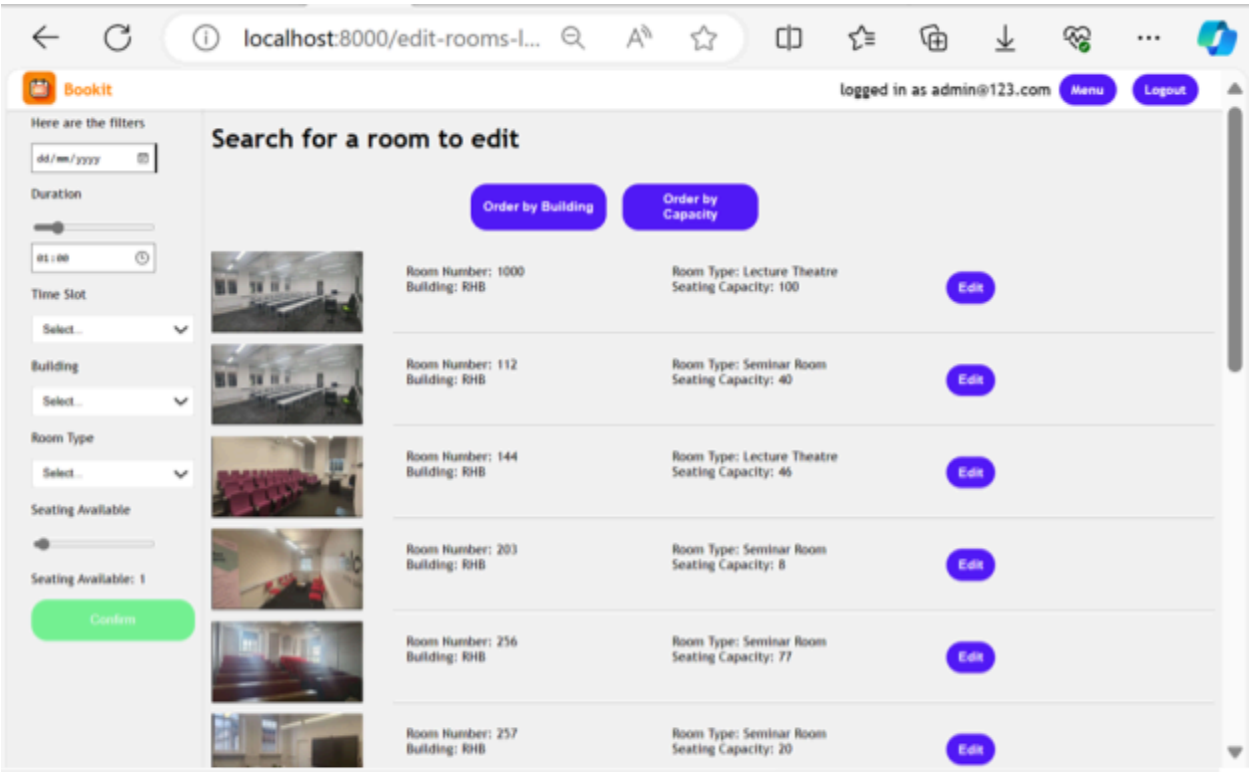


Figure 106: User Test V0 edit room list

User testing feedback

- This is the same as the rooms-list and has the same comments. Not added to the backlog.
- There should be a delete room button if you want to remove a room from the list. Logged in the backlog as issue 32.

Edit Room

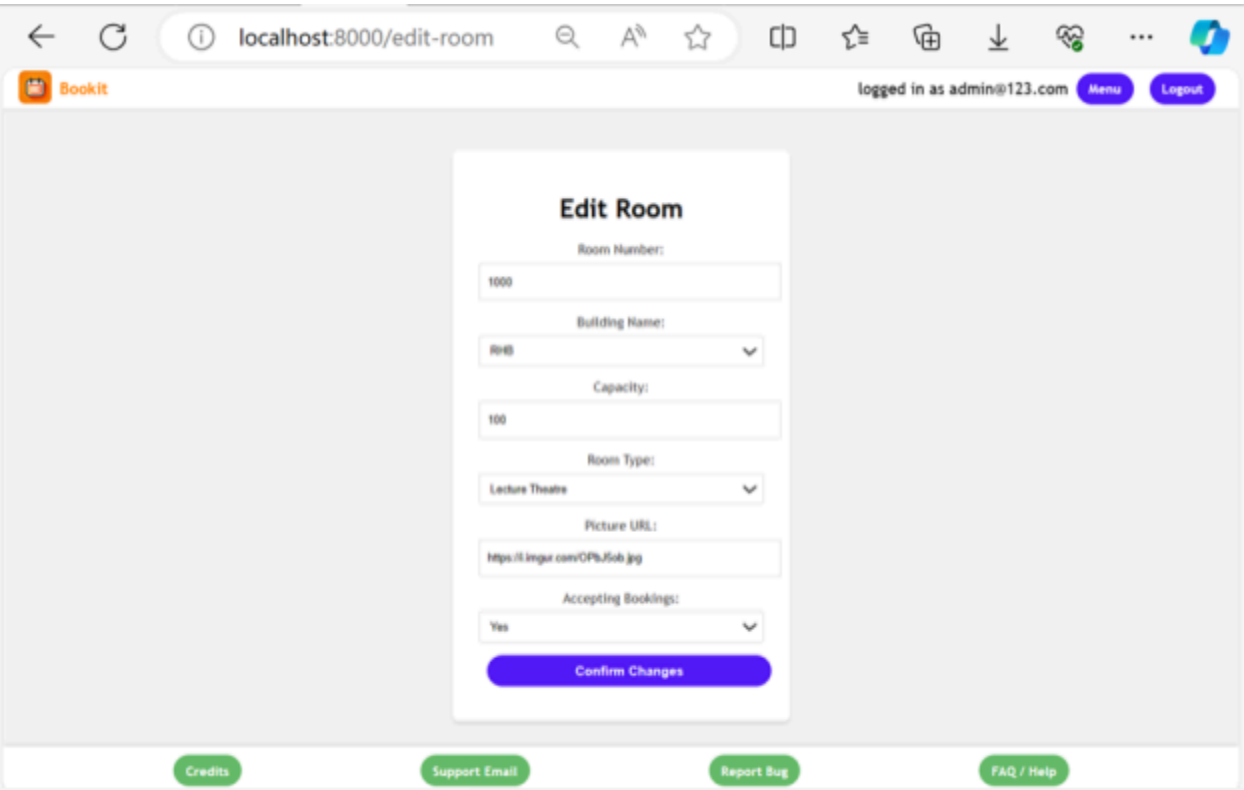


Figure 106: User Test V0 edit room

User testing feedback

- Same comment about uploading photos that was raised for add-room. Backlog item 8 updated to include edit room.

Credits Page

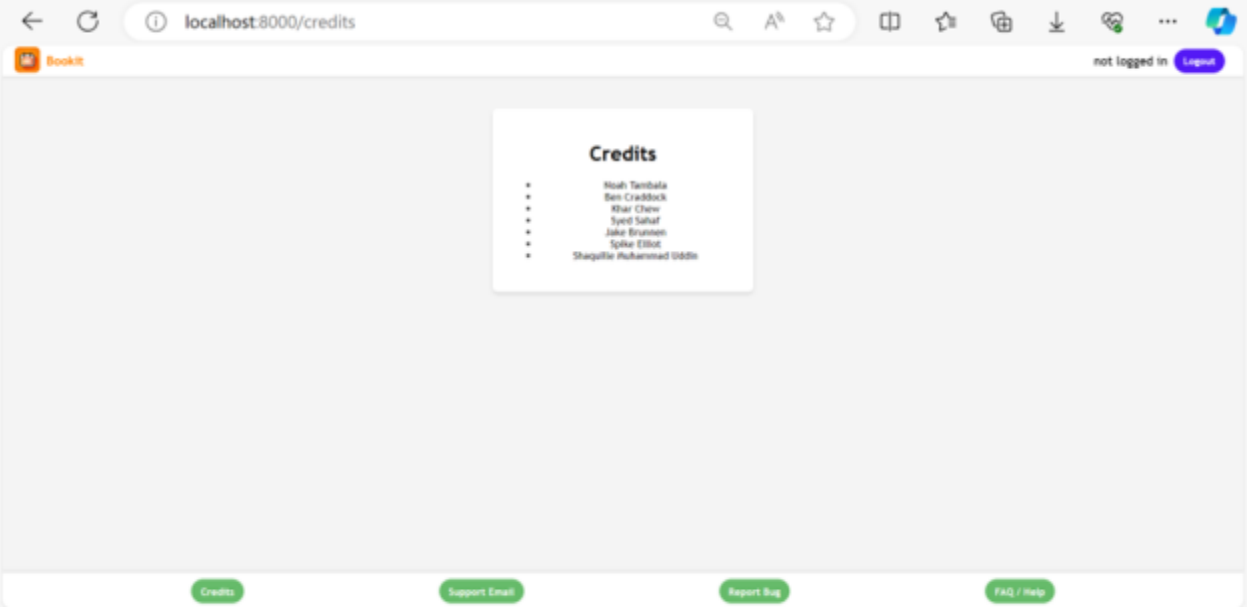


Figure 107: User Test V0 credits page

User testing feedback

- Remove the list points from this with css, so the names are centred. Logged as backlog item 28.
- Report bug link doesn't do anything. Logged as backlog item 29.

FAQ / Help

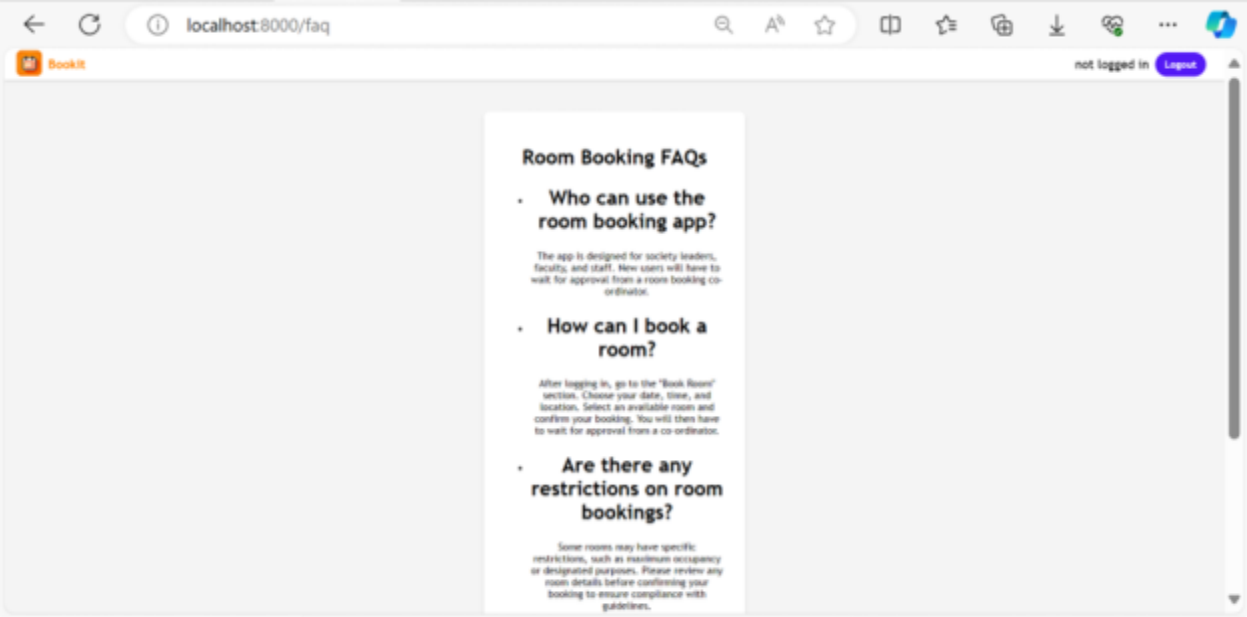


Figure 108: User Test V0 faq/help page

User testing feedback

- FAQ/Help is useful but needs styling. Logged as backlog item 30.

Version 1 user testing and feedback

Login

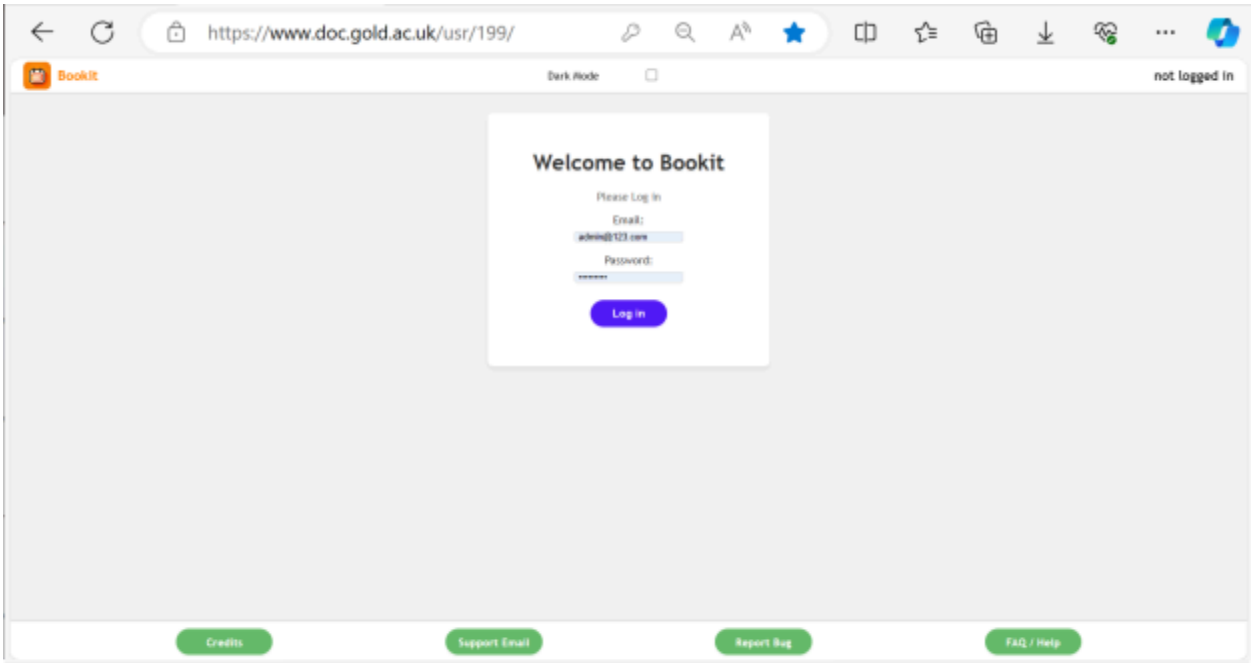


Figure 109: User Test V1 login page

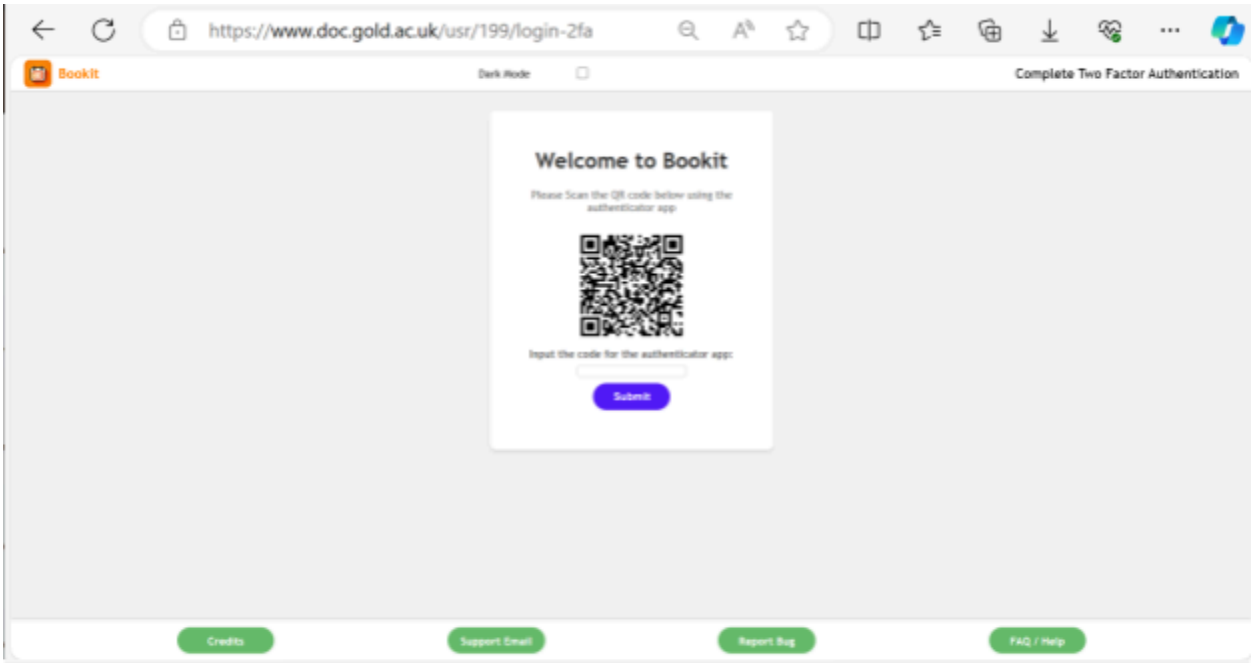


Figure 110: User Test V1 QR code

User testing feedback

- One user commented positively on the dark mode option.
- One user commented that there were no instructions about google authenticator and how to use that which is required for the two factor

Login Success

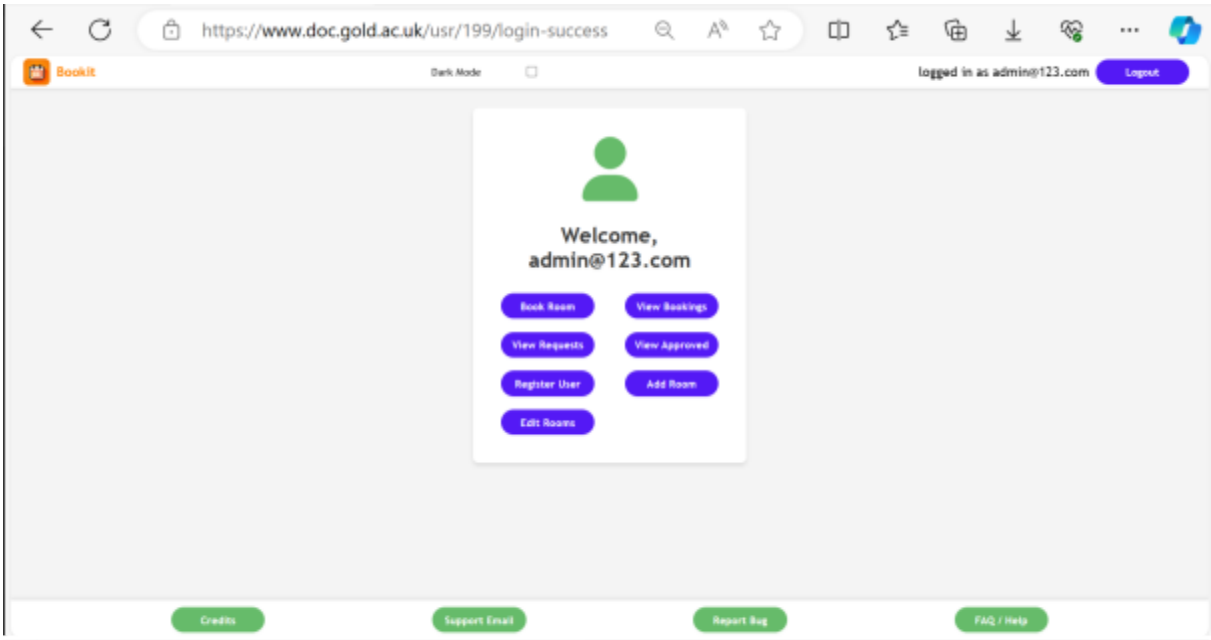


Figure 111: User Test V1 login success for admin

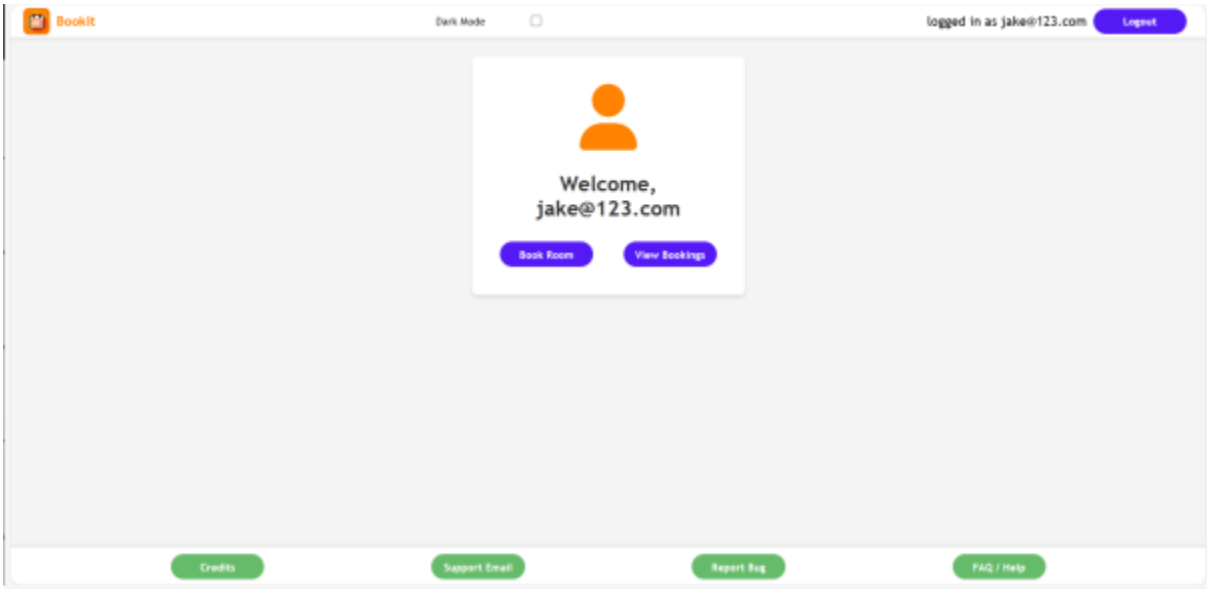


Figure 112: User Test V1 login success for user

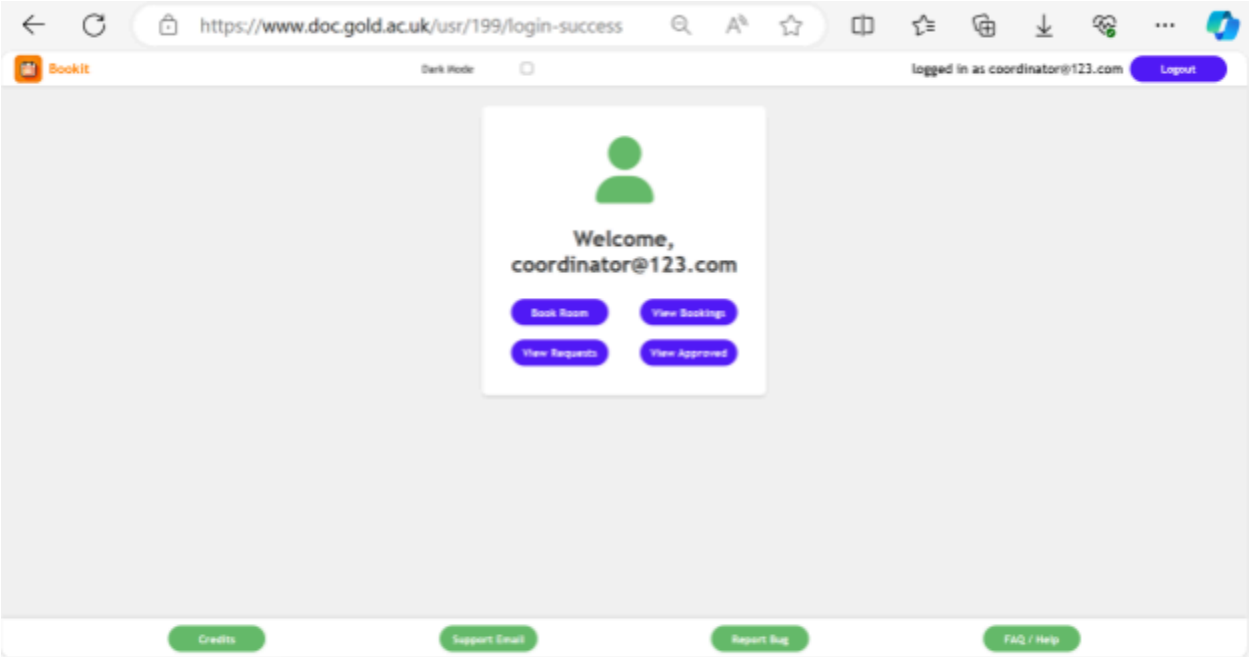


Figure 113: User Test V1 login success for coordinator

User testing feedback:

- One user commented that it would be nice to have profile pictures that could be used throughout the app.

Rooms List

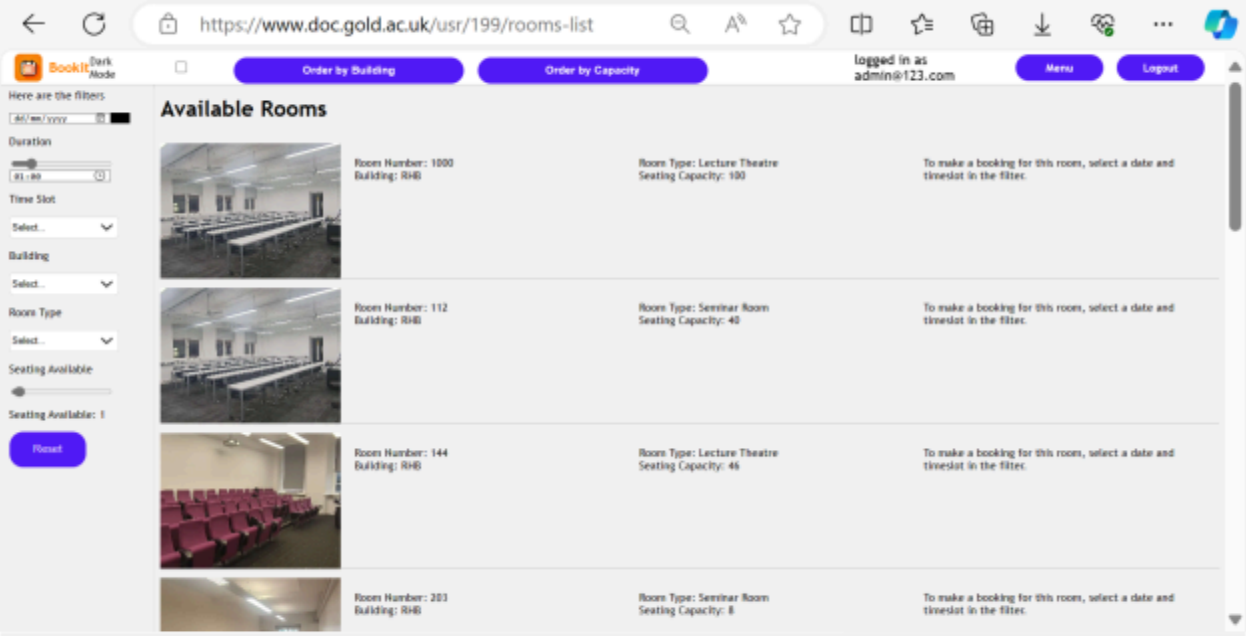


Figure 114: User Test V1 room list

User testing Feedback

- I like the filter and how the data is shown immediately if any of the filters are changed.
- Having the order buttons at the top gives more space for the list, but the top bar is crowded at different screen sizes.

Rooms list - with booking about to be requested

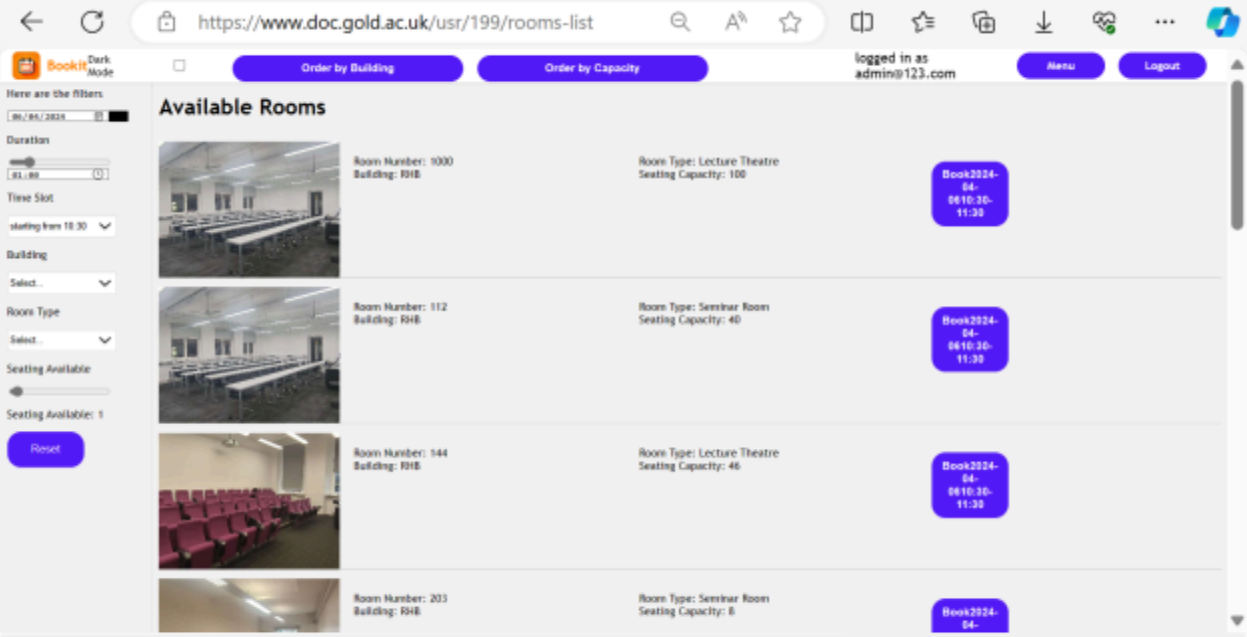


Figure 115: User Test V1 requested room list

User testing feedback

- Selecting a booking date in the past is not possible now, which is good.

Add Booking

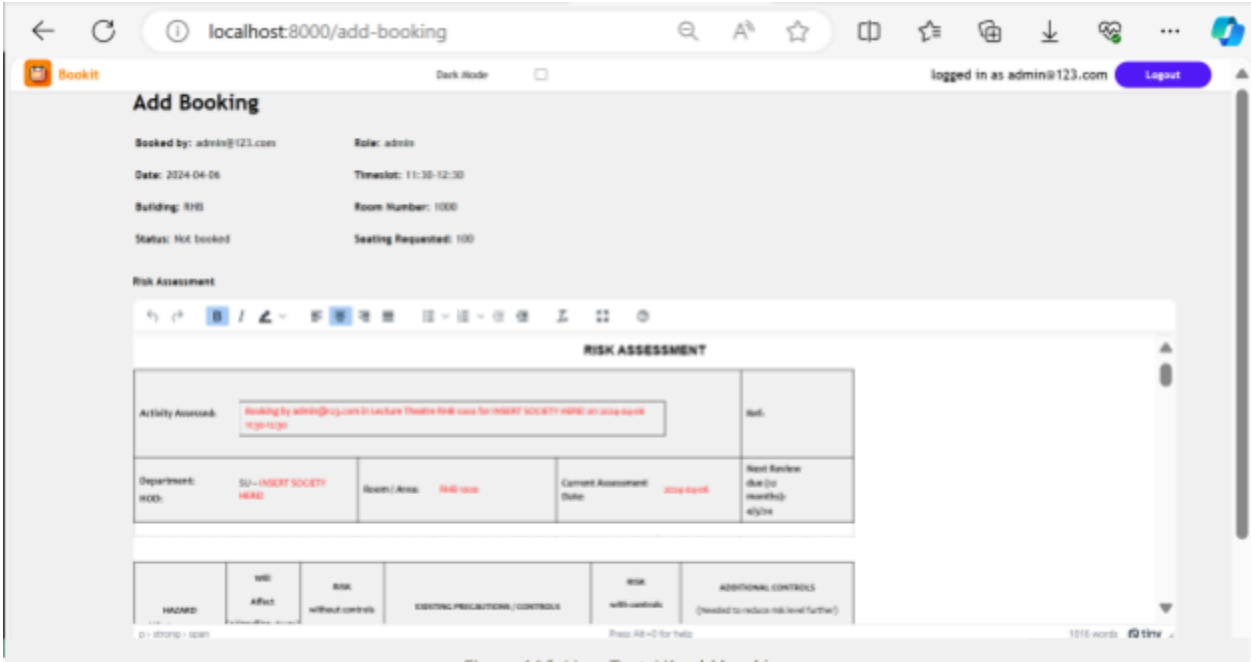


Figure 116: User Test V1 add booking

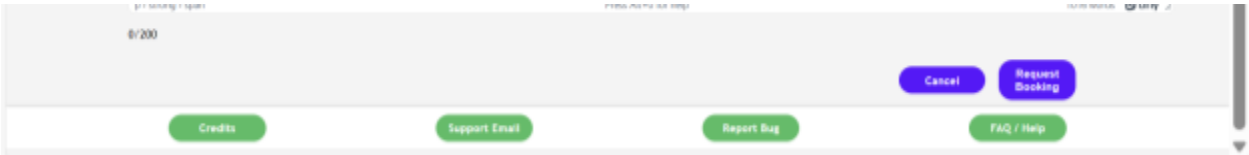


Figure 117: User Test V1 add booking buttons

User testing feedback

- The risk assessment showed as raw html on the production server. Response: This is a known issue with the production server web server which is documented in the Technical Difficulty section.
- In the development server version I tested, the risk assessment editor worked well. Some of the things I needed to enter, in red - like the room and date etc, were already filled in for me.
- I can see that this way of implementing the risk assessment is easier to change in the app if the risk assessment process ever changes. A coded html form with dropdowns and checkboxes would look better. Having used it, though, I think this is practical.

Booking List

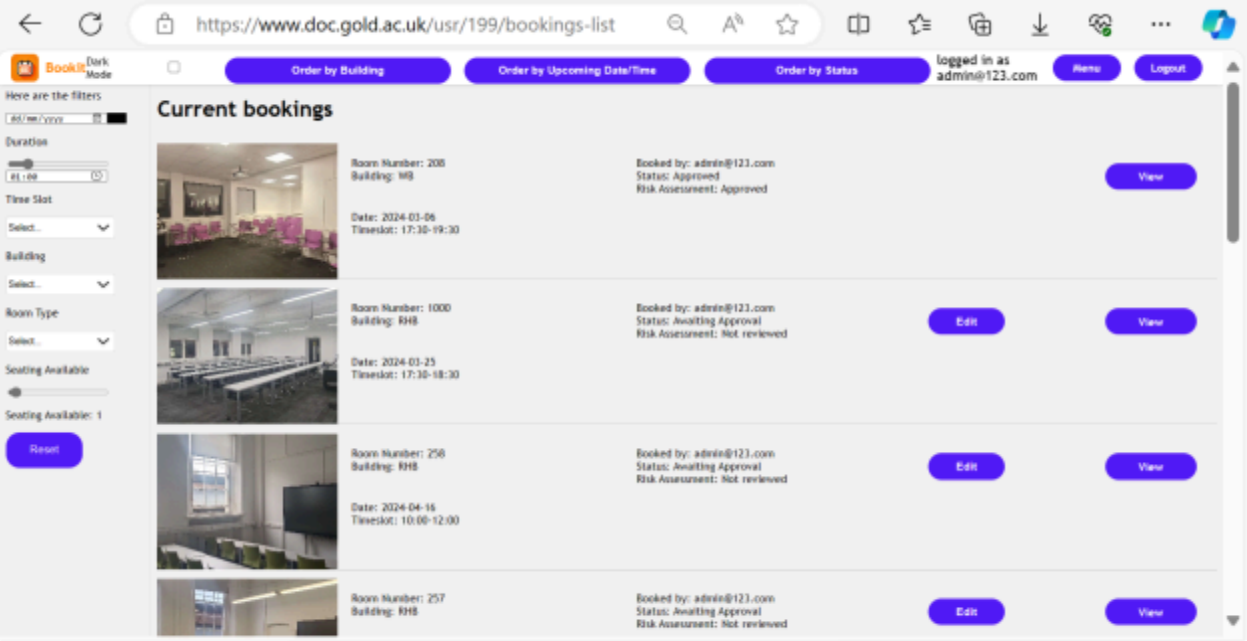


Figure 118: User Test V1 booking list

User testing feedback

- Having the order buttons at the top gives more space for the list, but the top bar is crowded at different screen sizes.
- The list only displays my bookings, which is good
- There are edit and view buttons and I only get an edit button when the booking is not yet approved.

View Booking

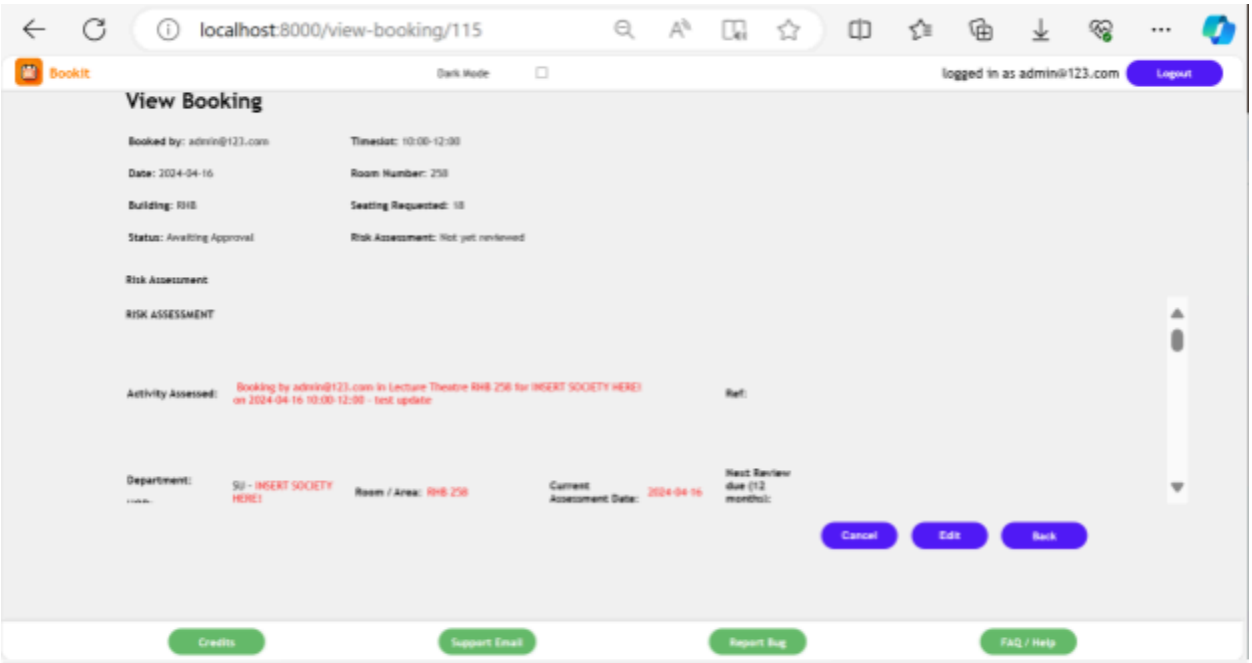


Figure 119: User Test V1 view booking

User testing feedback

- Button actions aren't clear, cancel booking, edit booking, back.

Edit Booking

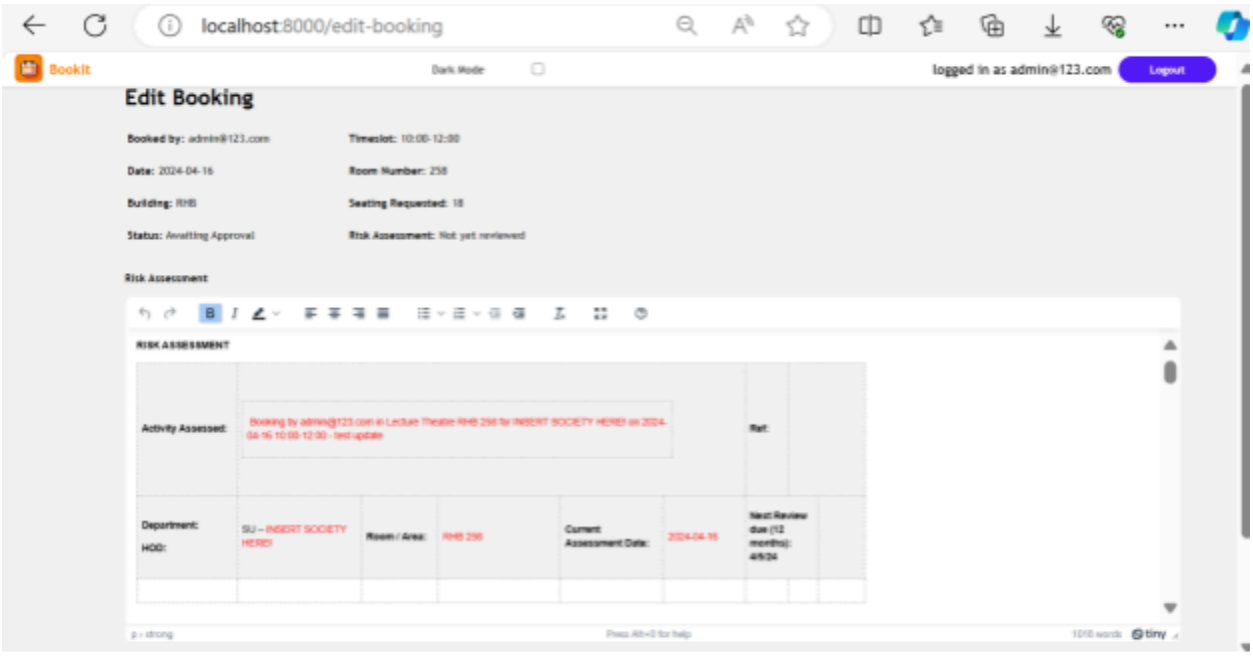


Figure 120: User Test V1 edit booking

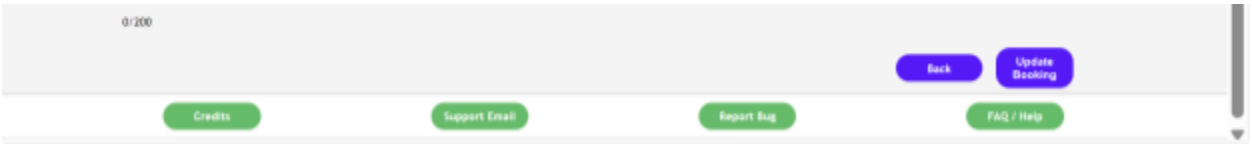


Figure 121: User Test V1 edit booking buttons

User testing feedback

- Would be nice if other parts of the booking were editable as well as the risk assessment.

Bookings awaiting Approval

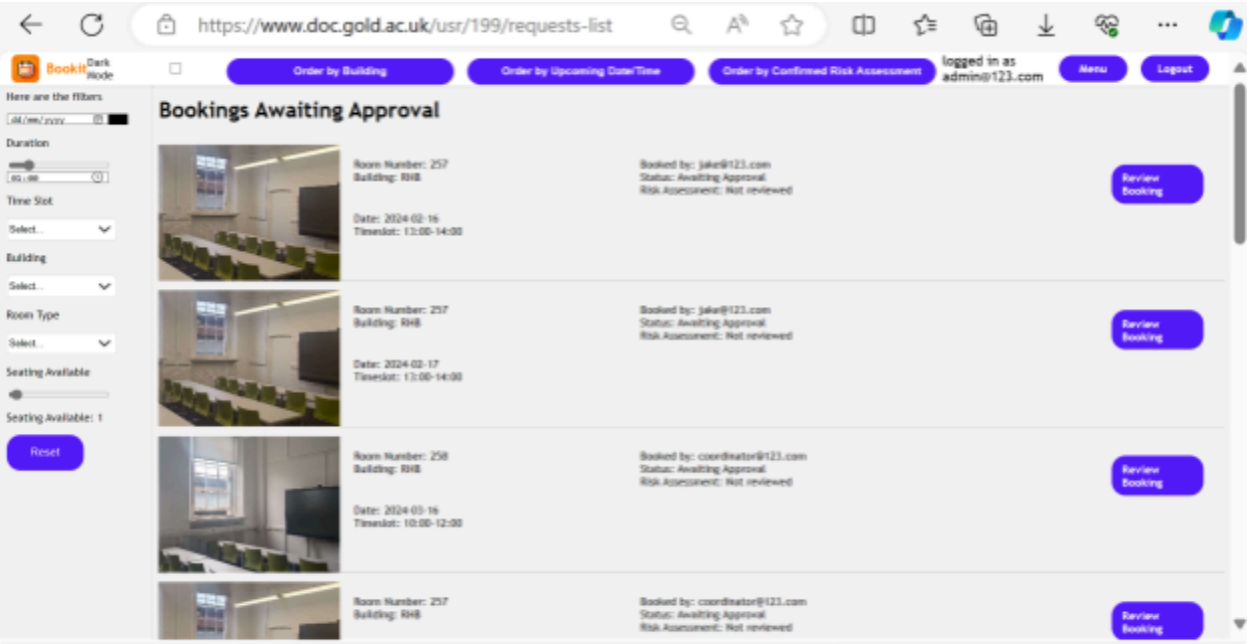


Figure 122: User Test V1 bookings awaiting approval

User testing feedback

- Having the order buttons at the top gives more space for the list, but the top bar is crowded at different screen sizes.

Review Booking

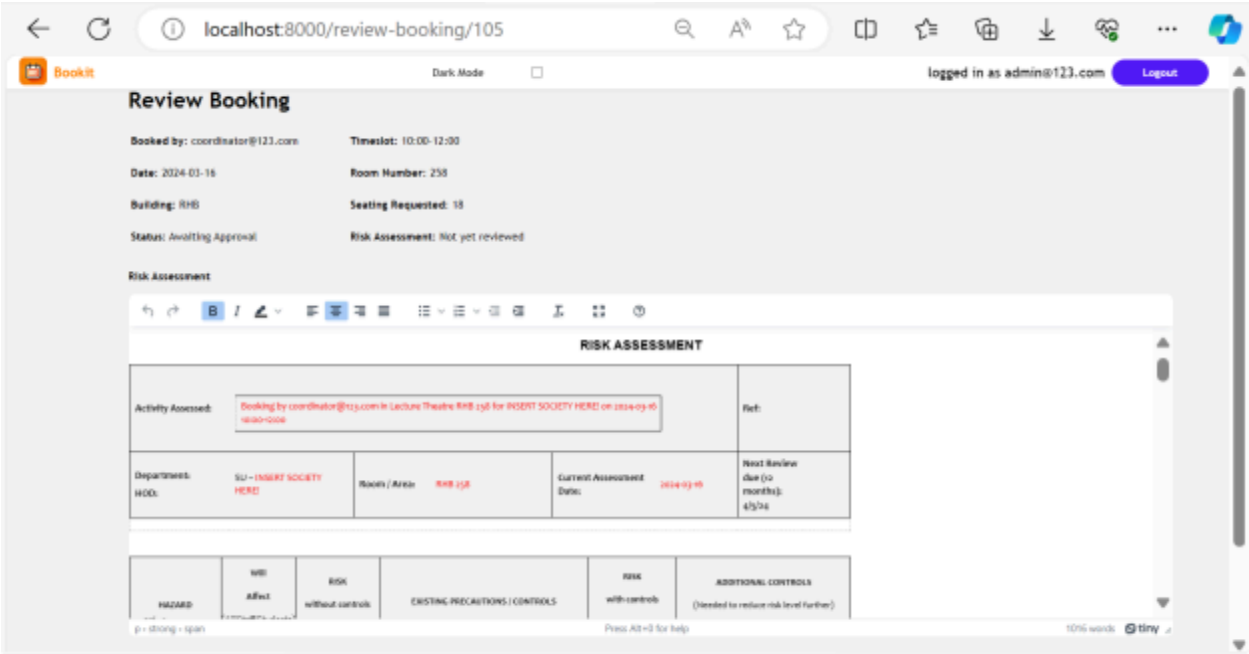


Figure 123: User Test V1 review booking

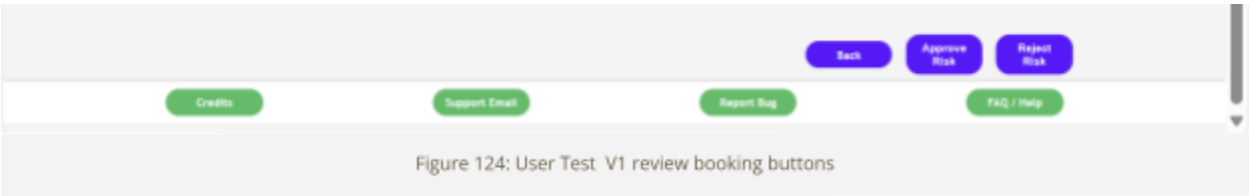


Figure 124: User Test V1 review booking buttons

User testing feedback.

- Can there be a quicker way to approve the risk assessment AND the booking at the same time.

Approved List

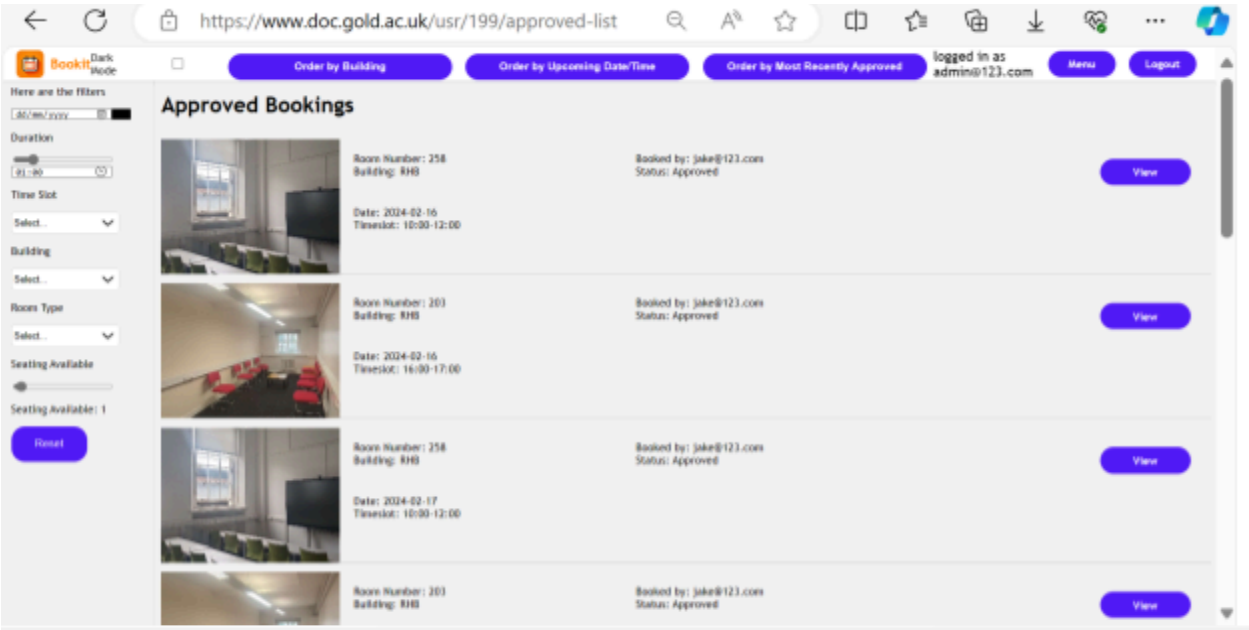


Figure 125: User Test V1 approved list

User testing feedback

- Having the order buttons at the top gives more space for the list, but the top bar is crowded at different screen sizes.
- Over time this page will fill up with a lot of older bookings, can there be a way of filtering out the bookings that are in the past?

Add user

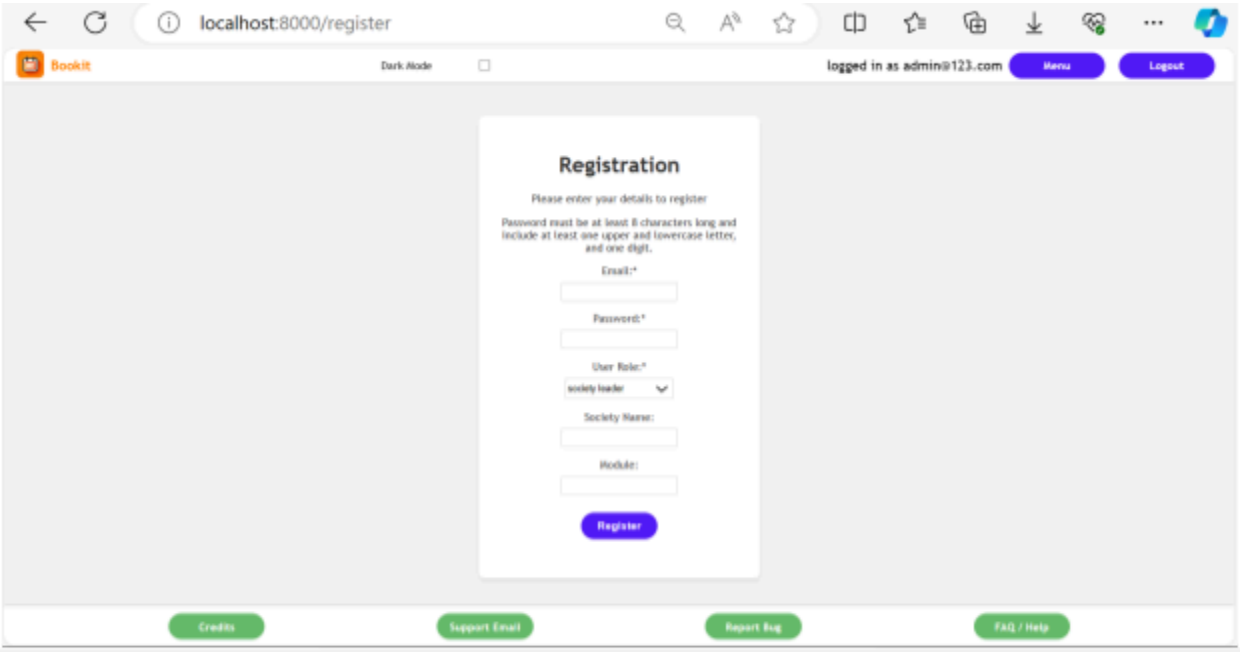


Figure 126: User Test V1 add user

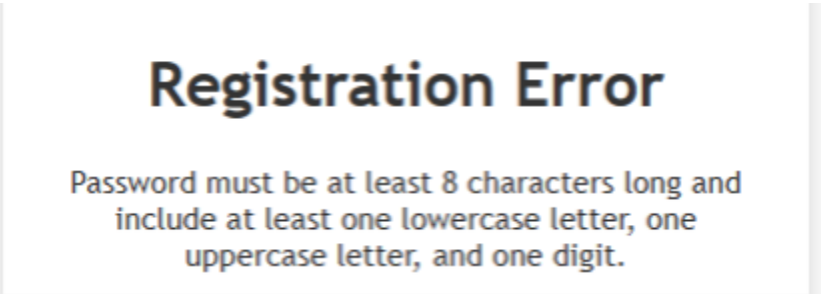


Figure 127: User Test V1 add user error

User testing feedback

- It's good to see that the password now has complexity requirements.

Add Room

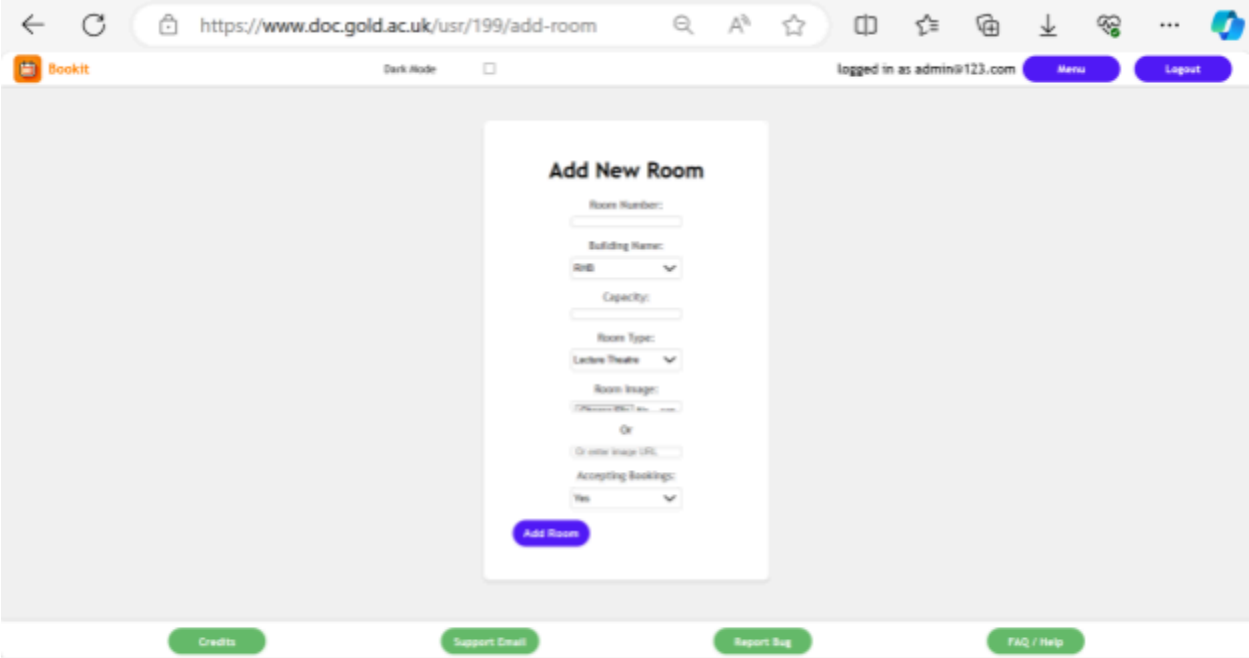


Figure 128: User Test V1 add room

User testing feedback

- Button alignment not consistent with other pages

Edit room list

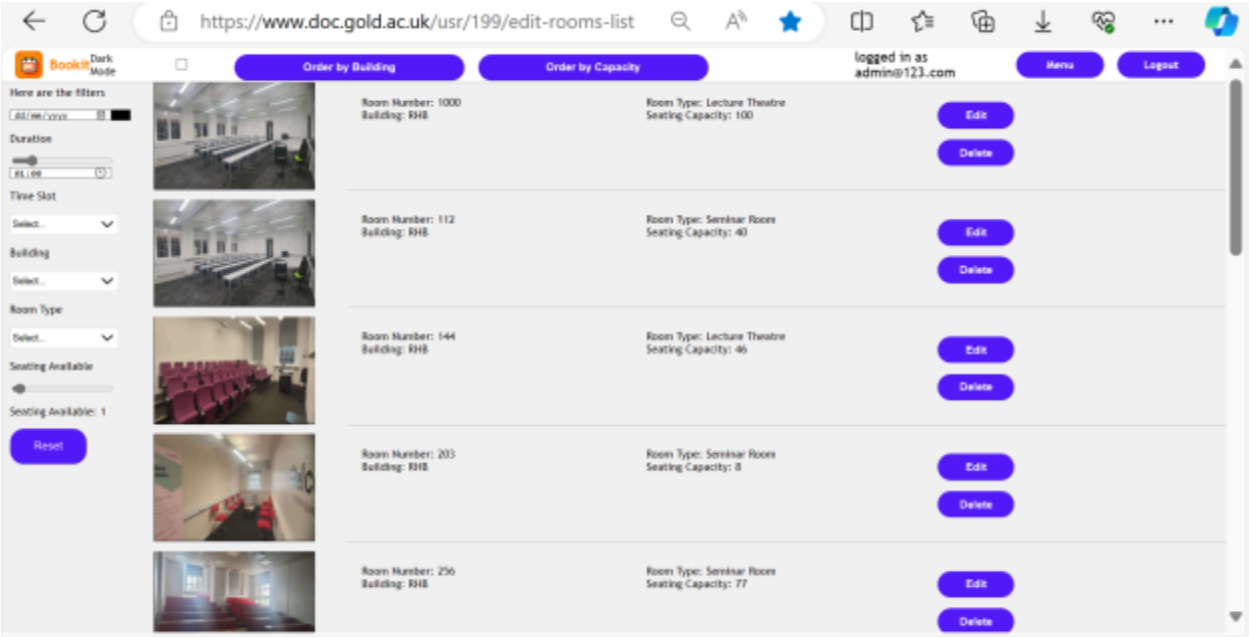


Figure 129: User Test V1 edit room list

User testing feedback

- Having the order buttons at the top gives more space for the list, but the top bar is crowded at different screen sizes.
- Not all the filters make sense on this page, they all work but they don't all apply.

Edit Room

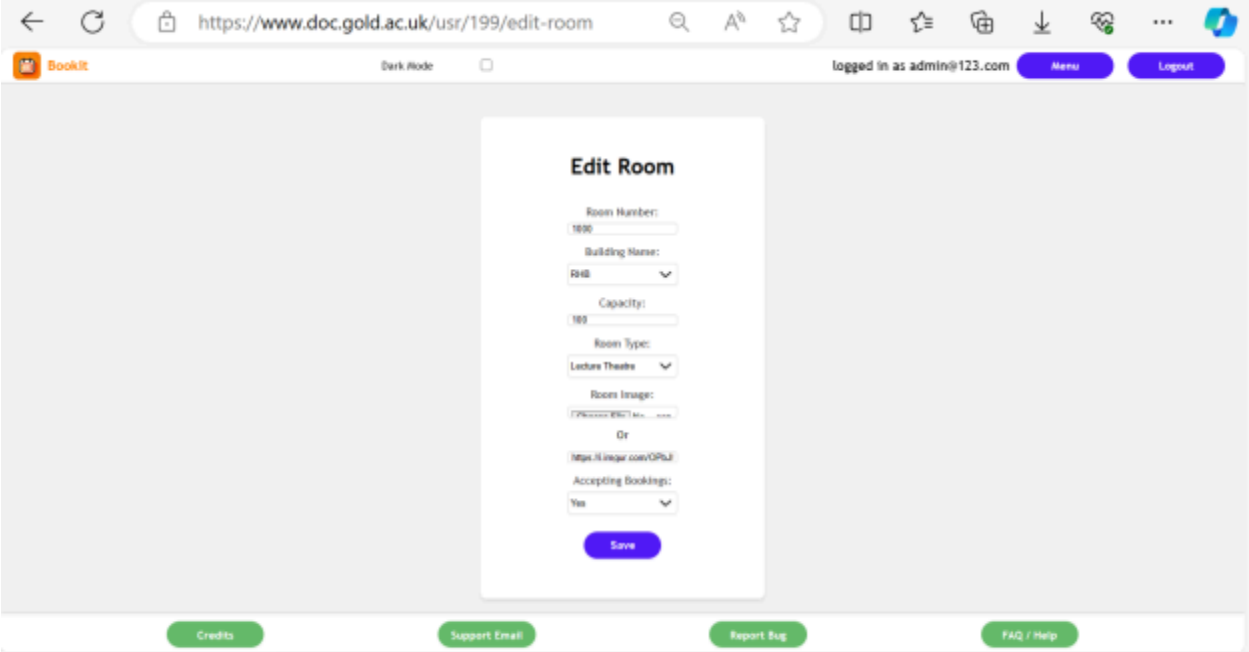


Figure 130: User Test V1 edit room

User testing feedback

- No comments.

Credits Page

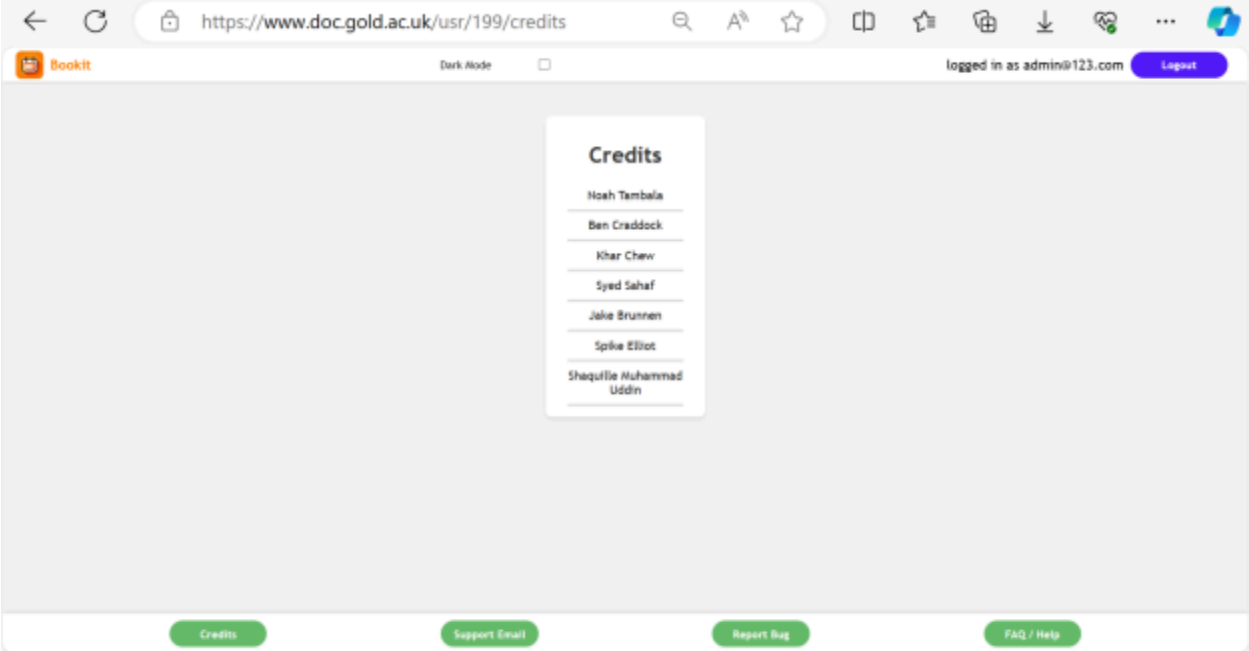


Figure 131: User Test V1 credits page

User testing feedback

- No feedback

FAQ / Help

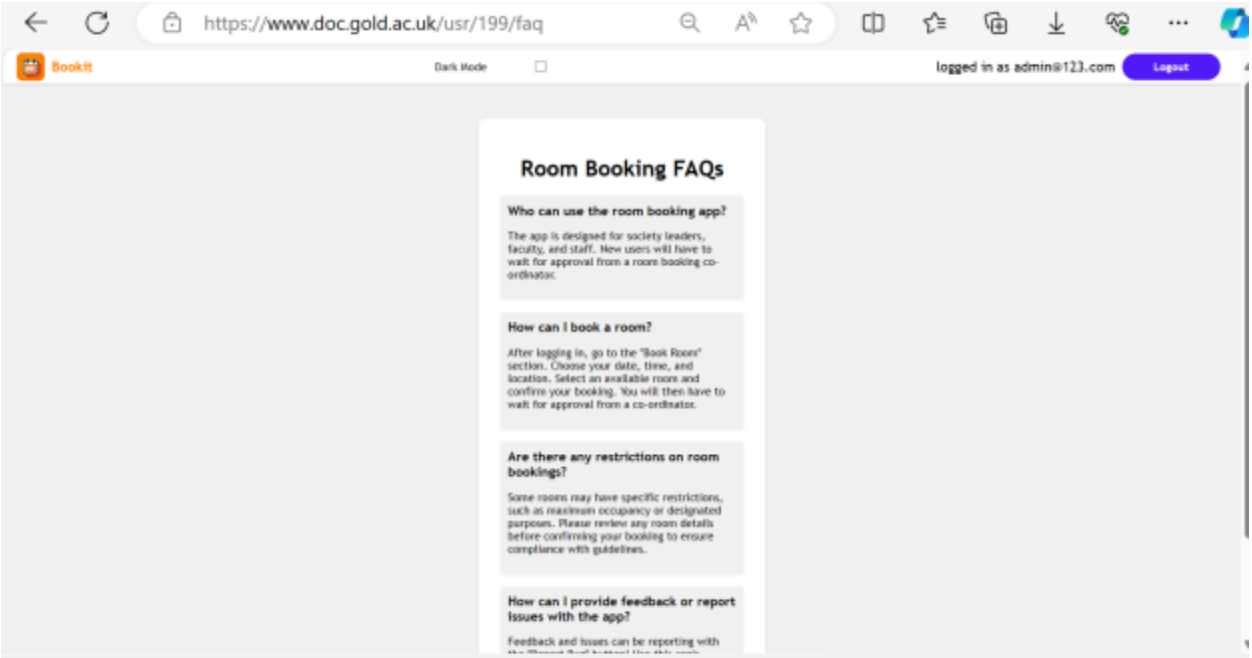


Figure 132: User Test V1 FAQ/help page

User testing feedback

- Provides basic help

Security Testing Plan

We decided to carry out security testing at key stages during the development cycle. This approach meant that we didn't find issues at the end of the development cycle that would require re-working the code significantly.

Our research identified two tools that we could use to undertake automated and manual security testing. The two tools were OWASP Zap and Qualys Web Application Scanner.

The tool we selected to perform security testing was OWASP Zap. This tool intercepts GETs POSTS and the HTML of the pages as you are using the product and makes recommendations based on the top 10 OWASP vulnerabilities. There are also add-on modules for SQL Injection attacks.[24]

[24] https://www.zaproxy.org/

The OWASP Top 10 is a project to increase security awareness for development teams. Its last release was in 2021, and the covered security areas are outlined below.[25]

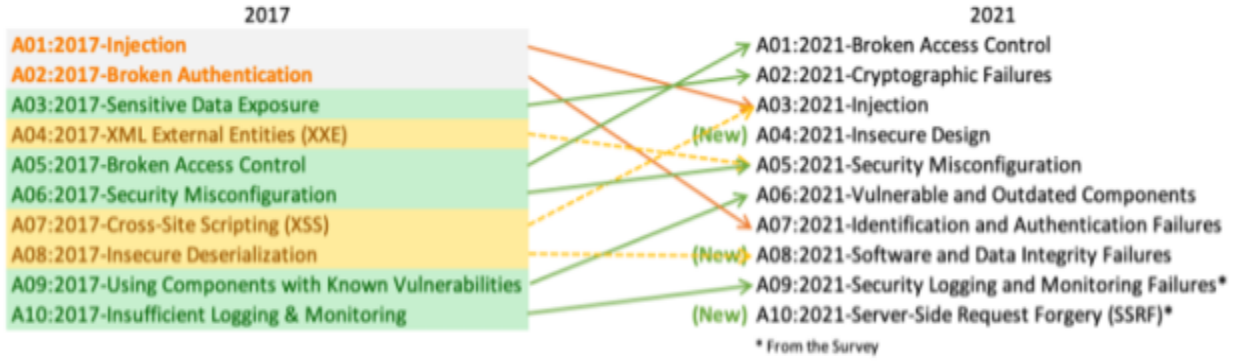


Figure 133: OWASP breakdown

[OWASP Top Ten | OWASP Foundation]

OWASP Zap was selected for testing because

- it was able to be used without automation, for weekly sprint reports. But also able to be used in an automated way at the end of a version release

- it had comprehensive tests that covered the common issues in developers' code (OWASP top 10)/

- It generated useful reports with hints and links to security resources to improve code.

This testing was completed during each sprint, and issues that were raised were allocated to a future sprint for completion.

The lists of tests that are carried out by OWASP Zap are listed on these webpages

- ZAP – Active Scan Rules (zaproxy.org)
- ZAP – Passive Scan Rules (zaproxy.org)
- ZAP – Retire.js (zaproxy.org)
- ZAP – DOM XSS Active Scan Rule (zaproxy.org)

The other tool that was considered was QUALYS Web Application Scanner.[26] This is a scanner that performs simulated attacks on the application.

[25] "OWASP Top Ten | OWASP Foundation." *Owasp.org*, 2021
[26] "Web Application Scanning | Qualys, Inc." *Www.qualys.com*, www.qualys.com/apps/web-app-scanning/.

For this product to work the application needs to be hosted on an internet facing server, the provider should be aware that tests are being carried out. This product is more complicated to set up than OWASP Zap and, like OWASP Zap, it provides reports of issues and lists actions that can be taken to resolve these issues.

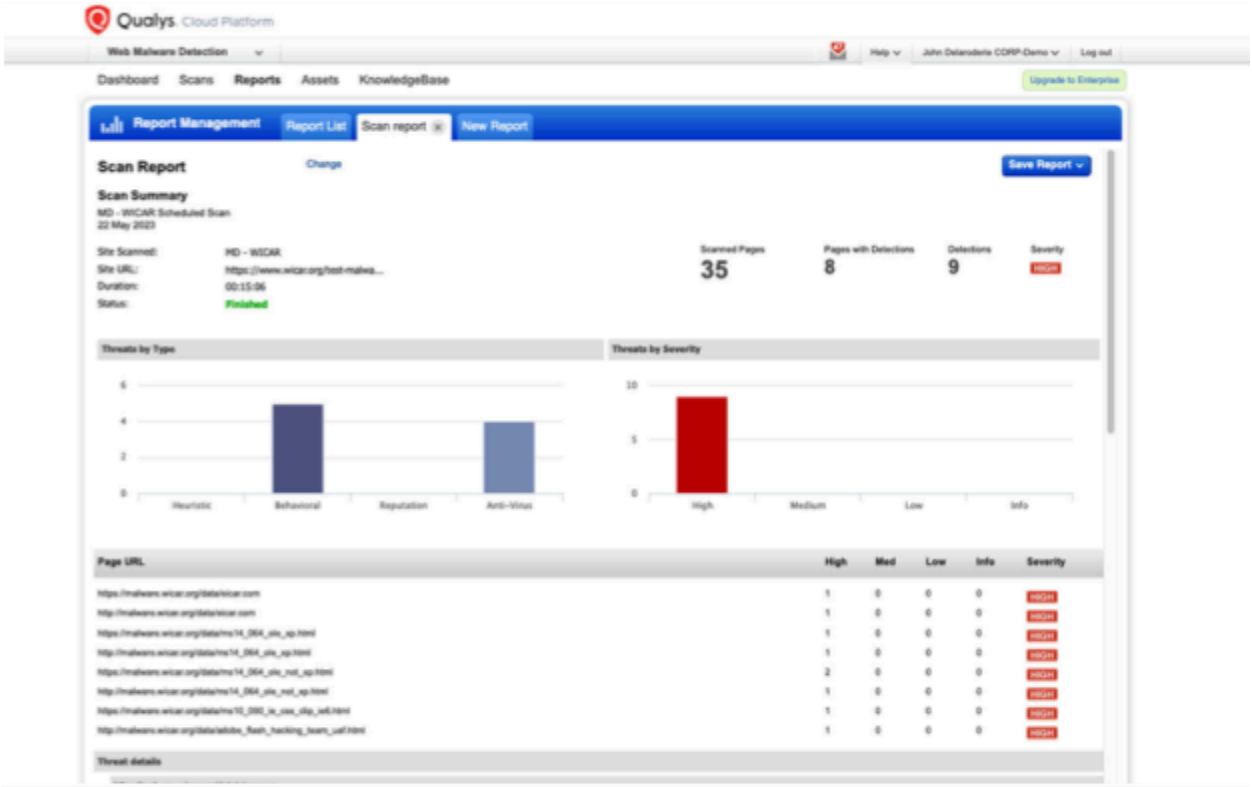


Figure 134 - Qualys scan report

[Qualys Web Application Scanning | Qualys]

This product:

- carries out a very comprehensive set of tests,
- simulates attacker behaviour,
- is a well-known and recognised product, which is updated regularly to include new security issues that are found.

We didn't choose Qualys because we felt that learning two products in the timescale was not efficient. However Qualys has a scheduling tool which can email results automatically when the scan is done, once the project is live, we plan on using Qualys Web Application Scanning tool on a monthly basis. Even though there will be fewer code changes after going live, running the tool monthly will identify new threats to the existing code.

Week 4 Security testing

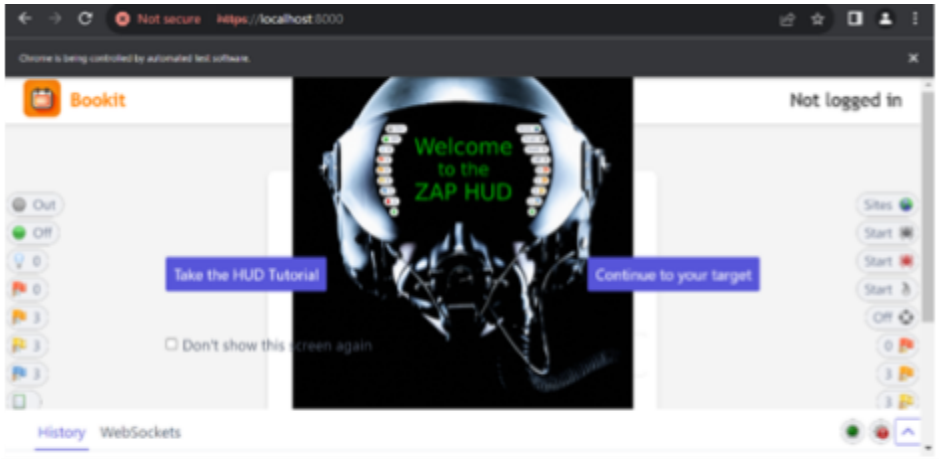


Figure 135 - ZAP HUD Bookit demonstration

This was the first week the testing was run.

The screen on the left shows the interface for OWASP Zap in manual mode.

These results are summarised here:

Summary of Alerts

Risk Level	Number of Alerts
High	0
Medium	3
Low	5
Informational	7
False Positives:	0

Figure 136 - OWASP summary

Alerts

Name	Risk Level	Number of Instances
Absence of Anti-CSRF Tokens	Medium	9
Content Security Policy (CSP) Header Not Set	Medium	19
Missing Anti-clickjacking Header	Medium	19
Cookie without SameSite Attribute	Low	2
Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)	Low	14
Strict-Transport-Security Header Not Set	Low	21
Timestamp Disclosure - Unix	Low	7
X-Content-Type-Options Header Missing	Low	20
Authentication Request Identified	Informational	3
Information Disclosure - Suspicious Comments	Informational	2
Loosely Scoped Cookie	Informational	2
Modern Web Application	Informational	10
Re-examine Cache-control Directives	Informational	17
Session Management Response Identified	Informational	3
User Controllable HTML Element Attribute (Potential XSS)	Informational	3

Figure 137 - OWASP security alert summary

Assessment

For the first test, we thought this was a good outcome. There are no high priority issues. Two of the medium issues are about configuration of node.js and not the application code. These two issues look to have standard resolutions.

The other medium item is about the implementation of Anti-CRSF tokens.[27] Research shows that this can be implemented by the generation of a unique ID when the user login session is created and then this unique ID is passed to each page. In each get or post this unique ID should be sent.[28] In app.js, when the route is run, the unique ID received is checked against the session value for this unique ID. A match indicates a regular user. No match indicates a potential attacker.

Planned Action

The medium-level fixes identified will be targeted for resolution during the week 5 sprint.

[27] "Cross Site Request Forgery (CSRF) | OWASP Foundation." *Owasp.org*, https://owasp.org/www-community/attacks/csrf

[28] Marasinghe, Lanil. "How to Secure Your NodeJS Web Application Using Synchronizer Tokens." *Medium*, 26 Oct. 2018, medium.com/@lanil.marasinghe/how-to-secure-your-nodejs-web-application-using-synchronizer-tokens-959c45200876

Week 5 – security test results

Summary of Alerts

Risk Level	Number of Alerts
High	0
Medium	3
Low	3
Informational	7
False Positives:	0

Figure 138 - OWASP alert summary

Alerts

Name	Risk Level	Number of Instances
Absence of Anti-CSRF Tokens	Medium	8
Content Security Policy (CSP) Header Not Set	Medium	15
Missing Anti-clickjacking Header	Medium	8
Strict-Transport-Security Header Not Set	Low	19
Timestamp Disclosure - Unix	Low	7
X-Content-Type-Options Header Missing	Low	9
Authentication Request Identified	Informational	1
Information Disclosure - Suspicious Comments	Informational	3
Loosely Scoped Cookie	Informational	1
Modern Web Application	Informational	7
Re-examine Cache-control Directives	Informational	16
Session Management Response Identified	Informational	3
User Controllable HTML Element Attribute (Potential XSS)	Informational	3

Figure 139 - Alert report

Four Low issues have been resolved.

An app.use middleware section was implemented in the app,js to add and change headers to resolve security issues. Added SameSite attribute to the session cookies, remove Powered by Express by using app.disable.

```
app.use(session({
   secret: 'secretforumkey',
   resave: false,
   saveUninitialized: true,
```

```
  cookie: {
    sameSite: 'strict'
  }
}));
```

Added headers for anti-click jacking X-Frame-Options and X-Content-Type-Options, however these errors have not been removed from the OWASP Zap scanning results so this will need to be reviewed. The screenshot below shows the headers have been implemented properly.

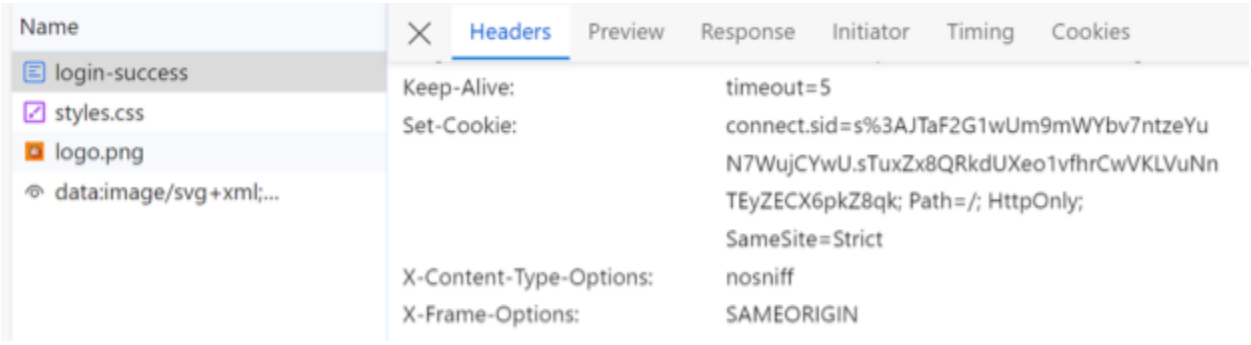


Figure 140 - properly implemented headers

Added a content security policy, but this created more errors in the scanning results.

```
  res.setHeader(
      'Content-Security-Policy',
      "default-src 'self'; script-src 'self'; object-src 'none'; base-uri 'none';"
    );
```

I think this is because we aren't hosting the pages in a specific location, this has been commented on for now and we will need to re-visit when the pages are implemented in the final location.

The anti CRSF changes require substantial code alteration – creating a UUID, and using that in every form. This will be in week 6's sprint, as well as working through the other security issues identified.

Week 6 – security test results

Summary of Alerts

Risk Level	Number of Alerts
High	0
Medium	4
Low	4
Informational	8
False Positives:	0

Figure 141 - OWASP summary

Alerts

Name	Risk Level	Number of Instances
Absence of Anti-CSRF Tokens	Medium	12
Content Security Policy (CSP) Header Not Set	Medium	18
Cross-Domain Misconfiguration	Medium	1
Missing Anti-clickjacking Header	Medium	8
Cross-Domain JavaScript Source File Inclusion	Low	4
Strict-Transport-Security Header Not Set	Low	23
Timestamp Disclosure - Unix	Low	7
X-Content-Type-Options Header Missing	Low	9
Authentication Request Identified	Informational	1
Information Disclosure - Suspicious Comments	Informational	5
Loosely Scoped Cookie	Informational	2
Modern Web Application	Informational	10
Re-examine Cache-control Directives	Informational	18
Retrieved from Cache	Informational	1
Session Management Response Identified	Informational	7
User Controllable HTML Element Attribute (Potential XSS)	Informational	10

Figure 142 - alert summary

The change this week is due to the addition of the ajax elements of the pages. This has introduced a new issue: Web browser data loading may be possible, due to a Cross Origin Resource Sharing (CORS) misconfiguration on the web server.

The code that is flagged is the use of Jquery, which comes from an external domain. Web browser data loading may be possible, due to a Cross Origin Resource Sharing (CORS) misconfiguration on the web server.[29]

To fix this we need to configure the "Access-Control-Allow-Origin" HTTP header and restrict the domains.

Security review of Jquery

Use of jquery also means that the app has a security dependency on jquery. If there were vulnerabilities with jquery, the app code would need to be updated to remove this risk.

We are using jquery 3.5.1 (https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js)

No current vulnerabilities were found with this version using (Jquery : Security vulnerabilities, CVEs (cvedetails.com)).

Version 3.x.x still has security support (jQuery | endoflife.date). The latest version is 3.7.1 which was released on 28 Aug 2023. Jquery 3.x.x is 7 years old, and previous releases have been made end of life between 7 and 10 years since first release. It is a reasonable assumption that v3.x.x could be end of life in the next 3 years and require upgrade to v4 when it is released.

We think we should update Jquery version to the latest production release (version 3.7.1). This would mean that if security issues were found in versions 3.5.1 – 3.7.0 we would not need to update the code.

[29] "Software Security | HTML5: Overly Permissive CORS Policy." *Vulncat.fortify.com*, vulncat.fortify.com/en/detail?id=desc.config.dotnet.html5_overly_permissive_cors_policy

Week 7 – security test results

Summary of Alerts

Risk Level	Number of Alerts
High	0
Medium	5
Low	5
Informational	8
False Positives:	0

Figure 143 - OWASP summary

Alerts

Name	Risk Level	Number of Instances
Absence of Anti-CSRF Tokens	Medium	5
CSP: script-src unsafe-inline	Medium	5
Content Security Policy (CSP) Header Not Set	Medium	8
Cross-Domain Misconfiguration	Medium	1
Missing Anti-clickjacking Header	Medium	8
Cross-Domain JavaScript Source File Inclusion	Low	2
Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)	Low	1
Strict-Transport-Security Header Not Set	Low	15
Timestamp Disclosure - Unix	Low	7
X-Content-Type-Options Header Missing	Low	9
Authentication Request Identified	Informational	1
Information Disclosure - Suspicious Comments	Informational	3
Loosely Scoped Cookie	Informational	1
Modern Web Application	Informational	4
Re-examine Cache-control Directives	Informational	13
Retrieved from Cache	Informational	1
Session Management Response Identified	Informational	5
User Controllable HTML Element Attribute (Potential XSS)	Informational	3

Figure 144 - Alert summary

There are medium issues that cannot be resolved because they are related to the ajax.google.com website. This website needs these security settings because it is providing javascript for other websites. The medium impact issues that are caused by this are:

- Content Security Policy (CSP) Header Not Set – allows access by any website
- Cross-Domain Misconfiguration – allows calling by any website

A stronger Content Security Policy was implemented (below). This tightly restricts the sources of scripts, images and access. When the website is put into production this will

need to be altered. Consider putting this information in the .env file to make it easier for installation.

```
default-src http://localhost:8000;      // will need changing when installed
script-src http://localhost:8000 'unsafe-inline' https://ajax.googleapis.com;
//unsafe-inline allows inline scripts
img-src 'self' https://cdn3.iconfinder.com https://i.imgur.com;
style-src http://localhost:8000;
font-src http://localhost:8000;
frame-ancestors http://localhost:8000;
// stops the page being used within external websites
form-action http://localhost:8000;
// only allow form submission to the server running the app
```

There is one medium issue, caused by the fact that the pages contain in-line javascript, that still exists based on the Content Security Policy.

- CSP: script-src unsafe-inline

Week 8 – security test results

Summary

Risk Level	Number of Alerts	Change from last week
High	0	0
Medium	5	0
Low	5	0
Informational	8	0
False Positives:	0	0

[Figure 145 - OWASP summary]

Summary of alerts

Name	Risk Level	Number of Instances	Change from last week and comment
Absence of Anti-CSRF Tokens	Medium	74	+69 Additional pages added this week with this issue
CSP: script-src unsafe-inline	Medium	15	+10 Additional pages added this week with this issue
Content Security Policy (CSP) Header Not Set	Medium	8	0
Cross-Domain Misconfiguration	Medium	1	-1
Missing Anti-clickjacking Header	Medium	8	0

Cross-Domain JavaScript Source File Inclusion	Low	9	-7
Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)	Low	1	0
Strict-Transport-Security Header Not Set	Low	22	-7
Timestamp Disclosure - Unix	Low	7	0
X-Content-Type-Options Header Missing	Low	9	0
Authentication Request Identified	Informational	1	1
Information Disclosure - Suspicious Comments	Informational	8	3
Loosely Scoped Cookie	Informational	1	0
Modern Web Application	Informational	14	+10 Additional Pages Using Ajax
Re-examine Cache-control Directives	Informational	17	-4
Retrieved from Cache	Informational	3	+2
Session Management Response Identified	Informational	11	-6
User Controllable HTML Element Attribute (Potential XSS)	Informational	13	+10 New Pages with parameters in post body (sanitised in app.js)

[Figure 146 - OWASP summary]

This week the focus has been getting all the v0 prototype pages completed, and there has not been a concentrated focus of the security corrections. The additional pages have increased the alerts, because they have been based on the other pages, which still have some alerts that need to be resolved. While there are more instances of security issues, there have been no new types of security issues introduced.

Week 9 – security test results - aggressive testing of version 0

In line with the security strategy when there is a version change more aggressive testing is performed. Now version 0 is complete, aggressive testing can take place.

This testing was carried out using OWASP Zap, with the ajax spider and the aggressive SQL injection add-in installed.

A special session in OWASP was set up for this test, called attack simulation against v0 of bookit app, and authentication rules were created so OWASP Zap could automatically log in.

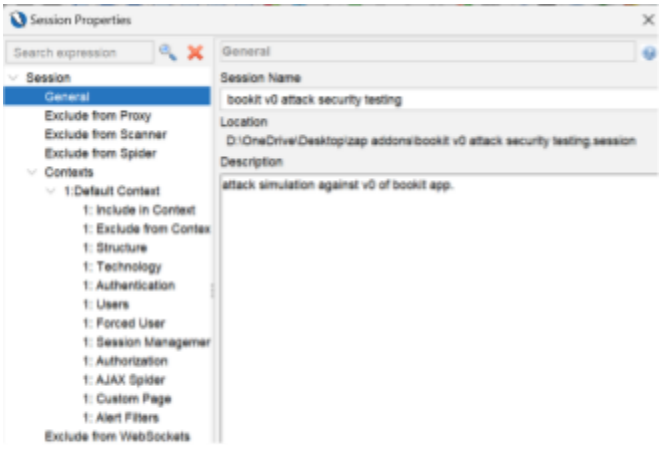


Figure 147 - attack simulation properties

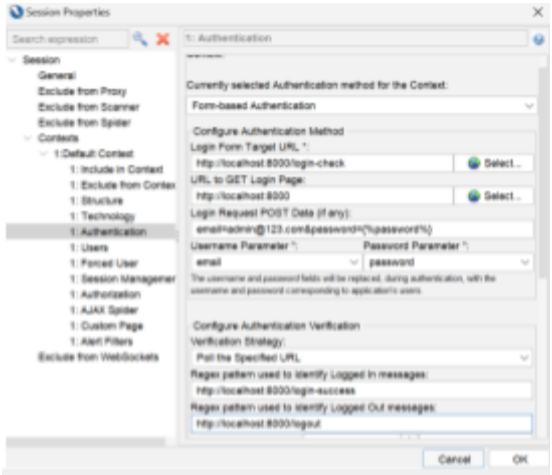


Figure 148 - Authentication properties

Each page has a logout button, so this was excluded from the scan. Otherwise as soon as the scan logged in, it would log itself out again by following the logout link which was one of the first on every page.

OWASP Zap was configured to run comprehensive tests, the Advanced SQL Injection add-in was also installed for additional testing. The full list of tests (how many individual tests were completed, and how long the testing took) is in the table below.

Tests carried out

	Strength	Progress	Elapsed Time	Tests	Status
Analyser			00:00.027	3	
Plugin					
Path Traversal	Medium	100	00:03.591	99	Completed

Remote File Inclusion	Medium	100	00:03.096	90	Completed
Source Code Disclosure - CVE-2012-1823	Medium	100	00:00.001	0	Skipped, scan rule does not target selected technologies.
Remote Code Execution - CVE-2012-1823	Medium	100	00:00.000	0	Skipped, scan rule does not target selected technologies.
Heartbleed OpenSSL Vulnerability	Medium	100	00:00.014	3	Completed
Source Code Disclosure - /WEB-INF Folder	Medium	100	00:00.037	4	Completed
External Redirect	Medium	100	00:02.610	81	Completed
Server Side Include	Medium	100	00:01.110	36	Completed
Cross Site Scripting (Reflected)	Medium	100	00:01.428	45	Completed
Cross Site Scripting (Persistent) - Prime	Medium	100	00:00.315	9	Completed
Cross Site Scripting (Persistent) - Spider	Medium	100	00:00.196	13	Completed
Cross Site Scripting (Persistent)	Medium	100	00:00.011	0	Completed
SQL Injection	Medium	100	00:08.107	242	Completed
SQL Injection - MySQL	Medium	100	00:03.041	90	Completed
SQL Injection - Hypersonic SQL	Medium	100	00:00.001	0	Skipped, scan rule does not target selected technologies.
SQL Injection - Oracle	Medium	100	00:00.000	0	Skipped, scan rule does not target selected technologies.
SQL Injection - PostgreSQL	Medium	100	00:00.000	0	Skipped, scan rule does not target selected technologies.
SQL Injection - SQLite	Medium	100	00:00.000	0	Skipped, scan rule does not target selected technologies.
Cross Site Scripting (DOM Based)	Medium	100	00:00.494	0	Skipped, failed to start or connect to the browser.
SQL Injection - MsSQL	Medium	100	00:00.000	0	Skipped, scan rule does not target selected technologies.

Log4Shell	Medium	100	00:00.000	0	Skipped, scan rule does not target selected technologies.
Spring4Shell	Medium	100	00:00.000	0	Skipped, scan rule does not target selected technologies.
Server Side Code Injection	Medium	100	00:00.000	0	Skipped, scan rule does not target selected technologies.
Remote OS Command Injection	Medium	100	00:10.557	315	Completed
XPath Injection	Medium	100	00:00.919	27	Completed
XML External Entity Attack	Medium	100	00:00.012	0	Completed
Generic Padding Oracle	Medium	100	00:00.012	0	Completed
Cloud Metadata Potentially Exposed	Medium	100	00:00.022	4	Completed
Server Side Template Injection	Medium	100	00:04.013	126	Completed
Server Side Template Injection (Blind)	Medium	100	00:01.147	36	Completed
Directory Browsing	Medium	100	00:00.226	13	Completed
Buffer Overflow	Medium	100	00:00.001	0	Skipped, scan rule does not target selected technologies.
Format String Error	Medium	100	00:00.000	0	Skipped, scan rule does not target selected technologies.
CRLF Injection	Medium	100	00:01.924	63	Completed
Parameter Tampering	Medium	100	00:02.201	63	Completed
ELMAH Information Leak	Medium	100	00:00.006	1	Completed
Trace.axd Information Leak	Medium	100	00:00.013	1	Completed
.htaccess Information Leak	Medium	100	00:00.001	0	Skipped, scan rule does not target selected technologies.
.env Information Leak	Medium	100	00:00.013	1	Completed
Spring Actuator Information Leak	Medium	100	00:00.001	0	Skipped, scan rule does not target selected technologies.

Hidden File Finder	Medium	100	00:00.252	50	Completed
XSLT Injection	Medium	100	00:00.630	24	Completed
GET for POST	Medium	100	00:00.015	2	Completed
User Agent Fuzzer	Medium	100	00:02.297	156	Completed
Script Active Scan Rules	Medium	100	00:00.001	0	Skipped, no scripts enabled.
Advanced SQL Injection	Medium	100	00:55.126	1611	Completed
SOAP Action Spoofing	Medium	100	00:00.013	0	Completed
SOAP XML Injection	Medium	100	00:00.013	0	Completed
Totals			01:43.188	3236	

[Figure 149 - OWASP test summary]

In total 3,336 requests were made to the application to test the above vulnerabilities.

URLs tested

The details of the URLs tested in this attack are set out below.

Figure 150 - URLs tested

Figure 151 - URLs tested

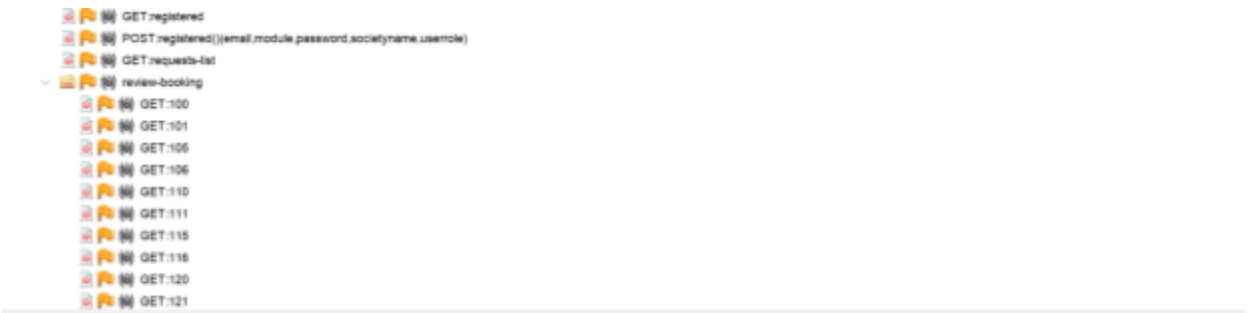


Figure 152 - URLs tested

Figure 153 - URLs tested

Issues identified and raised to backlog

- SQL-Injection detected. Login-check. Is it possible to post admin@123.com" AND "1"="1" -- as the email field in the post body and it will get executed in mysql. You can't enter this data on the login form because there is a requirement for it to be formatted as an email. On review the login-check SQL is not parameterised with ?'s. Fix is to parameterise. Logged as backlog issue 38.
- Previously identified issues will now be added to the v1 backlog for correction.
- Security. V0 aggressive test. Medium. Absence of Anti-CSRF tokens. 72 instances. A unique number or token should be generated and passed to the .ejs page so it can be submitted back with any FORM POST. When the response is received by the app.post route this number should be checked to make sure that it is the same number that was passed to the page. Research shows some routes to implement in node.js and express however we are using Ajax and we have inline javascript.[30] This issue is evidenced because the form html elements don't have a unique token in them, an example of one form would be the form that posts to login-check in the / route, but it applies to ALL forms in the application. Logged in the backlog as issue #39.
- Security. v0 aggressive test. Medium. Content Security Policy: default-src 'none' is being flagged as a wildcard. There are 9 instances in the app. This setting tells the browser what the approved sources of content are. Because it is set to none, this is a risk. This needs altering and testing in the app.use section where the security

[30] Khan, Sikandar. "CSRF Tokens in ExpressJS — Node.js Web Framework." *Medium*, 27 Mar. 2022, dearsikandarkhan.medium.com/csrf-tokens-in-expressjs-node-js-web-framework-cc331069de2d

headers are currently set. Evidence can be found by examining the headers of the page in developer tools > network in the browser when using the pages. Logged in the backlog as issue #40.

- Security. v0 aggressive test. Medium. Content Security Policy: script-src unsafe-inline. This has been flagged because javascript is allowed in the pages by the content security policy. The pages use javascript inline, so this will need to be considered and possibly the javascript will need to be moved to a folder on the server, and that folder be allowed via the content security policy. There are 42 instances of this flagged. Logged in the backlog as issue #41.
- Security. v0 aggressive test. Low. Cross-Domain JavaScript Source File Inclusion. This issue is because the code uses Ajax and this is imported from google from a URL which is not on the site that bookit is hosted on. This is by design and cannot be mitigated. <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script> is the line that causes the problem in the pages. This cannot be mitigated and has not been raised to the backlog.
- Security. v0 aggressive test. Low. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s). This issue is because express adds “X-Powered-By: Express” into each header. Attempts have been made in the app.use part of app.js to block it, but these have not worked. There are 4 instances of this. May have to consider using npm helmet to manage the headers rather than doing it manually. Logged in backlog as issue #42.
- Security. v0 aggressive test. Low. X-Content-Type-Options Header Missing. It looks like for images served from the /media and the css served from the /styles location there is no content type set or it isn’t set to ‘nosniff’. Research is required to see if a content header can be set for these folders which when used are accessed via / in the application. Examples are styles.css. person-green.png etc, but no pages. Logged as backlog issue #43.

The process of using OWASP ZAP to check for SQL injection and other vulnerabilities was a complicated one, however, the tests were comprehensive and tested permutations that we would not have thought of if we had manually designed the tests. The issue identified needed to be manually tested to prove that it was an actual issue.

Security testing of v0 full report

The report summary after issue 38 was corrected and a re-test was carried out is below. It shows no high issues. The remaining issues will be reviewed as part of the backlog in v1 and at the end of v1 another aggressive test will be carried out. The full report can be downloaded from here Security v0 full test results 2024-03-05-ZAP-Report-.zip (github.com). A summary not containing remediation suggestions and evidence of issues is below:

ZAP Scanning Report

Generated on Tue, 5 Mar 2024 15:01:41

ZAP Version: 2.14.0

Figure 154 - ZAP report

Summary of Alerts

Risk Level	Number of Alerts
High	0
Medium	3
Low	3
Informational	7

Figure 155 - OWASP summary

Alerts

Name	Risk Level	Number of Instances
Absence of Anti-CSRF Tokens	Medium	75
CSP: Wildcard Directive	Medium	9
CSP: script-src unsafe-inline	Medium	47
Cross-Domain JavaScript Source File Inclusion	Low	32
Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)	Low	4
X-Content-Type-Options Header Missing	Low	4
Authentication Request Identified	Informational	5
Information Disclosure - Suspicious Comments	Informational	51
Loosely Scoped Cookie	Informational	14
Modern Web Application	Informational	46
Session Management Response Identified	Informational	42
User Agent Fuzzer	Informational	108
User Controllable HTML Element Attribute (Potential XSS)	Informational	30

Figure 156 - Alert summary

Final security testing - Aggressive testing of v1

It is important to note that all security testing was conducted by providing a username and password to the automated testing tool and disabling two factor authentication. These attacks all assume that an attacker already has a valid user name, password and two factor authentication. Having username, password and two factors in place for access to the system greatly reduces the chance of any attacker getting to the point which is being tested by the automated testing, but it is still important to test for vulnerabilities from insiders or in case access was obtained to the system.

The automated testing for v1 was carried out with the same testing tool, OWASP Zap. The range of tests was expanded from the version 0 testing.

A detailed test was carried out. It ran for 92 minutes and made 10,992 tests of the system. The end of the testing report is shown below.

Host:	http://localhost:8000				
	Strength	Progress	Elapsed	Reqs	Alerts
GET for POST	Medium		00:00.100	14	0
User Agent Fuzzer	Medium		06:40.146	814	221
Script Active Scan Rules	Medium		00:00.000	0	0
Advanced SQL Injection	Medium		21:33.214	5586	1
SOAP Action Spoofing	Medium		00:00.047	0	0
SOAP XML Injection	Medium		00:00.031	0	0
Totals			92:57.856	10992	222

Figure 157 - detailed test report

There was one high issue flagged by the Advanced SQL Injection testing (which made 5,586 tests in total). This one issue related to the registered page, which is used to create a new user when logged in as a user which has an admin role. The evidence below shows that additional elements can be added to the password parameter when the form on that page is submitted.

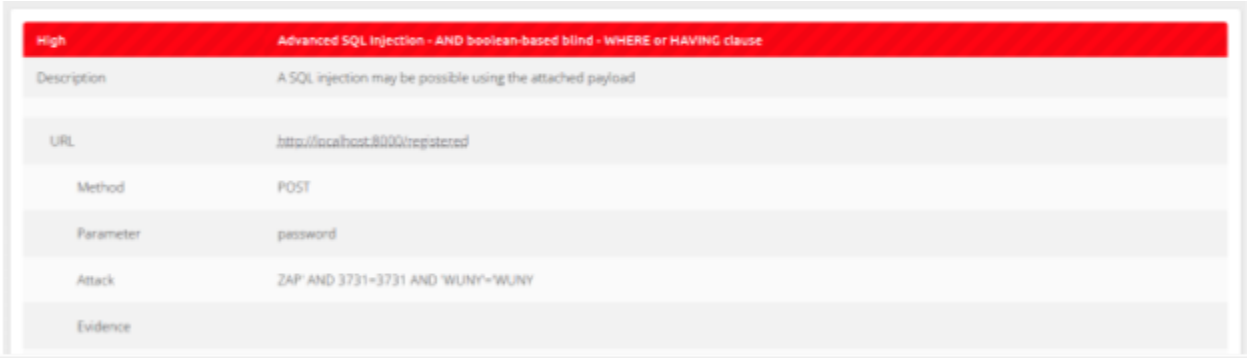


Figure 158 - High alert for advanced SQL injection

A review of the code for registered shows that all the SQL is parameterized, the data is sanitised and there is regex validation on the password. The code shows that the hashedPassword is what is inserted into the database and not the password typed. So this

might be a false positive, because at no point is the password from the body of the post is used in any SQL.

```
// the form in register.ejs has a target of /registered which will get managed by this code.
// this form uses http POST
app.post("/registered", isLoggedIn, function (req, res) {
  var email = sanitiseHtml(req.body.email);
  var password = sanitiseHtml(req.body.password);
  var userrole = sanitiseHtml(req.body.userrole);
  var societyname = sanitiseHtml(req.body.societyname);
  var module = sanitiseHtml(req.body.module);

  //pasword regex to check if the password meets the requirements
  var passwordRegex = /^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[a-zA-Z]).{8,}$/;

  // Check if the password meets the requirements
  if (!passwordRegex.test(password)) {
    db.query("SELECT * FROM lookup_user_role", (err, data) => {
      if (err) {
        console.error(err.message);
        return res.redirect("/");
      }
      // render the register-error page with the error message
      res.render("register-error.ejs", {
        errorMessage: "Password must be at least 8 characters long and include at least one
lowercase letter, one uppercase letter, and one digit.",
        formValues: { email, userrole, societyname, module },
        allUserRoles: data
      });
    });
  } else {
    var hashedPassword = hashPassword(password);
    let sqlqueryuser = "INSERT INTO user_account (email, password, user_role, society_name,
module) VALUES (?,?,?,?,?)";
    let newuser = [email.toLowerCase(), hashedPassword, userrole, societyname, module];

    db.query(sqlqueryuser, newuser, (err, result) => {
      if (err) {
        console.error(err.message);
        res.redirect("./");
      } else {
        res.redirect("login-success");
      }
    });
  }
});
```

[Figure 159 - False positive evidence for SQL injection]

Testing using the app shows that this isn't a valid input, because of the regex checks

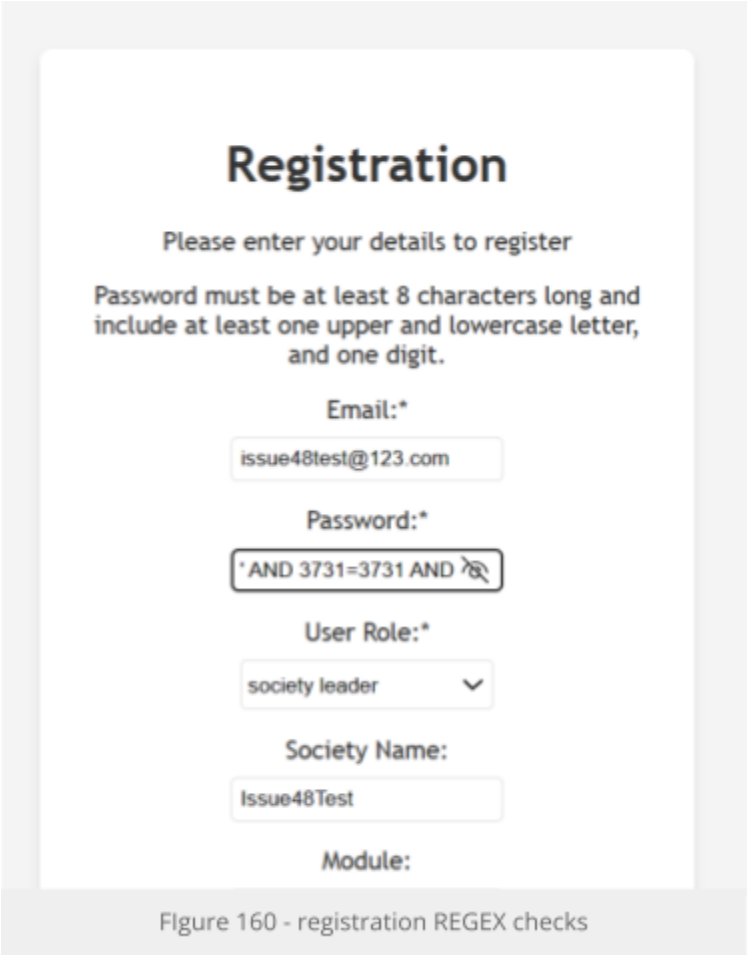


FIgure 160 - registration REGEX checks

So this shows that this password doesn't meet the regex criteria and therefore is not used as a password.

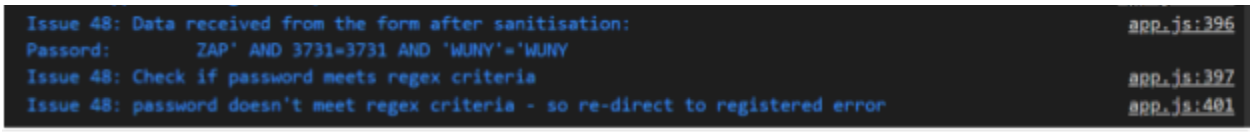


Figure 161 - Regex password checks routing

If we try with a lowercase character then we would pass the regex testing so lets try with *zap' and 3731=3731 AND 'WUNY'='WUNY*

Figure 162 - secondary SQL injection test

In this example the password is accepted by the system as it meets the regex criteria, but it is not used in the insert statement, the hashed password is used instead. This does mean that we have a user with an unhashed password of *zap' and 3731=3731 AND 'WUNY'='WUNY*

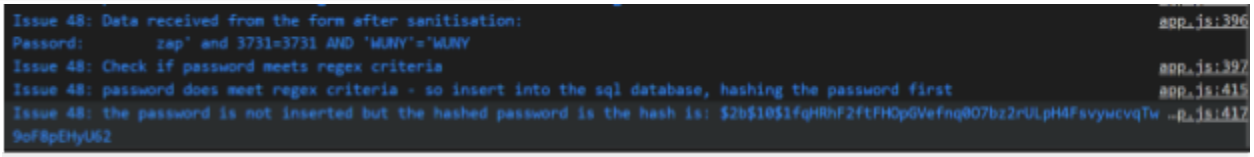


Figure 163 - password routing REGEX checks

Now testing if using this password for this user causes any SQL injection issues

Figure 164 - SQL injection test

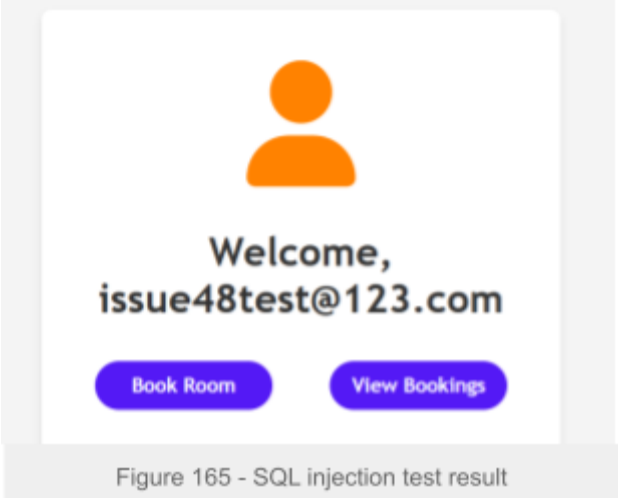


Figure 165 - SQL injection test result

The result is that this user can log in with no problems and no SQL injection risks. This issue will appear on the report but it has been proven to be a false positive and poses no risk to the system or the data within it.

The 221 user agent fuzzing issues found in the testing are marked as informational (which is below low risk). These tests randomly change parameters in posts to see what happens. We will not be addressing these issues within the project as they are considered very low risk.

The summary of the full report on version 1 is below

Figure 166 - ZAP report breakdown

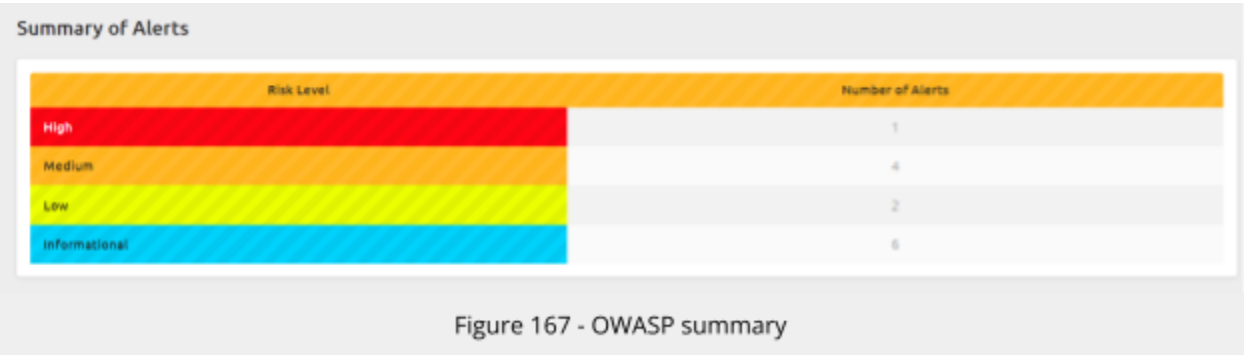


Figure 167 - OWASP summary

Figure 168 - alert summary

The final commentary for the high and medium issues is:

- **High: Advanced SQL Injection - AND boolean-based blind - WHERE or HAVING clause**
  This has been demonstrated to be a false positive, the password parameter is not used in any SQL within the code, and the code is parameterised.
- **Medium: Absence of Anti-CSRF Tokens**
  This is flagged as issue #39 in github. This issue should be resolved before the system is production ready. The system is protected from casual attacks because there is a username, password and two factor authentication in place, but for a full production version this should be implemented by adding a unique number (a nonce) to each transaction and passing this value to the page and it being returned in a hidden field within the form. Due to the length of time taken trying to resolve differences between development and the production environment this issue was not completed, but it is important.
- **Medium: CSP: script-src unsafe-inline**
  This is flagged as issue #41 in github. There is inline javascript in the pages, this relates to the AJAX used in the filter mechanisms. This should be moved to the /public folder as a .js before the system is production ready.
- **Medium: CSP: style-src unsafe-inline**
  This is required for the tinyMCE module to function. Without this entry tinyMCE will not display, this is a dependency that we have not found a way to remove.
  There are two issues that need to be resolved before this system is production ready, but overall the system has good security hygiene.

  Because threats change over time, even if the code is not changed, we recommend that OWASP zap is used to test any deployment at least every 2 months. A developer walkthrough video is in the github repository to explain how to do this.

Test plan - post development (Unit testing)

We carried out another set of tests after the production of V1. Alongside the user-centred tests that are detailed previously so we could figure out where Bookit succeeded and failed respectively from a usability and user experience perspective; this was to be a test plan for the entire application and therefore we felt it necessary to ensure that each individual component of Bookit was in adequate working order functionality-wise as well. This would therefore ensure that usability is up to an acceptable standard both from the user experience side of things as well as making sure that the app actually works in all relevant scenarios.

Rafał Miecznik states that unit testing is essential to ensure your documentation is comprehensive, your code practices are good - and that you catch bugs before making big changes to your application.[31] For these reasons we decided to use unit testing alongside our user testing.

This test plan is given below on a page-by-page basis for the entire web application.

add-booking

Test No.	Test case	Test type	Expected result	Actual result	Action required
1	Loading the page	Functional	User and room details are displayed - including level of clearance, room type, building and booking status	User and room details are displayed - including level of clearance, room type, building and booking status	N/A
2	Loading the page	Functional	An empty risk assessment template is created	An empty risk assessment template is created	N/A
3	Filling in the risk assessment	Functional	Typing within the risk assessment box allows for you to word-process the entire risk	Typing within the risk assessment box allows for you to word-process the entire risk assessment - tables	N/A

[31] Miecznik, Rafał . "The Importance and Benefits of Unit Testing." CodiLime, 22 June 2023, codilime.com/blog/unit-testing/

			assessment - tables function as expected and all text is visible	function as expected and all text is visible	
4	Clicking the “Cancel” button	Functional	Redirected back to the rooms-list page - all booking and risk assessment data is discarded	Redirected back to the rooms-list page - all booking and risk assessment data is discarded	N/A
5	Click on the “request booking” button	Functional	Redirects to the add-booking-submit page with a verification message	Redirects to the add-booking-submit page with a verification message	N/A

[Figure 169 - add-booking unit tests]

add-room

Test No.	Test case	Test type	Expected result	Actual result	Action required
6	Clicking the “Add room” button without filling in required fields (Room number, seating capacity)	Functional	Front end does not let you submit the form - prompting the fields to be filled in	Front end does not let you submit the form - prompting the fields to be filled in	N/A
7	Clicking the Add room button when all necessary fields are populated	Functional	Redirects to add-room-success with a confirmation message	Redirects to add-room-success with a confirmation message	N/A
8	Clicking on “Choose file”	Functional	Allows you to upload an image file from your system	Allows you to upload an image file from your system	N/A
9	Uploading a file that is not an image	Functional/ User experience	The user is prompted to use a valid file format	Bookit throws an error: “Only image files are allowed”	Change this to be more user-friendly, it does not break

					the functioning of the site but is not nice to experience

[Figure 170 - add-room unit tests]

approved-list

Test No.	Test case	Test type	Expected result	Actual result	Action required
10	Loading the page	Functional	Shows all bookings with an “Approved” status	Shows all bookings with an “Approved” status	N/A
11	Clicking the “View” button	Functional	Redirects to the view-booking page for that particular booking	Redirects to the view-booking page for that particular booking	N/A
12	Clicking the “Order by Building” button	Functional	Bookings are ordered by their building with regards to how they are in the list of rooms (RHB first, WB second)	Bookings are ordered by their building with regards to how they are in the list of rooms (RHB first, WB second)	N/A
13	Clicking the “Order by Upcoming Date/Time” button	Functional	Bookings are ordered in ascending date/time according to the starting time	Bookings are ordered in ascending date/time according to the starting time	N/A
14	Clicking the “Order by most recently approved” button	Functional	Bookings are ordered by the most recently approved booking to the least recently approved	Bookings are ordered by the most recently approved booking to the least recently approved	N/A

[Figure 171 - approved-list unit tests]

bookings-list

Test No.	Test case	Test type	Expected result	Actual result	Action required
15	Loading the page	Functional	Bookings that were made by the currently logged in user have an edit button and a view button - while all other bookings just have a view button	Bookings that were made by the currently logged in user have an edit button and a view button - while all other bookings just have a view button	N/A
16	Clicking the edit button	Functional	Redirects to the edit-booking page for that particular booking	Throws an error: this.buffer.charCodeAt is not a function	Fix the redirect for this button - a token is missing
17	Clicking the view button	Functional	Redirects to the view-booking/id page for that particular booking	Redirects to the view-booking/id page for that particular booking	N/A
18	Clicking the “Order by Building” button	Functional	Bookings are ordered by their building with regards to how they are in the list of rooms (RHB first, WB second)	Bookings are ordered by their building with regards to how they are in the list of rooms (RHB first, WB second)	N/A
19	Clicking the “Order by Upcoming Date/Time button	Functional	Bookings are ordered in ascending date/time according to the starting time	Bookings are ordered in ascending date/time according to the starting time	N/A
20	Clicking the “Order by Status” button	Functional	Bookings are grouped in the following order: Approved, Awaiting approval, Rejected	Bookings are grouped in the following order: Approved, Awaiting approval, Rejected	N/A

[Figure 172 - bookings-list unit tests]

credits

Test No.	Test case	Test type	Expected result	Actual result	Action required
21	Loading the page	Front-end aesthetics	Displays the names of all group members	Displays the names of all group members	N/A

[Figure 173 - credits unit tests]

edit-booking

Test No.	Test case	Test type	Expected result	Actual result	Action required
22	Loading page	Functional	Displays the booking info alongside the risk assessment	Displays the booking info alongside the risk assessment	N/A
23	Editing the risk assessment	Functional	The risk assessment is editable as if it were a document	The risk assessment is editable as if it were a document	N/A
24	Clicking the “Back” button	Functional	Redirects to bookings-list	Redirects to bookings-list	N/A
25	Clicking the “Update Booking” button	Functional	Inserts the newly updated risk assessment for that booking into the database - redirects to bookings-list	Inserts the newly updated risk assessment for that booking into the database - redirects to bookings-list	N/A

[Figure 174 - edit-booking unit tests]

edit-room

Test No.	Test case	Test type	Expected result	Actual result	Action required
26	Clicking the “Save” button without filling in required fields (Room number, seating capacity)	Functional	Front end does not let you submit the form - prompting the fields to be filled in	Front end does not let you submit the form - prompting the fields to be filled in	N/A

27	Clicking the Save button when all necessary fields are populated	Functional	Redirects to edit-room-success with a confirmation message	Redirects to edit-room-success - but this boots you back to the main menu with no confirmation	Add some sort of confirmation message - user experience
28	Clicking on “Choose file”	Functional	Allows you to upload an image file from your system	Allows you to upload an image file from your system	N/A
29	Uploading a file that is not an image	Functional/User experience	The user is prompted to use a valid file format	Bookit throws an error: “Only image files are allowed”	Change this to be more user-friendly, it does not break the functioning of the site but is not nice to experience

[Figure 175 - edit-room unit tests]

edit-rooms-list

Test No.	Test case	Test type	Expected result	Actual result	Action required
30	Clicking the “Order by Building” button	Functional	Rooms are ordered by the building they are in - grouped with RHB first, WB second, etc.	App throws a typeerror	Seems the order code is copied from the bookings - which has more attributes to check that don’t exist for the rooms - rewrite the code so it works for the rooms as well
31	Clicking the “Order by Capacity” button	Functional	Orders the rooms by seating capacity - largest first	App throws a typeerror	Seems the order code is copied from the bookings - which has more attributes to check that don’t exist for the

					rooms - rewrite the code so it works for the rooms as well
32	Clicking the “Edit” button for a room	Functional	Redirects to the edit-room page for that booking	Redirects to the edit-room page for that booking	N/A
33	Clicking the “Delete” button for a room	Functional	A confirmation alert appears asking if you’re sure	A confirmation alert appears asking if you’re sure	N/A
34	Clicking “OK” on the deletion alert	Functional	The room is removed from the database	The room is removed from the database	N/A
35	Clicking “Cancel” on the deletion alert	Functional	The alert disappears - nothing changes	The alert disappears - nothing changes	N/A

[Figure 176 - edit-rooms-list unit tests]

faq

Test No.	Test case	Test type	Expected result	Actual result	Action required
36	Loading the page	Functional	Four boxes appear with paragraphs on the most frequent questions	Four boxes appear with paragraphs on the most frequent questions	N/A

[Figure 177 - faq unit tests]

filters

Test No.	Test case	Test type	Expected result	Actual result	Action required
37	Selecting a date from the calendar	Functional	Reduces the list to what is available/occuring on that date	Reduces the list to what is available/occuring on that date	N/A
38	Attempting to type in a date in the past on the rooms-list page	Functional	Automatically sets the date to today’s date	Automatically sets the date to today’s date - but you can	Restrict time slots on this page accordingly

				still choose a time slot that has already passed	
39	Sliding the duration slider	Front-end aesthetics	Text for the duration updates accordingly in the input box	Text for the duration updates accordingly in the input box	N/A
40	Typing in a duration	Front-end aesthetics	Slider moves accordingly	Slider moves accordingly	N/A
41	Attempting to type in a duration larger than 4 hours	Functional	Automatically sets it to the max of 4 hours - both the slider and the input box	Automatically sets it to the max of 4 hours - both the slider and the input box	N/A
42	Moving the Seating Available slider	Front-end aesthetics	Updates the text to show how many seats are being requested accordingly	Updates the text to show how many seats are being requested accordingly	N/A
43	Clicking the “Reset” button	Functional	Resets the list to its original state upon page loading	Resets the list to its original state upon page loading	N/A
44	Choosing a building from the drop down menu	Functional	Reduces the list to only include rooms in those buildings	Reduces the list to only include rooms in those buildings	N/A
45	Choosing a room type from the drop down menu	Functional	Reduces the list to only include rooms of that type	Reduces the list to only include rooms of that type	N/A
46	Selecting a timeslot (rooms-list)	Functional	Shows rooms that are available at that time slot	Shows rooms that are available at that time slot	Time slot selection also appears on other pages besides rooms-list - perhaps remove?
47	Selecting a duration (rooms-list)	Functional	Shows rooms that are available at that duration	Shows rooms that are available at that duration	Duration selection also appears on other

					pages besides rooms-list - perhaps remove?

[Figure 178 - filters unit tests]

footer

Test No.	Test case	Test type	Expected result	Actual result	Action required
48	Clicking the “credits” button	Functional	Redirects to credits	Redirects to credits	N/A
49	Clicking the “support email” button	Functional	Opens your default email application and sets up an email with the address support@bookit.com	Opens your default email application and sets up an email with the address support@bookit.com	N/A
50	Clicking the “report bug” button	Functional	Redirects to report-bug	Redirects to report-bug	N/A
51	Clicking the “FAQ/Help” button	Functional	Redirects to faq	Redirects to faq	N/A

[Figure 179 - footer unit tests]

Headers (multiple different headers to be tested under this plan)

Test No.	Test case	Test type	Expected result	Actual result	Action required
52	Clicking the “dark mode” checkbox	Aesthetics	Changes the site to/from dark mode	Changes the site to/from dark mode	N/A
53	Clicking the “logout” button	Functional	Deletes the session and logs out the current user - any user welcome text should now say “not logged in”	Deletes the session and logs out the current user - any user welcome text now says “not logged in”	N/A
54	Being logged in	Aesthetics/UX	Text displays “Logged in as [USER EMAIL]”	Text displays “Logged in as [USER EMAIL]”	N/A

55	Clicking the Bookit logo in the top left	Functional	Redirects to login-success	Redirects to login-success	N/A
56	Clicking the “Menu” button	Functional	Redirects to login-success	Redirects to login-success	N/A

[Figure 180 - headers unit tests]

login-2fa

Test No.	Test case	Test type	Expected result	Actual result	Action required
57	Generating QR codes based on user login/Email	Functional	“True” Boolean result in code redirects to login-success	Boolean True Login Success	N/A
58	QR codes scannable by authenticator app	Functional	New QR code per login	New qr code per login - is scannable	N/A
59	QR code on screen for user to scan	Aesthetics /UX	QR code stays proportional on screen and large enough for a phone camera to scan.	Result as expected.	N/A
60	Alt state if QR code image fails to load	Aesthetics /UX/Functional	Placeholder provides a link which sources directly to a new tab holding the image of the QR code.	Result as expected - the placeholder stores a link to the generated qr code itself.	N/A
61	2FA checks per login/logout	Functional	a 2FA check should pop up for every new attempt at logging in.	2FA prompt appears during new login attempt.	N/A

[Figure 181 - login-2fa unit tests]

login-error

Test No.	Test case	Test type	Expected result	Actual result	Action required
62	Attempting to press the login button/enter key without entering a valid email	Functional	The page correctly prompts for the field to be filled in if it is empty - and for the correct email format if the entered value does not contain the name@address format	The page correctly prompts for the field to be filled in if it is empty - and for the correct email format if the entered value does not contain the name@address format	N/A
63	Attempting to press the login button/enter key without entering a valid password	Funcional	The page prompts for the password field to be filled in if it is empty.	The page prompts for the password field to be filled in if it is empty.	N/A
64	Inputting the incorrect login information	Functional	The page renders the login-check url - which redirects to the login-error page to show that the user information doesn't match with any database records.	The page renders the login-check url - which redirects to the login-error page to show that the user information doesn't match with any database records.	N/A
65	Inputting the correct login information	Functional	The login page redirects you to the 2-Factor authentication screen	The login page redirects you to the 2-Factor authentication screen	N/A

[Figure 182 - login-error unit tests]

login-success

Test No.	Test case	Test type	Expected result	Actual result	Action required
66	Loading into the page for the first time should display the	Functional	Page displays your email	Page displays your email	N/A

	correct email				
67	Logging in as an admin	Functional	Profile picture is green - options for pages to access are: Book Room View Bookings View Requests View Approved Register User Add Room Edit Rooms	Profile picture is green - options for pages to access are: Book Room View Bookings View Requests View Approved Register User Add Room Edit Rooms	N/A
68	Logging in as a coordinator	Functional	Profile picture is green - options for pages to access are: Book Room View Bookings View Requests View Approved	Profile picture is green - options for pages to access are: Book Room View Bookings View Requests View Approved	N/A
69	Logging in as a standard user	Functional	Profile picture is orange - options for pages to access are: Book Room View Bookings	Profile picture is orange - options for pages to access are: Book Room View Bookings	N/A
70	Clicking on Book room button	Functional	Redirects to the rooms-list page	Redirects to the rooms-list page	N/A
71	Clicking on View requests button	Functional	Redirects to the requests-list page	Redirects to the requests-list page	N/A
72	Clicking on the Register user button	Functional	Redirects to the register page	Redirects to the register page	N/A
73	Clicking on the Edit rooms page	Functional	Redirects to the edit-rooms-list page	Redirects to the edit-rooms-list page	N/A
74	Clicking on the View bookings button	Functional	Redirects to the bookings-list page	Redirects to the bookings-list page	N/A
75	Clicking on the View Approved button	Functional	Redirects to the Approved-list page	Redirects to the Approved-list page	N/A
76	Clicking on the Add room	Functional	Redirects to the add-room page	Redirects to the add-room page	N/A

[Figure 183 - login-success unit tests]

login

Test No.	Test case	Test type	Expected result	Actual result	Action required
77	Attempting to load the site with no user currently logged in	Functional	The login page renders as if no session has previously been established.	The login page renders as if no session has previously been established.	N/A
78	Attempting to press the login button/enter key without entering a valid email	Functional	The page correctly prompts for the field to be filled in if it is empty - and for the correct email format if the entered value does not contain the name@address format	The page correctly prompts for the field to be filled in if it is empty - and for the correct email format if the entered value does not contain the name@address format	N/A
79	Attempting to press the login button/enter key without entering a valid password	Funcional	The page prompts for the password field to be filled in if it is empty.	The page prompts for the password field to be filled in if it is empty.	N/A
80	Inputting the incorrect login information	Functional	The page renders the login-check url - which redirects to the login-error page to show that the user information doesn't match with any database records.	Bookit crashed.	There was a minor code error - making it so the code to handle incorrect information would only be executed should the login info actually be correct - moved this code block outside of that condition - no further

					problems.
81	Inputting the correct login information	Functional	The login page redirects you to the 2-Factor authentication screen	The login page redirects you to the 2-Factor authentication screen	N/A

[Figure 184 - login unit tests]

register-error

Test No.	Test case	Test type	Expected result	Actual result	Action required
82	Attempting to press the register button/enter key without entering a valid email	Functional	The page correctly prompts for the field to be filled in if it is empty - and for the correct email format if the entered value does not contain the name@address format	The page correctly prompts for the field to be filled in if it is empty - and for the correct email format if the entered value does not contain the name@address format	N/A
83	Attempting to press the register button/enter key without entering a valid password	Functional	The page prompts for the password field to be filled in if it is empty.	The page prompts for the password field to be filled in if it is empty.	N/A
84	Attempting to enter a password that does not match the following criteria: Password must be at least 8 characters long and include at least one lowercase letter, one uppercase letter, and one digit.	Functional	Redirect to register-error with an error message detailing the criteria for the password	Redirect to register-error with an error message detailing the criteria for the password	N/A
85	Clicking the Register button after filling in all required fields	Functional	Redirects to registered, then back to login-success	Redirects to registered, then back to login-success	N/A

[Figure 185 - register-error unit tests]

register

Test No.	Test case	Test type	Expected result	Actual result	Action required
86	Attempting to press the register button/enter key without entering a valid email	Functional	The page correctly prompts for the field to be filled in if it is empty - and for the correct email format if the entered value does not contain the name@address format	The page correctly prompts for the field to be filled in if it is empty - and for the correct email format if the entered value does not contain the name@address format	N/A
87	Attempting to press the register button/enter key without entering a valid password	Functional	The page prompts for the password field to be filled in if it is empty.	The page prompts for the password field to be filled in if it is empty.	N/A
88	Attempting to enter a password that does not match the following criteria: Password must be at least 8 characters long and include at least one lowercase letter, one uppercase letter, and one digit.	Functional	Redirect to register-error with an error message detailing the criteria for the password	Redirect to register-error with an error message detailing the criteria for the password	N/A
89	Clicking the Register button after filling in all required fields	Functional	Redirects to registered, then back to login-success	Redirects to registered, then back to login-success	N/A

[Figure 186 - register unit tests]

report-bug

Test No.	Test case	Test type	Expected result	Actual result	Action required

90	Loading the page	Functional	HTML form displayed	HTML form displayed	N/A
91	Submitting the form without a name	Functional	Front end prompts to fill in a name	Front end prompts to fill in a name	N/A
92	Submitting the form without an email	Functional	Front end prompts to fill in an email	Front end prompts to fill in an email	N/A
93	Submitting the form without a bug title	Functional	Front end prompts to fill in a bug title	Front end prompts to fill in a bug title	N/A
94	Submitting the form without the bug description	Functional	Front end prompts to fill in the bug description	Front end prompts to fill in the bug description	N/A
95	Submitting the completed form	Functional	Sends an email to support@bookit.com - logs the bug to the back end	Throws “Error: Failed to lookup view "contact" in views directory”	The logic is incomplete - the address contact does not yet exist. Create the contact then fix the logic.

[Figure 187 - report-bug unit tests]

requests-list

Test No.	Test case	Test type	Expected result	Actual result	Action required
96	Loading the page	Functional	All bookings with the “Awaiting Approval” status appear	All bookings with the “Awaiting Approval” status appear - but bookings that are set for times in the past also appear.	Remove any requests from the list where the time has already passed.
97	Clicking on the “Review booking” button for a particular request	Functional	Redirects to the corresponding review-booking/ID page	Redirects to the corresponding review-booking/ID page	N/A
98	Clicking the “Order by Building” button	Functional	Rooms are ordered by their building with regards to how they are in the list of rooms	Rooms are ordered by their building with regards to how they are in the list of rooms (RHB first, WB second)	N/A

			(RHB first, WB second)		
99	Clicking the “Order by Upcoming Date/Time button	Functional	Bookings are ordered in ascending date/time according to the starting time	Bookings are ordered in ascending date/time according to the starting time	N/A
100	Clicking the “Order by Confirmed Risk Assessment” button	Functional	Bookings are ordered with the confirmed risk assessments first - then the oldest bookings first	Bookings are ordered with the confirmed risk assessments first - then the oldest bookings first	N/A

[Figure 188 - requests-list unit tests]

review-booking

Test No.	Test case	Test type	Expected result	Actual result	Action required
101	Loading the page	Functional	The email of the person that made the booking, timeslot, date, room number, status, seating requested and status of the risk assessment approval is displayed along with the risk assessment	The email of the person that made the booking, timeslot, date, room number, status, seating requested and status of the risk assessment approval is displayed along with the risk assessment - but the status of the approval of the risk assessment is always “Not yet reviewed” - even when it has been.	Fix the conditionals for displaying the status of the risk assessment approval so it shows the correct data.
102	Loading the page when reviewing a booking with a risk assessment that has been approved	Functional	Button options for the booking are “Back”, “Reject Booking” and “Approve	Button options for the booking are “Back”, “Reject Booking” and “Approve Booking”	N/A

			Booking"		
103	Loading the page when reviewing a booking with a risk assessment that has not yet been approved	Functional	Button options for the booking are "Back", "Reject Risk" and "Approve Risk"	Button options for the booking are "Back", "Reject Risk" and "Approve Risk"	N/A
104	Loading the page when reviewing a booking with a risk assessment that has been rejected	Functional	Button options for the booking are "Back", "Reject Booking" and "Approve Booking"	Button options for the booking are "Back", "Reject Booking" and "Approve Booking"	N/A
105	Clicking the "Back" button	Functional	Redirects back to the page from whence came	Always redirects back to the approved-list page	Change the way the link works to go backwards rather than to the approved-list page
106	Clicking on the "Approve Risk" button	Functional	Changes state of the booking to have an approved risk assessment	Changes state of the booking to have an approved risk assessment	N/A
107	Clicking on the "Reject Risk" button	Functional	Rejects the risk assessment - and changes the status of the booking to reflect this	Rejects the risk assessment - and changes the status of the booking to reflect this	The booking can still be approved once the risk assessment has been rejected - we could change this or leave it as is if that's functionality we wish to keep
108	Clicking on the "Approve Booking" button	Functional	Changes the status of the booking to "Approved"	Changes the status of the booking to "Approved" - redirects back to requests-list	Perhaps some sort of confirmation could help with user experience.
109	Clicking on the "Reject Booking" button	Functional	Changes the status of the	Changes the status of the booking to	Perhaps some sort of

			booking to “Approved”	“Approved” - redirects back to requests-list	confirmation could help with user experience.

[Figure 189 - review-booking unit tests]

rooms-list

Test No.	Test case	Test type	Expected result	Actual result	Action required
110	Loading the page	Functional	The booking buttons are replaced with “To make a booking for this room, select a date and time slot in the filter.”	The booking buttons are replaced with “To make a booking for this room, select a date and time slot in the filter.”	N/A
111	Applying a timeslot and date and time in the filters	Functional	Rooms that are already booked at that date and time are removed from the list - rooms that are in the list can now be booked with the new “Book” button	Rooms that are already booked at that date and time are removed from the list - rooms that are in the list can now be booked with the new “Book” button	N/A
112	Clicking the “Order by Building” button	Functional	Rooms are ordered by their building with regards to how they are in the list of rooms (RHB first, WB second)	Rooms are ordered by their building with regards to how they are in the list of rooms (RHB first, WB second)	N/A
113	Clicking the “Order by Capacity” button	Functional	Rooms are ordered from the largest seating capacity to the smallest.	Rooms are ordered from the largest seating capacity to the smallest.	N/A

114	Clicking the “Book” button after selecting a date and time slot	Functional	Redirected to the add-booking page	Redirected to the add-booking page	N/A

[Figure 190 - rooms-list unit tests]

view-booking

Test No.	Test case	Test type	Expected result	Actual result	Action required
115	Loading the page	Functional	The email of the person that made the booking, timeslot, date, room number, status, seating requested and status of the risk assessment approval is displayed along with the risk assessment	The email of the person that made the booking, timeslot, date, room number, status, seating requested and status of the risk assessment approval is displayed along with the risk assessment - but the status of the approval of the risk assessment is always “Not yet reviewed” - even when it has been.	Fix the conditionals for displaying the status of the risk assessment approval so it shows the correct data.
116	Loading the page on a booking that was not made by the currently logged in user - when not logged in as an admin	Functional	Button options should only include “Back”	Button options only include “Back”	N/A
117	Loading the page on a booking that was made by the currently logged in user - when not logged in as an admin	Functional	Button options should include “Back” and “Edit”	Button options include “Back” and “Edit”	N/A
118	Loading the page on a booking while logged in as an admin or coordinator	Functional	Button options should include “Back” and “Cancel” - also “Edit” if the	Button options include “Back” and “Cancel” - but not “Edit” if the booking was made by this	Include the Edit button in this scenario

			booking was made by this user	user	
119	Clicking the “Back” button	Functional	Takes you back one page	Takes you back one page	N/A
120	Clicking on the “Cancel” button	Functional	Redirects you to cancel-booking with a confirmation message that the booking was cancelled successfully	Redirects you to cancel-booking with a confirmation message that the booking was cancelled successfully	N/A
121	Clicking on the “Edit” button	Functional	Redirects to “edit-booking” for that particular booking	Redirects to “edit-booking” for that particular booking	N/A

[Figure 191 - view-booking unit tests]

Evaluation

Technical difficulty

Development

One of the greatest challenges for us was AJAX - for V0 we mainly used redirects and separate EJS routes to handle changing information and forms, but as the application grew we realised this was not fit for purpose. We had to research and learn how to implement AJAX for Bookit, which proved to be challenging. Not only because we had no experience with it as a group but also because we had to handle middleware, front end and AJAX JavaScript code all communicating and passing data between them. In the end - we created a viable solution that uses AJAX for loading new rooms and booking information for the end user while keeping load times to a minimum.

We attempted to follow the accessibility plan we put forward in our project proposal - making the buttons and text large enough to read for visually impaired users and ensuring that all colour/text combinations had AAA contrast ratings. We implemented this to the best of our ability using our custom CSS code - and despite some minor layout and spacing issues due to last minute layout changes, our CSS code and therefore our application is

accessible for both dyslexic and visually impaired users. We also made sure our application was responsive for all the common desktop screen sizes.

We also had to learn how to implement many different node modules that none of us had experience with. Including TinyMCE, Multer, Speakeasy and Bcrypt. This resulted in us learning many different embedding and data handling techniques - such as document structure and encryption/hashing. This was challenging - but due to the large size of our group we were able to allocate people to research one particular module or technique each, resulting in a good level of specialised knowledge in our group for different components of the application (e.g., Noah was in charge of email handling, Jake was in charge of encryption, etc.). As a result of how we approached this, we have produced a system that implements the critical node modules required for the functionality of Bookit.

Security testing was another large undertaking. Jake was the one in charge of planning and carrying out security tests; which ended up taking up a decent portion of his time throughout the project. Learning the process through OWASP was a challenge as we were not well-versed in security tests as a group. Furthermore, we experienced further challenges upon deployment as the Goldsmiths servers had their own security protocols that did not immediately align with the security measures we put in place when developing bookit locally - which required further amendment to get Bookit to a functional and deployed state. Overall - our system has the level of security that we intended originally - protecting against standard threats like interceptions, session hijacking and SQL injection.

Another challenge was working on a large GitHub repository, as many of our group members had never done it before. Thankfully - a couple of our group members were well-versed in GitHub and could help the entire group get acquainted. Furthermore, this was many of our first time using GitHub projects to handle organisation - which took some getting used to but was overall a good use of time as it improved productivity in the V1 development stage of the project.

We learned many techniques during the development of Bookit - both with organisation and planning, but also with application development and coding in general.

Planetscale

One of the more interesting challenges in the span of the Bookit project were the complications that were brought about by Planetscale. For the majority of the span of the project we were hosting our database on Planetscale - but they announced that they would be closing their free “hobby tier” - the tier we were using to get hosting - on April the 8th, which is before when our project submission was likely to be graded. Because of this, we had to migrate our database.

Jake managed to successfully migrate all of our data from planetscale to the Goldsmiths servers (Igor) and set up a separate connection to that version of the back end. This was in part thanks to the consistent updates we had been making to the create_db planetscale

version.sql file hosted on our GitHub repository, that allowed us to make a duplicate of our database and then insert the dumped data from Planetscale into it.

This proved to be an effective solution to the issues presented by the Planetscale Free-tier shutdown. The overall functionality of Bookit was not affected by the change.

Deployment difficulties

We experienced issues getting tinyMCE to work on the production servers, and we were unable to get this resolved within the project timescale. The root cause of the issue is that tinyMCE requires a html standards compliant document before it can run, and the production servers are stripping off the <!DOCTYPE from every page that they serve. We didn't have access to the configuration of the web servers on the University servers, to do any further analysis or resolve. The <!DOCTYPE tag is in the served pages in the development environment. This issue used up a lot of time that could have been spent progressing other issues. The lesson we learned from this is that we should have been pushing code to the production system at intervals during the project and not just at the end of the project. This would not have solved the issue, but it would have highlighted it earlier. The steps to reproduce this issue are:

- open this page Evidence for missing doctype (which is hosted on the production server
- View-source in your browser and check for <!DOCTYPE at the top of the source.
- If there is no <!DOCTYPE tag then the issue has been reproduced.

This is the source code of the page

Figure 192 - source code

Nodemailer

We envisioned a notifications system some time into the development which would send out email notifications ranging from booking confirmations to updates and cancellations of booking

Unfortunately with external issues with SMPT service providers, we had issues resolving the authentication of external app logins using the nodemailer middleware. While this feature

is user-friendly, this does not affect the functional aspects of the Bookit application, rather its user-friendly aspects.

Difficulties

When running the implementation testing node mailer we kept reproducing this error

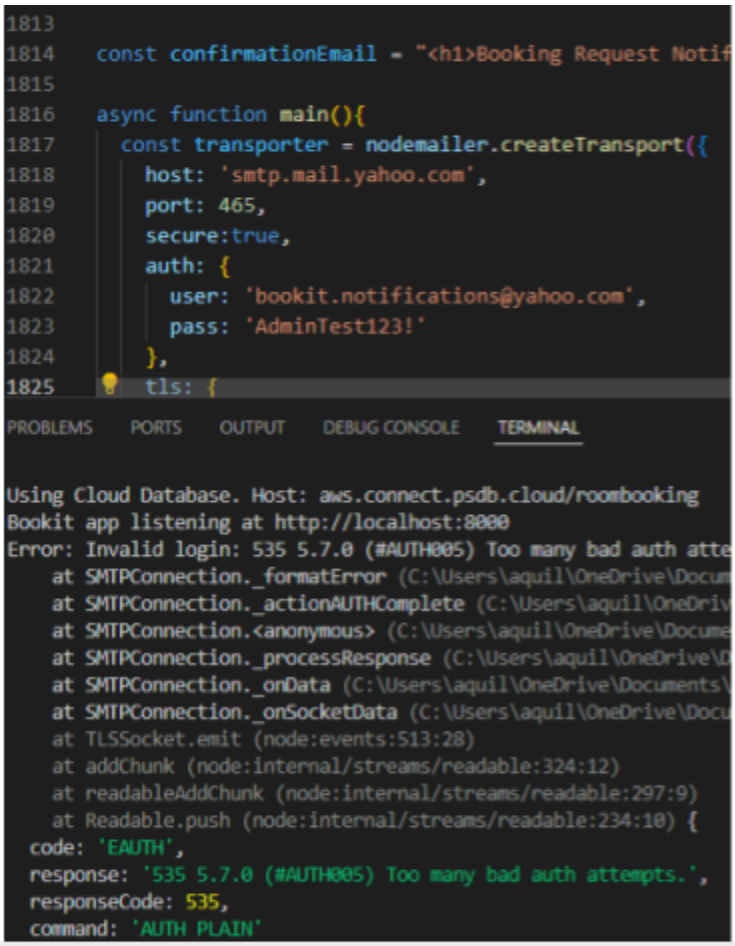
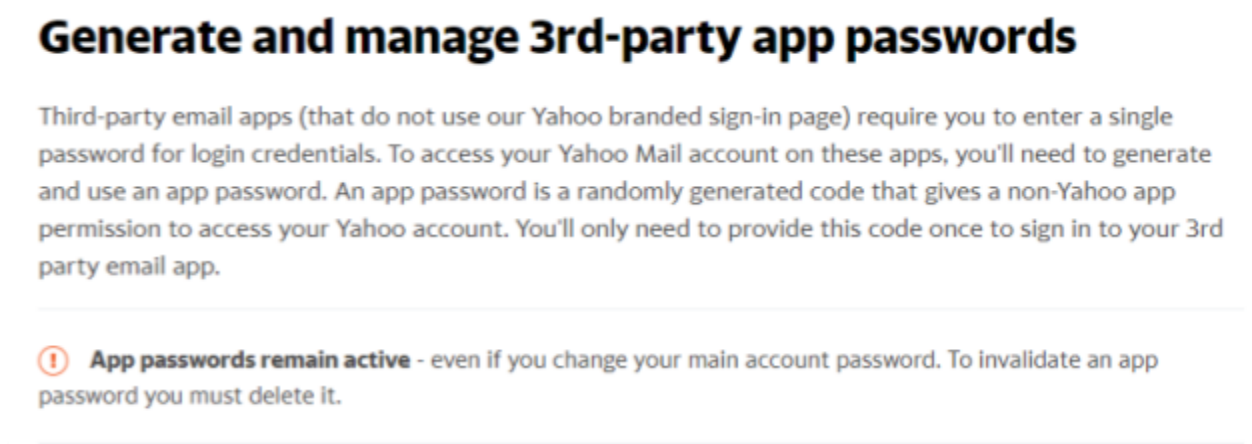


Figure 193 - source code

Noah made attempts through various email providers but ran into issues. Gmail would not allow him to authenticate his account and Microsoft SMTP services were unavailable at the time.

Promisingly, Yahoo mail has an allowance for generating external app passwords, and also uses the SMTP 456 port as industry standard and is secure.. This proved to be promising as a direct way of logging in.

However, after some investigation, Noah found numerous users unable to use the ‘generate app password’ feature without frequent use of the email account, prior to setting up (a timeframe we could not afford to rely on or measure).



Generate and manage 3rd-party app passwords

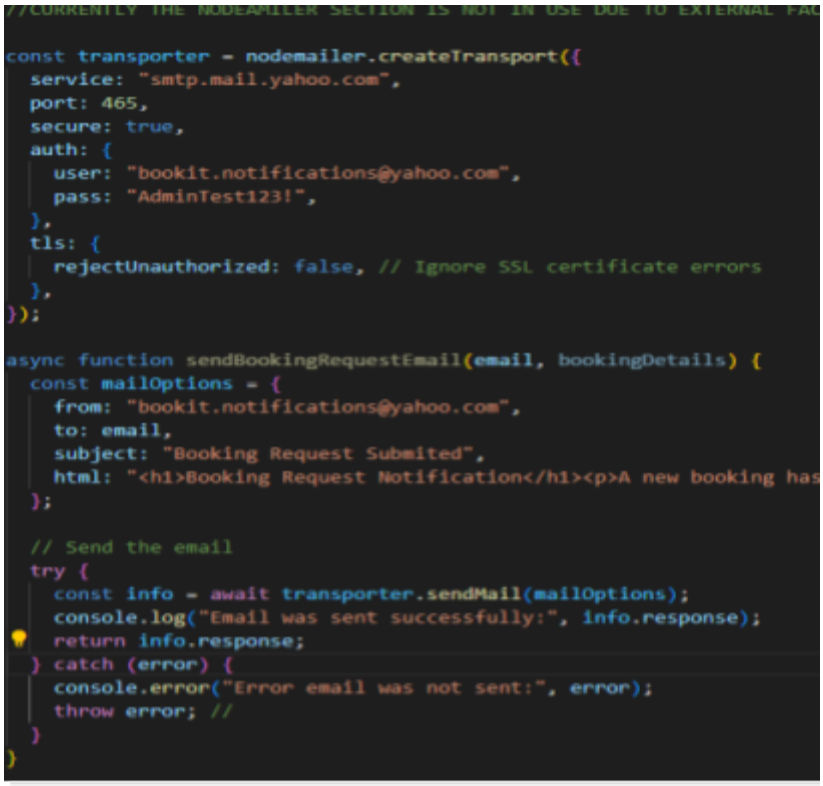
Third-party email apps (that do not use our Yahoo branded sign-in page) require you to enter a single password for login credentials. To access your Yahoo Mail account on these apps, you'll need to generate and use an app password. An app password is a randomly generated code that gives a non-Yahoo app permission to access your Yahoo account. You'll only need to provide this code once to sign in to your 3rd party email app.

App passwords remain active - even if you change your main account password. To invalidate an app password you must delete it.

Figure 194 - Yahoo 3rd party app information

https://help.yahoo.com/kb/SLN15241.html

This meant that unfortunately he was unable to get his most basic template working for the emailer. However, he structured to the same level, the functions and implementation of

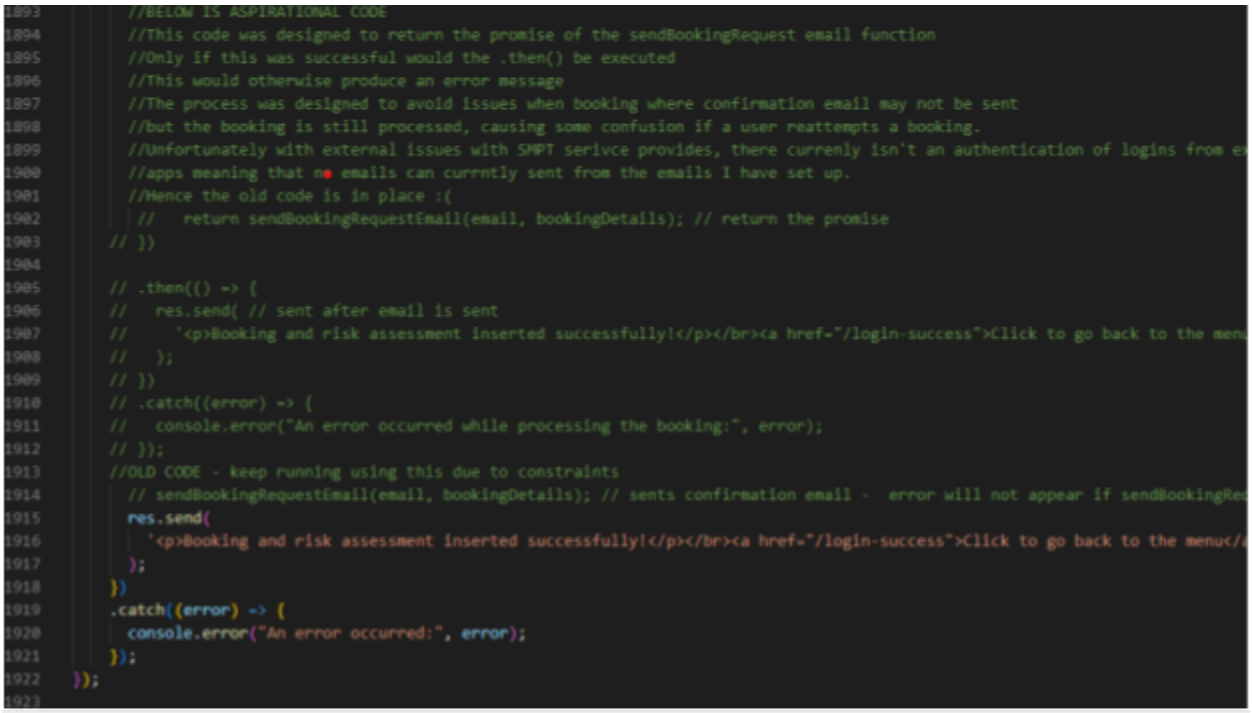


```
//CURRENTLY THE NODEMAILER SECTION IS NOT IN USE DUE TO EXTERNAL FAC

const transporter = nodemailer.createTransport({
  service: "smtp.mail.yahoo.com",
  port: 465,
  secure: true,
  auth: {
    user: "bookit.notifications@yahoo.com",
    pass: "AdminTest123!",
  },
  tls: {
    rejectUnauthorized: false, // Ignore SSL certificate errors
  },
});

async function sendBookingRequestEmail(email, bookingDetails) {
  const mailOptions = {
    from: "bookit.notifications@yahoo.com",
    to: email,
    subject: "Booking Request Submited",
    html: "<h1>Booking Request Notification</h1><p>A new booking has
  };

  // Send the email
  try {
    const info = await transporter.sendMail(mailOptions);
    console.log("Email was sent successfully:", info.response);
    return info.response;
  } catch (error) {
    console.error("Error email was not sent:", error);
    throw error; //
  }
}
```

Figure 195 - nodemailer middleware

the nodemailer middleware, into the code in the form of various functions taking information from the session data (see code snippet above).

This does however mean that all of this code is aspirational and while the principal development of this feature is complete, it will not be accessible and functional at the time of submission. Due to the scale of work however, it still exists - albeit commented out - in the main app.post("add-booking-submit") post response. See below



```
        //BELOW IS ASPIRATIONAL CODE
        //This code was designed to return the promise of the sendBookingRequest email function
        //Only if this was successful would the .then() be executed
        //This would otherwise produce an error message
        //The process was designed to avoid issues when booking where confirmation email may not be sent
        //but the booking is still processed, causing some confusion if a user reattempts a booking.
        //Unfortunately with external issues with SMPT serivce provides, there currenly isn't an authentication of logins from ex
        //apps meaning that no emails can currntly sent from the emails I have set up.
        //Hence the old code is in place :(
        //   return sendBookingRequestEmail(email, bookingDetails); // return the promise
      // })

      // .then(() => {
      //   res.send( // sent after email is sent
      //     '<p>Booking and risk assessment inserted successfully!</p></br><a href="/login-success">Click to go back to the menu
      //   );
      // })
      // .catch((error) => {
      //   console.error("An error occurred while processing the booking:", error);
      // });
      //OLD CODE - keep running using this due to constraints
        // sendBookingRequestEmail(email, bookingDetails); // sents confirmation email -  error will not appear if sendBookingRe
        res.send(
          '<p>Booking and risk assessment inserted successfully!</p></br><a href="/login-success">Click to go back to the menu</a
        );
      })
      .catch((error) => {
        console.error("An error occurred:", error);
      });
  });
```

Figure 196 - commented code

Technical novelty

We believe this project is novel enough to warrant its creation as it intended to improve upon a pre-existing process that we, as a group, deem to not be entirely fit for purpose.

Therefore, we designed Bookit with the "worst" aspects of the current room booking system in mind. As a result of this - we believe we have successfully delivered on our original proposed solution, and streamlined the room booking process. We believe that Bookit allows for an entirely central system for all aspects of the booking process, minimising back-and-forth between bookers and coordinators, as well as allowing for direct access to the room data for admins so it can be easily amended.

Overall, while Bookit is not "deployment ready" for an organisation the size of Goldsmiths in its current state, we believe that it is a sufficient prototype that is fit for a booking system for Goldsmiths. A future version would be more to the scale that is necessary for Goldsmiths.

Teamwork

Where could we improve with regards to our work as a team?

During the design and prototyping phases of the project the team worked well with individual tasks matching individual skills and areas of interest, for example within the group we had a society leader and individuals familiar with the existing booking process. We also had members who had more experience of parts of the design process than others. Bringing these skills together helped produce good design documentation for the later stages of the project. The group was quite large, and not all of the group members were available all the time (illness for example) but since there were always 3-4 members available we were able to complete the work.

Initially we used a combination of discord and weekly in-person meetings to communicate. This worked well through the design phases of the project. We chose weekly sprints, which were good for showing progress on smaller tasks, but less helpful for longer running tasks which took more than 1 week to complete. A two weekly sprint interval may have been better later on in the project.

Discord worked well for task allocation, informing each other of what we were working on, for immediate feedback and asking questions. This was more effective than having to arrange a meeting to discuss everything when we were mid sprint. Meeting up in person on Friday to discuss the project and make task allocations was also an important part of the teamwork process.

It is fair to say that at the end of the design phase there was more effort put in to get the report ready than in the time leading up to the deadline.

During the implementation phase we had a number of elements of the system that had dependencies on other elements. For example, we decided to put a lot of effort into getting the bookings-list and the filter working just the way we wanted, before we made the other “list” pages. We did this because even if all the other pages were written they would all need to be re-written when “bookings-list” was finished. Because of this approach we had “bottlenecks” in the project where one person’s work could not be completed until other work was done. This meant that there weren’t as many tasks to allocate early on in the project, and it was frustrating for the wider team, and this resulted in a lack of motivation. While the approach was efficient, in terms of only writing the code once, it didn’t help motivation and made it harder for the whole team to engage together, later on.

We switched from discord to github issues and a github project after the version 0 test results and this switch helped improve team engagement, and motivation. The graphs helped show what was completed and what was left to do, and showed the contributions visually which helped some team members.

We had good communication processes in place (even creating developer walkthroughs to explain processes), and the design phase went well, but we didn’t have the whole team

motivated during all of the implementation phase. We could have allocated tasks that we knew would probably need to have been re-done, later, but this felt preemptive in light of the efforts of our collaboration as a time and allowance for developmental growth as programmers. Perhaps if we'd started the implementation with github issues and a github project we could have ensured a clear scope and layout of the issues and task requirements through the earlier stages of our project..

But this is closely related to our philosophy of accommodation and scalability to experience when assigning tasks and group roles.

For instance Jake took on board the initial and crucial database setup and directed us towards using planetscale in the early phases of our setup. This is because Jake had the most comprehensive understanding of databases from our Dynamic Web Applications coursework and he felt the most confident to do so.

We could also point to Ben and how we would take on many bug fixing tasks in V1 and generally being a key motivator for the team, in line with earlier established dynamics from the proposal stage.

Another example is Shaquille being the most available to take on the duty of floor walking for photo collection and front end development. In fact, front end development was a task that a few members with less backend experience felt most confident working on because these are skills that have been most thoroughly tested on in coursework (especially because Bookit itself is a dynamic web application).

These are just a few examples and by no means are they a comprehensive list of group roles that evolved/scaled or the tasks undertaken by the group, but an insight into the way we allowed these tasks to be allocated in an evolving way throughout development..

Our approach was not perfect. We know this because there is a fine line between giving each other adjacency and allowing for complacency. We dont believe any member of our group was reluctant to work, but certainly during points of development some felt a sense of intimidation towards the tasks when these were the only tasks available. This was related to experience and orientation inside of code that was being developed quickly. In contrast to our aim, this meant there were times where our tenured developers were overtasked by comparison and this issue seemed to quickly expand exponentially.

We had contingencies however. As mentioned earlier in V1 we had lists of issues on Github.

We knew that some members who worked mostly on the front end earlier on, that github and bug fixing would quickly become a lifeline for these team members to segway into backend development.

In similar fashion this is why we looked to the use of 3rd party middleware for certain aspects of development. Using speakeasy allowed Noah to develop this feature without having to rely solely on a comprehensive mental map of the code base for instance.

In conclusion it is safe to say that we did not have a perfect approach to handling group roles and development. Our sentiments were to be accommodating, but realistically this was too broad and we definitely could have found ways of defining roles fairly and succinctly while avoiding stagnation in development.

Evolution from our project proposal

The evolution of our project proposal came from user feedback on version 0. The comments made were helpful. 4 issues raised led us to evolve the proposal.

User experience -List pages refresh results on change of fields and removal of confirm button (Issue 5)

This enhancement changes the user experience significantly and is also easier to use. Immediate feedback from alteration of the filter list provides the user a way of reducing large amounts of information quickly and in fewer steps. This does increase the number of sql database calls that the app makes, but it is not considered that these will be excessive, even in a multiuser environment.

User Experience - Dark Mode (Issue 13)

This was suggested through the v0 user testing process and quickly became the only option used by the development team. The development team might not be representative of all users, but having two options for the app colours will be a practical enhancement for the target users. Many of the society users will be familiar with dark mode themes being available in the applications that they use regularly.

User Workflow - Risk Assessment templating using in page text editor (Issue 35)

The risk assessment has been implemented during tinyMCE which is a “what you see is what you get” document editor embedded within the page. This approach allows the editing of a template html document, which can be pre-filled with the data from the booking, but also allow all of the updates required to complete a full risk assessment. The resulting document also is a record of the actual risk assessment, which if required can be produced later. If the risk assessment process changes, in the future, then it is a simple task to update the template which is held in views/risk-template.ejs, and doesn’t require any coding to update.

The other approach considered was to implement an html representation of the current risk assessment form. This would look better within the app - although it’s a very complex form, but would require significant coding changes if the risk assessment template ever required an update.

The route selected is practical and easy for existing users to understand and use. However for future enhancement we would consider implementing the form in html.

User Security - Multi-factor login (issue 7)

Implementation of multi-factor authentication using the google authenticator was a significant enhancement. For an application to be used within an organisation multi-factor authentication should be essential. Future changes to this would be to provide options for the multi-factor authentication supplier. Currently google authentication has been implemented. The University uses Microsoft Authenticator, and so for full deployment an additional implementation of Microsoft Authenticator, and single sign on should be considered. While the implemented multi factor significantly reduces risk of account compromise, implementing the same authentication as the University and single sign on would also mean that when an account is closed by University system administrators it would be automatically denied access to the bookit system.

Reflecting on our delivered system

Our original aim was to improve the room booking system that is currently in place at Goldsmiths. The booking system should allow for bookings to be made easily - to be made for a room that the booker wishes to use, to have risk assessments be quick and painless to access and complete, and to be accessible for coordinators and admins to approve and reject bookings as necessary - as well as amend the room and booking data as they deem fit.

We believe Bookit achieves this aim. Navigation is simple, and our testers were able to navigate the application with relative ease. We believe that almost all the features of the R4 MVP model have been delivered - as well as a few things that go outside of this scope - such as extra security measures and allowing admins to directly change the data on the back end through the application itself.

While the majority of the pages use a similar, simplistic layout which could be considered confusing - our testers had no problem discerning the pages from one another, and therefore we consider our layout to fulfil its purpose of being easily understandable and traversable. Our CSS is responsive within our defined scope of common desktop sizes - and we therefore believe that our CSS and front-end aesthetic choices are successful.

Splitting our focus between user testing and unit testing was a good decision. The user tests helped us gain insight into where we could improve Bookit from a user experience standpoint while the unit tests helped us get into the cracks of the system and iron out the bugs. Furthermore - this has given us a good starting point for the next version of Bookit to improve the system further.

We believe all of our basic functional requirements have been met by our system.

Going forward - future work for Bookit

Bookit is not perfect. Aside from the previously documented bugs (see Testing > Test Plan, Post Development), and the issues that are still logged in the backlog (See Appendix), there are a few things that Bookit is desperately missing.

Foremost, we did not complete the R4 MVP model. Bookit is still missing the notifications system. While we did attempt to implement one with Nodemailer, unfortunately the notifications system is not in the delivered system. For the next version of Bookit, this is our primary task. Furthermore, our TinyMCE system for templating the risk assessment works for submitting the adequate information regarding the risk assessment, but as mentioned previously, it is debatable as to whether the current document structure actually speeds up the risk assessment process. A dedicated HTML form, with dropdowns and checkboxes to select data (as detailed in the R4 MVP model), would be another primary task for the next version of Bookit.

There are also more aspirational features that were detailed previously in the “Prototyping > Iteration” section (see “Going forward”).

We believe our use of node modules was effective - but our code quality needs improving - as we have left in some commented code from features that were not complete at the time of submission.

On the subject of code quality - it is evident that our code has a lot of variation in the techniques and styles used. This is down to the code being written by many different developers without a standardised style or structure. For the future a task for us to standardise the code throughout Bookit to ensure readability and accessibility for future developers - as it stands right now the code, while functional, is very confusing to read for someone not familiar with the system.

Also, we still have some placeholder “confirmation” pages in the submitted prototype using res.send() commands. These show the user the result of the queries to the backend; like when a booking is successfully added. Unfortunately, this does not have any CSS or styling at all, and does not conform to the standards of the rest of the application. This issue is logged in the back log, but as it does not directly hinder the functionality of Bookit it got left behind during development. As the project stands right now, this task has become a priority for the next version of Bookit.

Lastly, we believe that a full and critical analysis of the accessibility considerations and execution is needed - as we did not survey the entire application to ensure that Bookit follows the proposed accessibility parameters put forward in our project proposal and our R4 MVP model. Therefore, while we made an effort to make Bookit accessible - the entire application does not conform to these standards and this must be corrected in the future.

Conclusion

While Bookit may not be an entirely comprehensive solution to our proposed problem - we believe it is a suitable prototype for our stakeholders. We believe that it is a definite improvement from the current booking system as it completely centralises the booking process and does not require so much back and forth communication. Thus speeding things up for educators and society leaders alike.

While for coordinators and admins the lack of a dedicated notification system may make things more difficult to find - the functionality for confirming and denying bookings is all there - meaning we are staying true to the “one application for all” vision as we apply this to the booking process.

Alongside the aforementioned missing features, a system to confirm you will actually be using a space before the booking takes place would help solve one of the problems identified in our project definition in that people can make large amounts of bulk bookings way in advance for spaces they may not end up using. For this system we will also need notifications to remind people to confirm their bookings - which is a task we will prioritise after the completion of the Nodemailer logic.

Overall, we believe that while our system lacks in some of the aforementioned regards, Bookit as it stands right now is an adequate prototype for a Booking system that streamlines and simplifies the current booking system that is in place here at Goldsmiths.

User Guide

Society leader/lecturers

How can I book a room?

After logging in, go to the "Book Room" section. Choose your date, time, and location from the filter on the left hand side. Select an available room and fill in the risk assessment - then confirm your booking. You will then have to wait for approval from a co-ordinator.

How can I view current bookings?

After logging in, go to the “View Bookings” page. This will show you all outstanding bookings.

How can I edit my booking?

After logging in, go to the “View Bookings” page. This will show you all outstanding bookings. Navigate to a booking you’ve made (it will have an additional option to “edit”). EIther click the “edit” button on your booking card or go to view > edit at the bottom of the page. Update your risk assessment and then click the “Update booking” button at the base of the page.

Credentials

Email: jake@123.com
Password: test

Coordinators

Coordinators have access to the above functionality alongside the following (they can also cancel any booking - not just ones made by themselves):

How can I view booking requests?

After logging in, go to the “View requests” page. This will show you all unapproved bookings.

How can I review risk assessments?

From the “View requests” page; click on “Order by confirmed risk assessment” - then click on the “review booking” button on a booking card labelled “Risk Assessment: Not reviewed”. At the base of the page there will be options to either approve or reject the risk assessment.

How can I review bookings?

Follow the same steps as for reviewing risk assessments but for a booking card labelled as “Risk Assessment: Approved” or “Risk Assessment: Rejected” - the options for approving or rejecting the risk assessment will be replaced with options to approve or reject the booking overall.

How can I view all approved bookings?

After logging in, click the “View approved” button and this page is all approved bookings.

How can I cancel a booking?

When reviewing a booking - there will be a cancel button at the bottom of the page - click this button to remove the booking.

Credentials

Email: coordinator@123.com
Password: test

Admins

Admins have access to all the above functionality along with the following:

How can I register a new Bookit user?

After logging in - go to the “Register user” page - fill in the form with the user details and click the “Register” button.

How can I add a new room?

After logging in, go to the “Add room” page - fill in the form with the room data, upload a room image and click the “Add room” button.

How can I edit room data?

After logging in, go to the “Edit rooms” page - from here you can either click the “delete” button on a room card to remove that room - or click the “edit” button on the room card - then amend the form and click the “save” button to finalise your changes.

Credentials

Email: admin@123.com
Password: admintest

Miscellaneous

How can I report a bug?

Go to the “report bug” page in the footer and fill out and submit the form.

How can I change the theme?

Click the tick box that says “dark mode” in the header to switch back and forth from dark mode.

How can I email support?

Click the “support email” button in the footer to email support@bookit.com.

How do I log out?

Click the “menu” button in the header - then click “logout”.

Reference list/bibliography

1. "About OWASP - OWASP Top 10:2021." *Owasp.org*, owasp.org/Top10/A00-about-owasp/.
2. Andrei, Bogdan-Alexandru. "A STUDY on USING WATERFALL and AGILE METHODS in SOFTWARE PROJECT MANAGEMENT." 2019.
3. Bach, Cedric, and Dominique L. Scapin. "Comparing Inspections and User Testing for the Evaluation of Virtual Environments." *International Journal of Human-Computer Interaction*, vol. 26, no. 8, 30 July 2010, pp. 786–824, https://doi.org/10.1080/10447318.2010.487195.
4. Balaji, S, and M Sundararajan. "International Journal of Information Technology and Business Management WATEERFALLVs V-MODEL vs AGILE: A COMPARATIVE STUDY on SDLC." *International Journal of Information Technology and Business Management*, vol. 2, no. 1, 29 June 2012, mediaweb.saintleo.edu/Courses/COM430/M2Readings/WATEERFALLVs%20V-MODEL%20Vs%20AGILE%20A%20COMPARATIVE%20STUDY%20ON%20SDLC.pdf.
5. British Dyslexia Association. "Dyslexia Friendly Style Guide." *British Dyslexia Association*, 2023, www.bdadyslexia.org.uk/advice/employers/creating-a-dyslexia-friendly-workplace/dyslexia-friendly-style-guide.

6. Cadle, James, et al. *BUSINESS ANALYSIS TECHNIQUES 72 Essential Tools for Success*. 2014.
7. Devaraj, Kiruthika. “Common Screen Resolutions | What Are They & How to Test?” *Testsigma Blog*, 25 Oct. 2023, testsigma.com/blog/common-screen-resolutions/.
8. “Developing Booker with the University of Cambridge.” *Www.eventmapsolutions.com*, www.eventmapsolutions.com/success-stories/creating-booker-with-the-university-of-cambridge.
9. Dragos, Paul. “THE IMPACT of STAKEHOLDERS in AGILE SOFTWARE DEVELOPMEN.” *THE ANNALS of the UNIVERSITY of ORADEA. ECONOMIC SCIENCES*, vol. 30, no. 2, Dec. 2021, pp. 353–362, https://doi.org/10.47535/1991auoes30(2)037.
10. Dustin, Elfriede, et al. *Implementing Automated Software Testing*. Pearson Education, 4 Mar. 2009.
11. Huck-Fries, Veronika, et al. “Investigating the Role of Stakeholders in Agile Information Systems Development Projects: A Mixed Methods Approach.” *Hawaii International Conference on System Sciences 2021 (HICSS-54)*, 4 Jan. 2021, aisel.aisnet.org/hicss-54/st/agile_development/6/.
12. Khan, Sikandar. “CSRF Tokens in ExpressJS — Node.js Web Framework.” *Medium*, 27 Mar. 2022,

dearsikandarkhan.medium.com/csrf-tokens-in-expressjs-node-js-web-framework-cc331069de2d.

13. “Login - Booker.” *Booker.eventmapsolutions.com*, booker.eventmapsolutions.com/.
14. Marasinghe, Lanil. “How to Secure Your NodeJS Web Application Using Synchronizer Tokens.” *Medium*, 26 Oct. 2018, medium.com/@lanil.marasinghe/how-to-secure-your-nodejs-web-application-using-synchronizer-tokens-959c45200876.
15. Miecznik, Rafał. “The Importance and Benefits of Unit Testing.” *CodiLime*, 22 June 2023, codilime.com/blog/unit-testing/.
16. “Most Popular Web Browsers in 2021 [Jul ’21 Update] | Oberlo.” *Www.oberlo.com*, 2023, www.oberlo.com/statistics/browser-market-share.
17. Nguyen, Thao. *IMPROVING STUDENTS’ UX in ONLINE LEARNING PLATFORM Case Study: LAB Faculty of Business and Hospitality Management*. 2020.
18. OWASP. “Cross Site Request Forgery (CSRF) | OWASP.” *Owasp.org*, 2023, owasp.org/www-community/attacks/csrf.
19. ---. “OWASP ZAP.” *Zaproxy.org*, 2020, www.zaproxy.org/.
20. Park, Sangwon, et al. “Understanding of Online Hotel Booking Process: A Multiple Method Approach.” *Journal of Vacation Marketing*, vol. 25, no. 3, 25 June 2018, pp. 334–348, https://doi.org/10.1177/1356766718778879.

21. Perera, K. M. P. B. N. “LECTURE HALL SCHEDULING and TIMETABLE MANAGEMENT SYSTEM.” *Dl.ucsc.cmb.ac.lk*, 4 Aug. 2021, dl.ucsc.cmb.ac.lk/jspui/handle/123456789/4407.
22. “Room Booking System - Online Room Booking Software.” *Roombookingsystem.co.uk*, roombookingsystem.co.uk/.
23. “SchoolBooking - the Education Booking System.” *Www.schoolbooking.com*, schoolbooking.com.
24. “Software Security | HTML5: Overly Permissive CORS Policy.” *Vulncat.fortify.com*, vulncat.fortify.com/en/detail?id=desc.config.dotnet.html5_overly_permissive_cors_policy.
25. Spindler, Gerald, and Philipp Schmechel. “Personal Data and Encryption in the European General Data Protection Regulation.” *Journal of Intellectual Property, Information Technology and Electronic Commerce Law*, vol. 7, 2016, p. 163, heinonline.org/HOL/LandingPage?handle=hein.journals/jipitec7&div=18&id=&page=.
26. Steven, M, et al. *Parallel Worlds: Agile and Waterfall Differences and Similarities*. 2013.
27. Sun, Yunchuan, et al. “Data Security and Privacy in Cloud Computing.” *International Journal of Distributed Sensor Networks*, vol. 10, no. 7, Jan. 2019, p.

190903. *Sagepub*, journals.sagepub.com/doi/full/10.1155/2014/190903, https://doi.org/10.1155/2014/190903.

28. Toonen, Edwin. "The Carbon Footprint of Your Website and How to Reduce It." *Yoast*, 19 Apr. 2023, yoast.com/carbon-footprint-of-website/.
29. Totaljobs. "Jobs | UK Job Search | Find Your Perfect Job - Totaljobs." *Totaljobs.com*, 2020, www.totaljobs.com/.
30. Um, Seoho, and John L. Crompton. "The Roles of Perceived Inhibitors and Facilitators in Pleasure Travel Destination Decisions." *Journal of Travel Research*, vol. 30, no. 3, Jan. 1992, pp. 18–25, https://doi.org/10.1177/004728759203000303.
31. "Web Application Scanning | Qualys, Inc." *Www.qualys.com*, www.qualys.com/apps/web-app-scanning/.
32. "QR Code Best Practices: Tips & Tricks To Create QR Codes That Engage And Convert" - Amanda Cox February 18, 2023, https://tritonstore.com.au/qr-code-best-practices/#:~:text=The%20size%20of%20a%20QR,be%20readable%20and%20scanned%20successfully.

Appendix

Group Logs and task assignments:

Sprints:

Week 3 w/b 17/01/2024

Multi week sprint task. Shaquille: Obtain data for the rooms, get pictures of rooms, the names of rooms. The pictures should be renamed to be the name of the room, so they can be linked later, the pictures go in the media folder. Need at least enough rooms so the list of rooms is big enough to fill the screen. This must be entered into the system using the add room page.

GROUP ID: Kelvin's Kittens			
Aim – Tasks you aim to achieve this week		Who is responsible? Names	Did you achieve this aim?
1	Research online mysql databases and get databases created. Prove database connects with example node.js code. Insert test data via mysql command line Create login mechanisms, and sessions, with hashed passwords stored in the database, and also create a register page to create new users. Set up GitHub Create stylesheet in line with high definition wireframes to act as a template for other team members work, add menu for coordinator and society leader Track progress on tasks by members of the group Update the admin role Stop people accessing the register page if they're not logged in as admin	Jake	☹ 😕 😐 🙂 🙂

	Fix issues with module not being saved when new user registered Put in the readme how to create a new user Add temporary logo Make some changes to the styling for addroom and the blue buttons as well as make the “view” button smaller Implement one of the rooms data with a photo from shaq’s data		
2	Create an admin login success menu - change the profile photo to green - should be identical to the coordinator login menu but also include options for registering user and adding a room Do custom CSS formatting for the bookings list page - use a “card format” to give things a more boxy aesthetic.	Ben	☹ 😕 😐 🙂 😀
3	HTML & CSS filters. Those are the sliders for duration and seating availability. Input boxes for timeslot, building and room type as well as the calendar to pick the date of the room you wish to book	Kelvin	☹ 😕 😐 😀 🙂
4	Complete an add room page, which is connected to the database so data can be added	Sahaf	☹ 😕 😐 🙂 😀
5	Css: - Font sizing and styling in line with dyslexia requirements - Editing flex spacing across pages	Noah/Spike	☹ 😕 😐 😀 🙂

	Work on Calendar for the filter section		
GROUP HEALTH:			
	Group Member	Happiness	Contribution
1	Jake	☹ 😕 😐 🙂 ☺	☹ 😕 😐 🙂 ☺
2	Ben	☹ 😕 😐 🙂 ☺	☹ 😕 😐 😀 🙂
3	Kelvin	☹ 😕 😐 🙂 ☺	☹ 😕 😐 😀 🙂
4	Sahaf	☹ 😕 😐 🙂 ☺	☹ 😕 😐 😀 🙂
5	Noah	☹ 😕 😐 😀 🙂	☹ 😕 😐 😀 🙂
6	Spike	☹ 😕 😐 😀 🙂	☹ 😕 😐 😀 🙂

Week 4 w/b 26/01/2024

GROUP ID: Kelvin's Kittens			
	Aim – Tasks you aim to achieve this week	Who is responsible? Names	Did you achieve this aim?
1	Adding room_type and building_name to the room database as they were missing in the original design. Add lookup_room_type and the initial data Add a developer walkthrough video for our group	Jake	☹ 😕 😐 🙂 ☺

	E1 (Jake) ✅& E2 (Jake) ✅Security testing with fixing some of the risks if found Update of the project report		
2	A1 (Ben) ✅filter getting the dropdowns to work means showing time slots that aren't booked (for book room) A2 (Ben) ✅- order list by buttons at top of right column. A3 50% - connecting the filter and data to show the list Add an “Order by building” button to the bookings list page and make it group the list by the buildings that contain the rooms. Add an “Order by status” button to the bookings list page and make it group the list by the bookings’ status. Add an “Order by time” button to the bookings list page and list the bookings by the earliest upcoming booking first.	Ben	☹ 😕 😐 🙂 ☺
3	A4 (Sahaf) ✅ AJAX for the filter researched	Sahaf	☹ 😕 😐 🙂 ☺
4	B1 (Shaq) ✅ room pictures taken and room data gathered	Shaq	☹ 😕 😐 🙂 ☺
5	D1 (Kelvin) ✅ Book room (version of view bookings - Requires view bookings to be finished) Needs re-doing when A is completed. D2 (Kelvin) ✅ View-requests (version of view bookings - Requires view bookings	Kelvin	☹ 😕 😐 🙂 ☺

	to be finished. Needs re-doing when A is completed D3 (Kelvin) ✅ View accepted (version of view bookings - Requires view bookings to be finished. Needs re-doing when A is completed		
6	C1 (Noah, Spike) ✅ { View booking details (look at the details of a booking) C2 (Noah, Spike) ✅ { Edit booking (version of view booking details - Requires view booking details to be finished) } C3 (Noah, Spike) ✅ { View room (version of view booking details - Requires view booking details to be finished) }	Noah & Spike	☹ 😕 😐 🙂 😀

GROUP HEALTH:

	Group Member	Happiness	Contribution
1	Jake	☹ 😕 😐 🙂 😀	☹ 😕 😐 🙂 😀
2	Ben	☹ 😕 😐 🙂 😀	☹ 😕 😐 🙂 😀
3	Kelvin	☹ 😕 😐 🙂 😀	☹ 😕 😐 🙂 😀
4	Shaq	☹ 😕 😐 🙂 😀	☹ 😕 😐 🙂 😀
5	Spike	☹ 😕 😐 🙂 😀	☹ 😕 😐 🙂 😀
6	Noah	☹ 😕 😐 🙂 😀	☹ 😕 😐 🙂 😀

Week 5 w/b 02/02/2024

GROUP ID: Kelvin's Kittens			
Aim – Tasks you aim to achieve this week		Who is responsible? Names	Did you achieve this aim?
1	Put the header and footer in include files so it's easier to manage and change later on Add a menu button to go back to the login-succes menu Create icons and upload them with implementation to display them Make it so all header files work in all situations by passing in "loggedInMessage" Security testing with fixing some of the risks found	Jake	☹ 😕 😐 🙂 🙂
2	Connecting the filter data to the back end Redirect from the confirmed filter to a new "filtered list". Fix the front-end JS that was stopping the sliders from working	Ben	☹ 😕 😐 🙂 🙂
3	Adding room and its data into MySQL	Shaq	☹ 😕 😐 🙂 🙂
4	A5 - dynamic filtering that updates the displayed list of items in real-time based on user input	Sahaf	☹ 😕 😐 🙂 🙂
5.	C4 connect view booking, view accepted, view room, and edit booking to the database	Noah/Spike	☹ 😕 😐 🙂 🙂

GROUP HEALTH:			
Group Member		Happiness	Contribution
1	Jake	☹ 😕 😐 🙂 😃	☹ 😕 😐 🙂 😃
2	Ben	☹ 😕 😐 🙂 😃	☹ 😕 😐 😃 🙂
3	Shaq	☹ 😕 😐 🙂 😃	☹ 😕 😐 😃 🙂
4	Sahaf	☹ 😕 😐 😃 🙂	☹ 😕 😐 😃 🙂
5.	Noah	☹ 😕 😐 🙂 🙂	☹ 😕 😐 🙂 🙂
6.	Spike	☹ 😕 😐 🙂 🙂	☹ 😕 😐 🙂 🙂

Week 6 w/b 09/02/2024

The other "list" pages will follow the template of the bookings-list, so I think the priority is getting the bookings list working the way we want. Making different versions will be simpler copy, edit, after we are done with bookings-list.

Now that the bookings list has real data and the bookings dataset contains (bookingId & roomId) , more of the Part C and Part D pages can be completed.

GROUP ID: Kelvin's Kittens			
Aim – Tasks you aim to achieve this week		Who is responsible? Names	Did you achieve this aim?
1	Add getRoomTypes and getRoomName functions to get the select boxes to only show the RoomTypes and building Names that the logged in person has got bookings for. Add test booking data Update app.js to get the bookings from the database Alter getBookings to match getRoomTypes and getBuildingNames Mapping the database fields to the existing names used in bookings in the test data. Alter bookings-list to use the picture_URL that is coming from the database to display the picture against each booking. Make sure database data is formatted the way the filter needs it Add an animation to the bookings-list so when the data is changed so the user can see something has been updated Prototype rooms-list.ejs Complete all filters by taking the date	Jake	☹ 😕 😐 🙂 🙂

	and time slot parameters from the filter calculate duration and if there is a date and a timeslot the main database query gets rooms that don't have an existing booking in or overlapping the timeslot we have in the filter. if no date OR time slot then all rooms are retrieved. Update the button to show what time the user is booking for Security testing with fixing some of the risks if found Update of project report.		
2	Fix issue T1: - T1 ❌ Seating filter - works for "1" in the filter but for any higher number it doesn't work. Fix issue T2: - Order by buttons reset the data to the original bookings list data (so if you put a filter on, then you order the data - the filters are removed and all data is shown)	Ben	☹ 😕 😐 🙂 🙂
3	Create Credits Page	Shaq	☹ 😕 😐 😀 🙂
4	A5 - dynamic filtering that updates the displayed list of items in real-time based on user input	Sahaf	☹ 😕 😐 😀 🙂
5	C4 - reattempt to connect view booking, view accepted, view room, and edit booking to the database	Noah / Spike	☹ 😕 😐 🙂 🙂

GROUP HEALTH:

	Group Member	Happiness	Contribution
1	Jake	☹ 😕 😐 🙂 🙂	☹ 😕 😐 🙂 🙂

2	Ben		
3	Shaq		
4	Sahaf		
5.	Noah		
6	Spike		

Week 7 w/b 16/02/2024

GROUP ID: Kelvin's Kittens			
Aim – Tasks you aim to achieve this week		Who is responsible? Names	Did you achieve this aim?
1	Complete room-list.ejs and addbooking.ejs Security testing with fixing some of the risks if found	Jake	☹ 😕 😐 🙂 😀
2	Fix the CSS for the addbooking.ejs page. Add a confirmation screen to the login-success.ejs page for when a booking is successfully added	Ben	☹ 😕 😐 🙂 😀
3	Research methods of implementing 2FA into the login process - middleware - usability	Noah / Spike	☹ 😕 😐 🙂 😀
4	D4 Edit room (version of add room - Requires add room to be finished	Kelvin	☹ 😕 😐 🙂 😀
5	A5 ✅- dynamic filtering that updates the displayed list of items in real-time based on user input	Sahaf	☹ 😕 😐 🙂 😀
GROUP HEALTH:			
Group Member		Happiness	Contribution
1	Jake	☹ 😕 😐 🙂 😀	☹ 😕 😐 🙂 😀
2	Ben	☹ 😕 😐 🙂 😀	☹ 😕 😐 😄 🙂
3	Sahaf	☹ 😕 😐 😄 🙂	☹ 😕 😐 😄 🙂

4	Kelvin		
5	Spike		
6	Noah		

Week 8 w/b 23/02/2024

GROUP ID: Kelvin's Kittens			
Aim – Tasks you aim to achieve this week		Who is responsible? Names	Did you achieve this aim?
1	Create a project in github to manage v1 issues. Security testing. Update of documentation for v0 user testing feedback. Complete whatever is left over from Parts C and D if these are not complete, by current assignees, by Tuesday. Create requests-list. Create approved-list Create edit-room-list Edit-room changes and completion Display risks in view-booking Create Review-booking Edit-booking changes & completion View booking back button so can be used in approved-list as well as booking-list	Jake	☹ 😕 😐 🙂 🙂
2	Add all headings to a document - create the basic structure for the project report. Complete the introduction and planning sections. Create an updated GANTT chart for the process we have been following.	Ben	☹ 😕 😐 🙂 🙂
3	- Learn how to use Speakeasy - Prototype first version of 2Fa example pseudocode - Prototype implementation in codebase	Noah	☹ 😕 😐 🙂 🙂
4	Create FAQ Page	Shaq	☹ 😕 😐 🙂 🙂

GROUP HEALTH:			
Group Member		Happiness	Contribution
1	Jake	☹ 😕 😐 🙂 😀	☹ 😕 😐 🙂 😀
2	Ben	☹ 😕 😐 🙂 😀	☹ 😕 😐 😀 🙂
3	Noah	☹ 😕 😐 😀 🙂	☹ 😕 😐 😀 🙂
4	Shaq	☹ 😕 😐 😀 🙂	☹ 😕 😐 😀 🙂
5	Sahaf	☹ 😕 😐 🙂 🙂	☹ 😕 😐 🙂 🙂

Week 9 w/b 01/03/2024

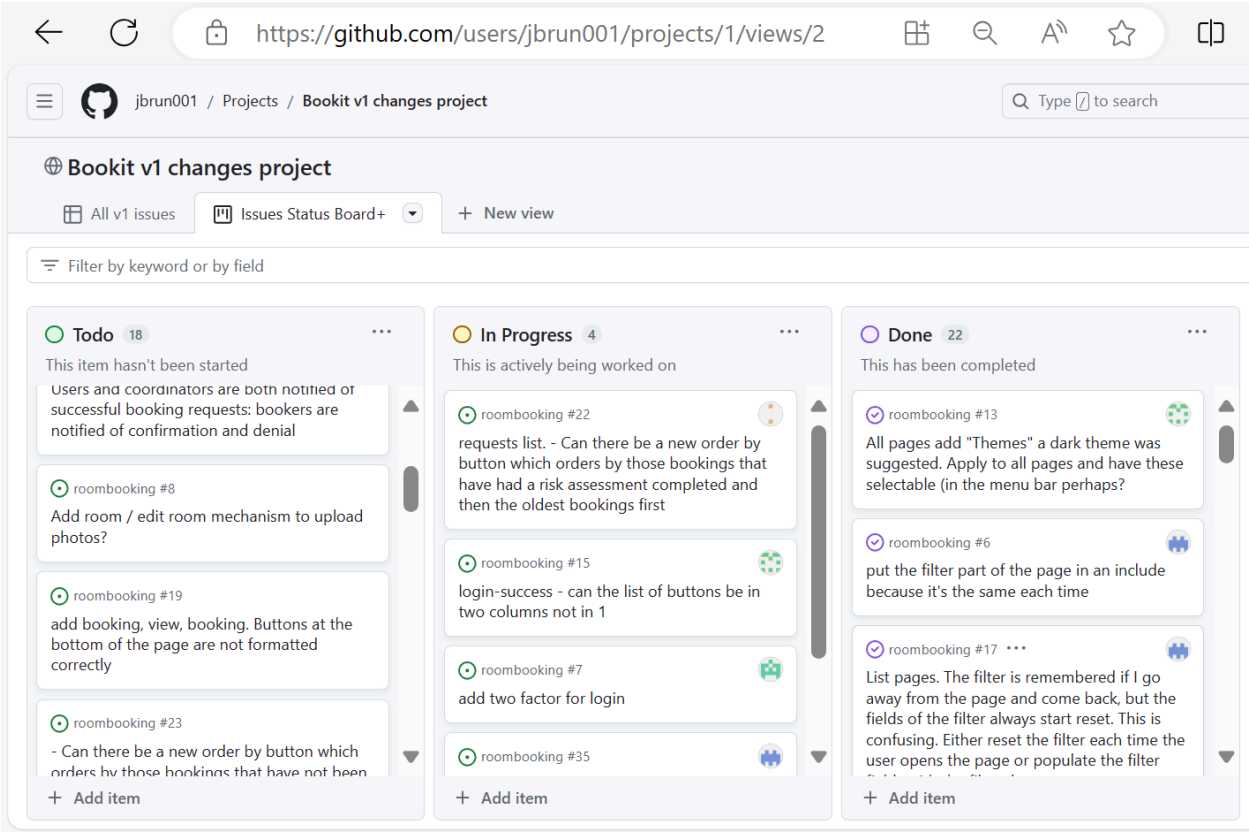
Issues from version 0 have now been added to the backlog, by using github issues Issues · jbrun001/roombooking (github.com) and a project within github. The task allocations have been moved from discord messaging to github for this part of the project.

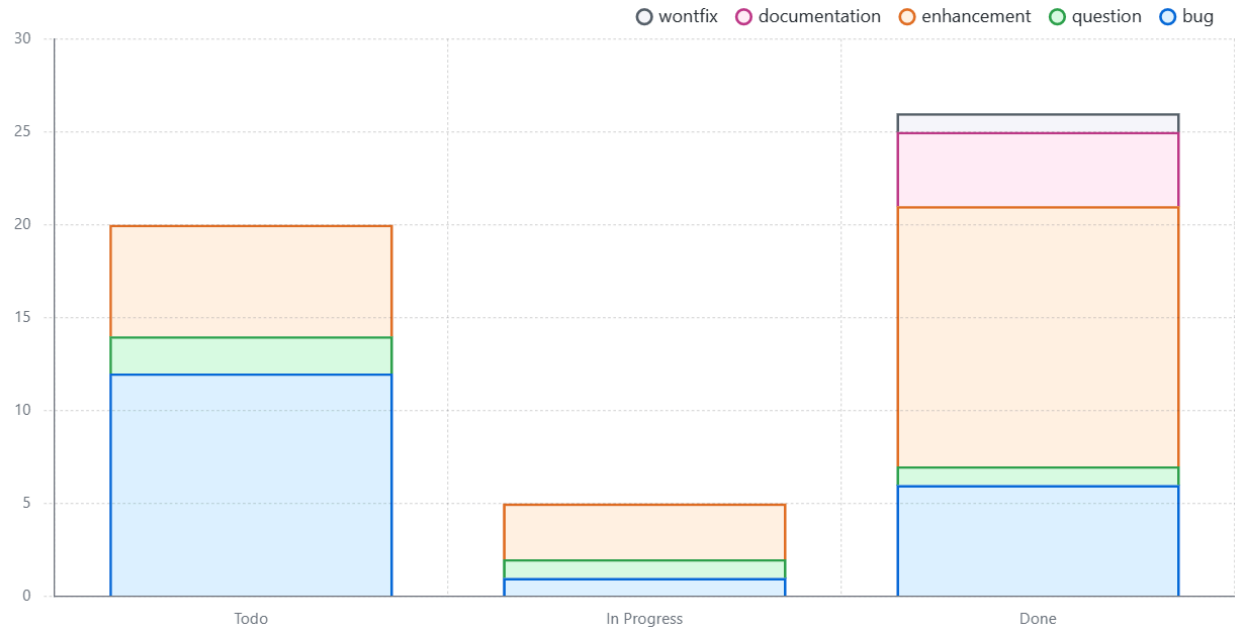
Discord was good in the early stages where larger tasks could be allocated to smaller teams who could work independently and at the end of the work collaborate with other smaller teams to integrate the work.

A board has been set up to show which issues are being worked on at any time by the group (see below). At this stage of the project this mechanism is better for managing sprints and workload, because there are many small issues and only a small number of issues have dependencies on other issues. Allocation of work is easier because team members can self allocate from the backlog of issues, and issues can be raised during the testing process as the testing is complete rather than wait until the end. This means there is a backlog of items that team members can allocate based on their own skills or areas they wish to develop.

When an issue is completed it is marked as “done” and a field was created for the date the work was completed. Commits to github have been tagged with the issue number, so the commit comments appear in the issue history. Issues are closed when the work has been reviewed. Comments and questions can be added to an item by any member of the team - for example to explain what options they think there are for dealing with the issue, or for giving feedback on the fix.

The current project issues board is here Issues Status Board+ · Bookit v1 changes project (github.com), and this is a screenshot at the end of week 9, and progress chart by category (Bookit Live - v1 changes progress by category). If there are no more issues raised we are currently about 50% of the way through the changes identified from v0.


https://github.com/users/jbrun001/projects/1/views/2
jbrun001 / Projects / Bookit v1 changes project
Type / to search
Bookit v1 changes project
All v1 issues
Issues Status Board+
New view
Filter by keyword or by field
Todo 18
This item hasn't been started
Users and coordinators are both notified of successful booking requests: bookers are notified of confirmation and denial
roombooking #8
Add room / edit room mechanism to upload photos?
roombooking #19
add booking, view, booking. Buttons at the bottom of the page are not formatted correctly
roombooking #23
- Can there be a new order by button which
Add item
In Progress 4
This is actively being worked on
roombooking #22
requests list. - Can there be a new order by button which orders by those bookings that have had a risk assessment completed and then the oldest bookings first
roombooking #15
login-success - can the list of buttons be in two columns not in 1
roombooking #7
add two factor for login
roombooking #35
Add item
Done 22
This has been completed
roombooking #13
All pages add "Themes" a dark theme was suggested. Apply to all pages and have these selectable (in the menu bar perhaps?
roombooking #6
put the filter part of the page in an include because it's the same each time
roombooking #17
List pages. The filter is remembered if I go away from the page and come back, but the fields of the filter always start reset. This is confusing. Either reset the filter each time the user opens the page or populate the filter
Add item


wontfix
documentation
enhancement
question
bug
30
25
20
15
10
5
0
Todo
In Progress
Done

GROUP ID: Kelvin's Kittens			
Aim – Tasks you aim to achieve this week		Who is responsible? Names	Did you achieve this aim?
1	Backlog issues relating to the filter: - On field change refresh data list (#5), standardise code (#4), move to an include (#6), requests list filter problem fix (#14), fix filter remembered between pages issue (#17). Backlog issues relating to display of risk assessment status (#11,#21) Backlog issues relating to getBookings and getRooms fixing not using parameterised SQL (#3) Full security testing following release of v0 in line with Security Testing Strategy. Proof of concept work for issue 35 - risk templates created tinyMCE example.	Jake	☹ ☹ 😐 ☺ 🙂
2	Add restriction for dates in the past when booking rooms (issue #2) Add a button for the most recently approved bookings and ordering by the last booking to be approved first on the list pages (issue #25) Rename the “Order by time” button to make it more clear as to what it actually does. Also fix the CSS for centering the buttons (issue #24) Add a character counter for the risk assessment page - limit both boxes to only 200 characters and show the user how close they are with HTML (issue #1) Add a reset filters button to reset the list page to its default state. Make the list images all the same size on the bookings page - they should grow and shrink with the window (issue #35)	Ben	☹ ☹ 😐 ☺ 🙂

	Bookings should be cancellable. Add a cancel button to the view-booking page. Bookings-list should not include cancelled bookings. (issue #33) Fix the issue where you can book the same room at the EXACT same time as someone else		
3	Implement 2FA by expanding on prototype: - Use the pseudocode you made - Watch for error handling when extending login submit - Create login 2Fa page - Test - Log for the report (add later) -	Noah	
4	Create Reports Bug Page + Make it functional	Shaq	
5	Add a Dark Theme #13 Styling Credits page #28 Styling FAQ page #30	Sahaf	

GROUP HEALTH:

	Group Member	Happiness	Contribution
1	Jake		
2	Ben		
3	Noah		
4	Shaq		

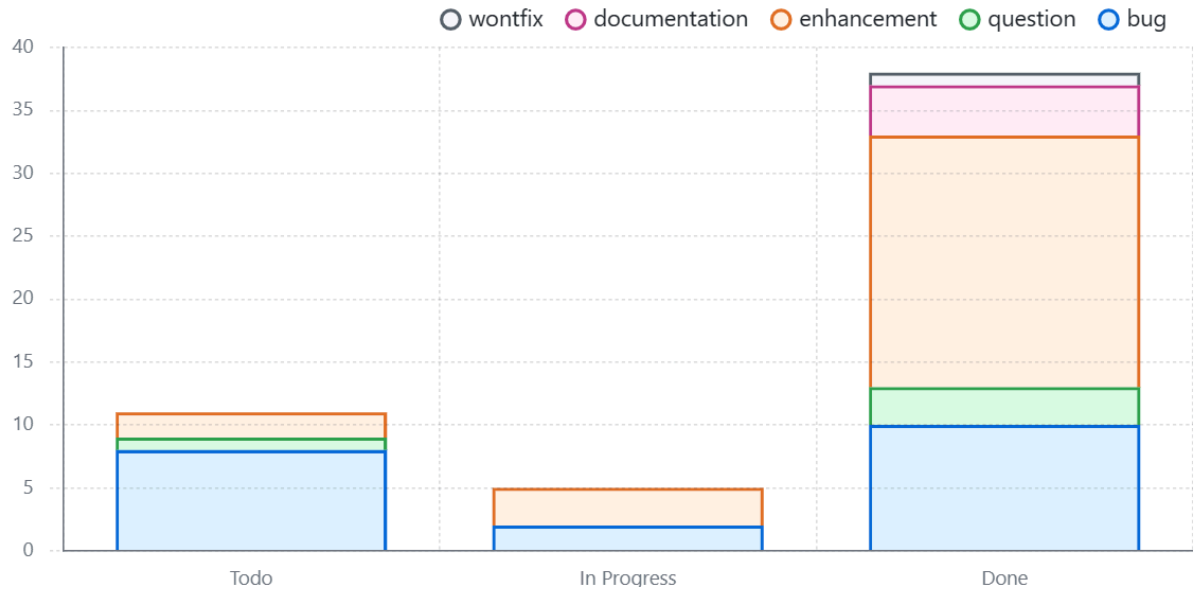
5	Sahaf	☹ 😕 😐 🙂 😀	☹ 😕 😐 🙂 😀

Week 10 w/b 08/03/2024

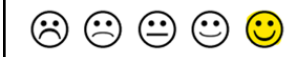
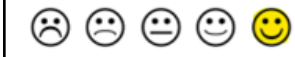
GROUP ID: Kelvin's Kittens			
Aim – Tasks you aim to achieve this week		Who is responsible? Names	Did you achieve this aim?
1	Issue #35 (Implement TinyMCE for risk assessments), Issue #19 (formatting of action buttons), Issue #11 Add an edit button to the bookings list if the booking can be edited.	Jake	☹ 😕 😐 🙂 😀
2	Complete the analysis and research section of the project report	Ben	☹ 😕 😐 🙂 🙂
3	Notifications system prototyping and research 1) Find middleware 2) Check Feasibility of implementation into the code based on time available 3) Develop pseudocode for high level prototype for nodemailer testing 4) Setup official email for app 5) Implement nodemailer test code	Noah	☹ 😕 😐 🙂 😀
4	List of buttons be in two columns not in 1#15 Adding complexity to password/complexity checks for adding user	Sahaf	☹ 😕 😐 🙂 😀
GROUP HEALTH:			
Group Member		Happiness	Contribution

1	Jake	☹ 😕 😐 🙂 😀	☹ 😕 😐 🙂 😀
2	Ben	☹ 😕 😐 🙂 😀	☹ 😕 😐 🙂 🙂
3	Noah	☹ 😕 😐 🙂 😀	☹ 😕 😐 🙂 🙂
4	Sahaf	☹ 😕 😐 🙂 😀	☹ 😕 😐 🙂 😀

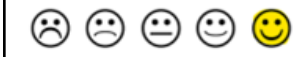
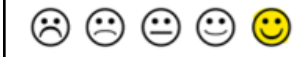
Week 11 w/b 15/03/2024



GROUP ID: Kelvin's Kittens			
	Aim – Tasks you aim to achieve this week	Who is responsible? Names	Did you achieve this aim?
1	Issue #45 review options for moving the database from planetscale due to hobby tier closing on 9 April which will	Jake	☹ 😕 😐 🙂 😀

	mean that the system won't run after that date.		
2	Add a button to order by confirmed risk assessments on the requests list page. Finish the analysis and research section of the project report.	Ben	
3	Find solutions to authorisation issues with email providers smtp services: - Yahoo - Gmail - Microsoft - Aol Develop pseudocode for implementation of confirmation emails - regardless of functionality - Log for the report (add later)	Noah	
4	File upload to add room	Sahaf	

GROUP HEALTH:

	Group Member	Happiness	Contribution
1	Jake		
2	Ben		
3	Noah		
4	Sahaf		

Week 12 w/b 23/03/2024

GROUP ID: Kelvin's Kittens			
	Aim – Tasks you aim to achieve this week	Who is responsible? Names	Did you achieve this aim?
1	Issue #46 implementation of database and install of app on goldsmith servers. Kill existing virtual servers and re-create servers and install from scratch. Issue #47 resolve differences between page behaviour in dev environment and production environment. #47.1 & 2 urls assume / as the folder for the app but in production will be /usr/199. Impact in .ejs's, app.js and in ajax javascript. #47.3 tinymce doesn't render because the html is not in standards mode. This works in dev. And in production something is stripping off <!doctype from the html that gets to the browser. Security: re-implement header security so that it works on the university servers and in the development environment by using the .env file (#42, & #43). Project report documentation for this week and assistance with other elements.	Jake	☹ 😕 😳 🙂 ☺
2	Fix the app crashing when entering incorrect login information on the login page. Fix the issue on the rooms-list.ejs route - the images are still not all the same size. Fix the crash when loading the report-bug page Complete the project report with group members.	Ben	☹ 😕 😐 🙂 🙂

3	Fix error with SMTP authorisation (Comment out nodemailer code if a solution is not found) Complete the project report with group members: - *remember add citations for previous work - work on reference list - work on code snippets - Add nodemailer section to to the Evaluation Add 2FA to evaluation Citations Group roles	Noah	
4	#16 Order by buttons on list pages should be alongside the heading and not taking up space below the heading, this would leave more space for the data list on each list page	Kelvin	
5	Issue #45 Change the way risk assessment approval is saved. Requires changes to the database, app.js file, and the necessary ejs files for pages involved in risk assessment approval.	Spike	
6	File upload to edit room Delete room button if you want to remove a room from the list	Sahaf	

GROUP HEALTH:

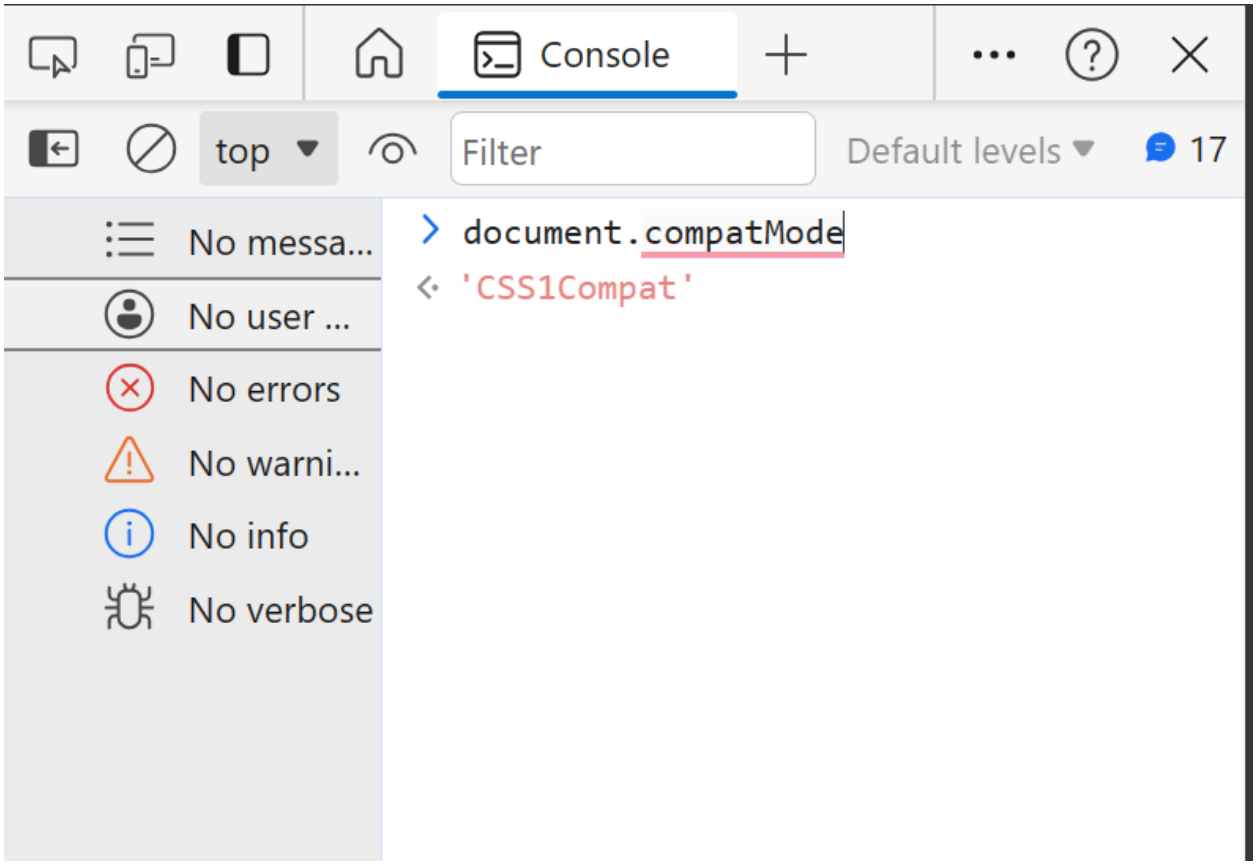
	Group Member	Happiness	Contribution
1	Jake		
2	Ben		
3	Noah		

4	Kelvin	☹ 😕 😐 😀 🙂	☹ 😕 😐 😀 🙂
5	Spike	☹ 😕 😐 😀 🙂	☹ 😕 😐 🙂 🙂
6	Sahaf	☹ 😕 😐 🙂 🙂	☹ 😕 😐 🙂 🙂

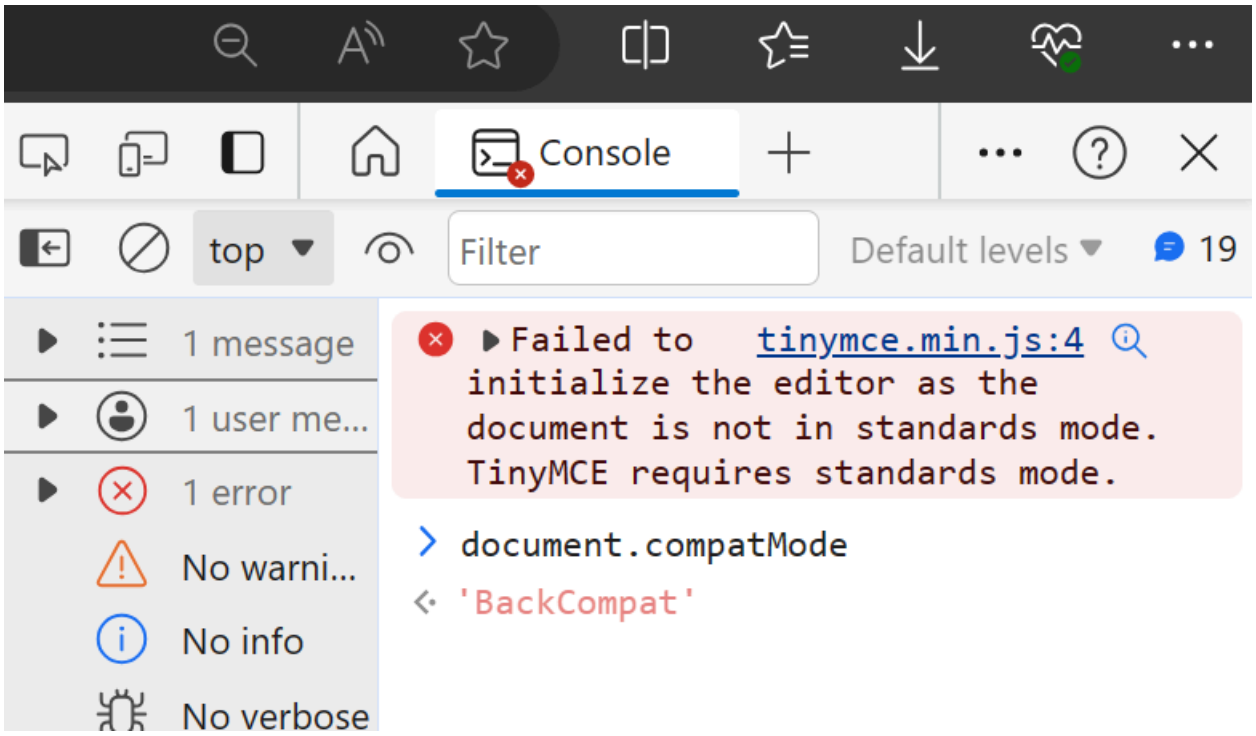
It was not possible, within the project, to resolve the issue caused by the <!DOCTYPE tag being removed by the servers before reaching the browser. The consequence of this was that the tinyMCE display and editor for the risk assessment would not display on the production server, but it worked in the development environment. Our tests resulted in the conclusion that this was something related to how the web pages were displayed via apache on the servers (we did not have access to this config). TinyMCE requires a standards compliant html document to work within and when the <!doctype is not in the html it is not in standards compliant mode. Here is a comparison of the developer tools console for the edit booking page on the development environment and the production environment. All code was synced by git and so there is no difference in the source.

Understanding and trying to fix this issue wasted a lot of time that could have been spent closing other items on the Issues list.

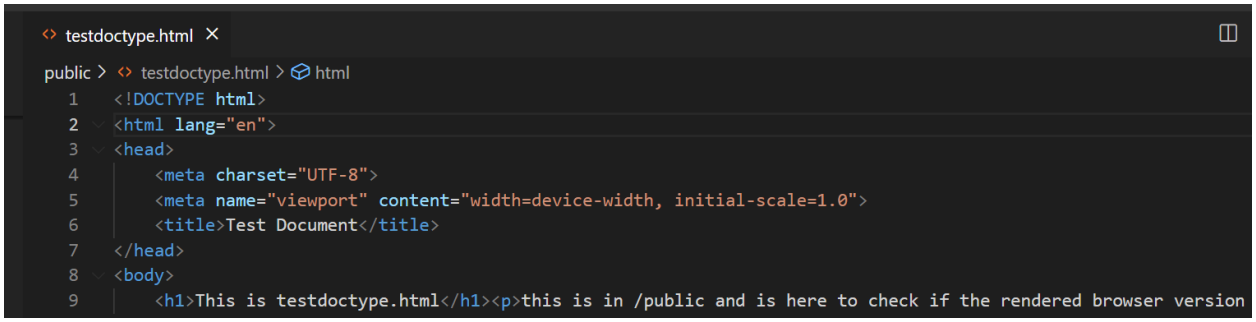
Local development environment result:



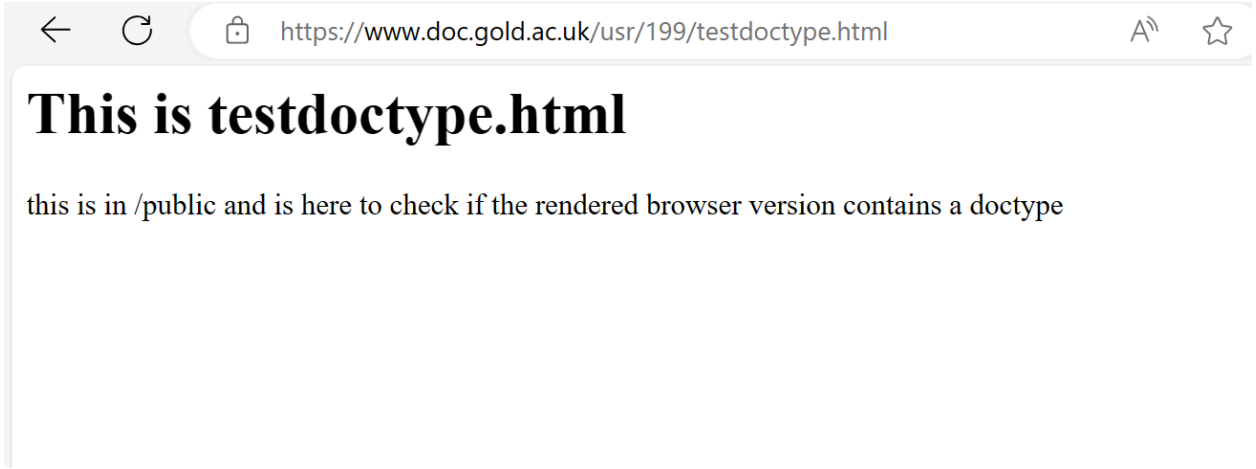
Production server result:



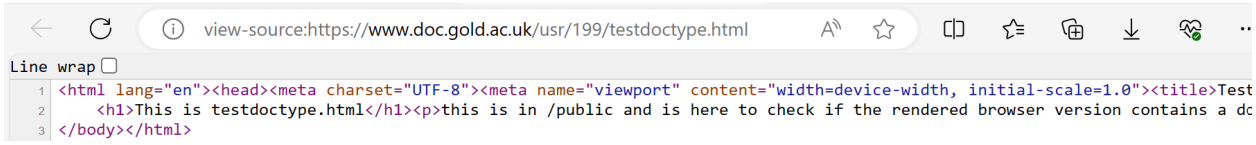
In addition this page was created in the /public folder of the project (Test Document (gold.ac.uk):



This is the page when displayed on the server:



And viewing the source shows the missing <!doctype



Version control logs

Roombooking commit history

All issues that were in the backlog

Bookit roombooking Issues list (github.com)

Interviews:

Interview structure:

Q. How will we conduct interviews?

A. In person, with an interviewee being given free reign to speak in response to talking points

Q. Who will we interview?

A. Society Leads and Lecturers.

Q. How to get access to these people?

A. Two societies lead in our group. Multiple social links provide us access to society leads at Goldsmiths and other Universities. Email through lecturers.

Q. How will this affect our scope?

A. Allows us to adjust the requirements by priority of things they wish to see or struggle with in room booking and management. This will also inform things such as low fidelity to high fidelity.

Q. How will we follow-up?

A. These interviews will enable us to create a survey in response to the interviews/ low fid feedback which we can then feed back into our plan and high fidelity versions..

INTERVIEW TRANSCRIPTS:

Khar Chew

INTERVIEW BEGIN

Speaker 1: So could you please tell me a little bit about yourself, your name, what university you go to, any societies that you are part of, and what position you hold?

Speaker 2: My name is Khar Chew. I am a second-year student at Goldsmiths University of London. I am currently the president for the Anime Games Society.

Speaker 1: Okay. So, we are developing a room booking system for societies and for lecturers. We're going to ask you a couple of questions about how you feel about the current system to help us inform our development process. As a society member, how do you feel about your involvement in the room booking process and the timetabling process for your events?

Speaker 2: That position, that role for room booking is mainly in charge of the secretary. However, I am partially responsible when picking the rooms suitable for events.

Speaker 1: Are there any problems that you face with the current systems, such as scheduling issues or overlooking the allocation of space?

Speaker 2: Yeah, it's a lot of issues, like having to book the room two weeks in advance and it takes time for us to get a response. By the time we get a response, we don't get the room we want, and they end up giving us a different room, which is completely unsuitable for our event. And when we submit a new request, replying to that saying we want the room to change to a new, more suitable one, they don't come back to us in time or they say, "You should have booked it two weeks in advance." So it's uncooperative.

Speaker 1: Okay. So given those problems that you face with the system, do you feel that the current format for accessing and utilising the system satisfies your needs?

Speaker 2: It does, the main function is to book a room, which does its thing. Timing for schedules is just hard because we don't know which room is available or not.

Speaker 1: So what functionality do you feel the current system is missing?

Speaker 2: A calendar where it shows which room would be booked and when it's available, so everyone can see which room has been booked, and you can plan ahead to make sure that that room is guaranteed to be yours at least.

Speaker 1: Do you feel that there are any unnecessary aspects of the system that should be removed, and what are they?

Speaker 2: For me, the health and safety forms become repetitive, which isn't really necessary since most societies fill out new forms for the same event types. Which is really not needed.

Speaker 1: So the current format for the booking processes is through Google Forms, is there a format that you would appreciate using more than Google forms?

Speaker 2: Probably a redesign - using a separate app/website which has a UI. Just easier to understand and more accessible and simpler and straightforward to look at and understand. Because on the form, it's just a straight paragraph of just putting details in, which gets a bit annoying.

Speaker 1: Okay. Thank you. That's all the questions we have for you today.

INTERVIEW END

INTERVIEW BEGIN

Syed Sahaf

Speaker 1: So could you please tell me a little bit about yourself, your name, what university you go to, any societies that you are part of, and what position you hold?

Speaker 2: My name is Sahaf. I am a second-year student at Goldsmiths University of London. I am currently the secretary for the Anime Games Society.

Speaker 1: Okay. So, we are developing a room booking app for the university. We're going to ask you a couple of questions about how you feel about the current system to help us inform our development process. As a society member, how do you feel about your involvement in the room booking process and the society's scheduling?

Speaker 2: As the secretary, I'm the one who handles the room booking process. In our society, it's very cumbersome as we don't know exactly when rooms are free. We're not given a clear time to be available when rooms are free. So we just have to guess which rooms are free, which times it's free and, and it takes a while to get feedback on these room booking requests to see when they are free.

Speaker 1: Are there any problems that you face with the current systems, such as scheduling issues or overlooking the allocation of space?

Speaker 2: Even when we get feedback, a lot of the time they give us rooms that aren't suitable if we didn't get the room we wanted. This feeds into scheduling issues, not just booking and allocation.

Speaker 1: Given those challenges, do you feel that the current format for accessing and utilising the system satisfies your needs?

Speaker 2: It’s too overcomplicated and cumbersome. We need a better system in place.

Speaker 1: So what functionality do you feel the current system is missing?

Speaker 2: A calendar system, seeing which rooms are free at certain times and days. Maybe a map to see where rooms are would be useful.

Speaker 1: Are there any unnecessary aspects of the system that should be removed, and what are they?

Speaker 2: Maybe the fact that we have to complete multiple different forms, and the email system for everything is really cumbersome. Combining forms and integrating risk assessments with the booking system would be useful.

Speaker 1: Compared to the current Google forms, is there a particular format that you would appreciate more?

Speaker 2: Maybe a webpage or something like that? It would be more efficient as part of the room booking system.

Speaker 1: Okay. Thank you for your time. That's all the questions we have for now.

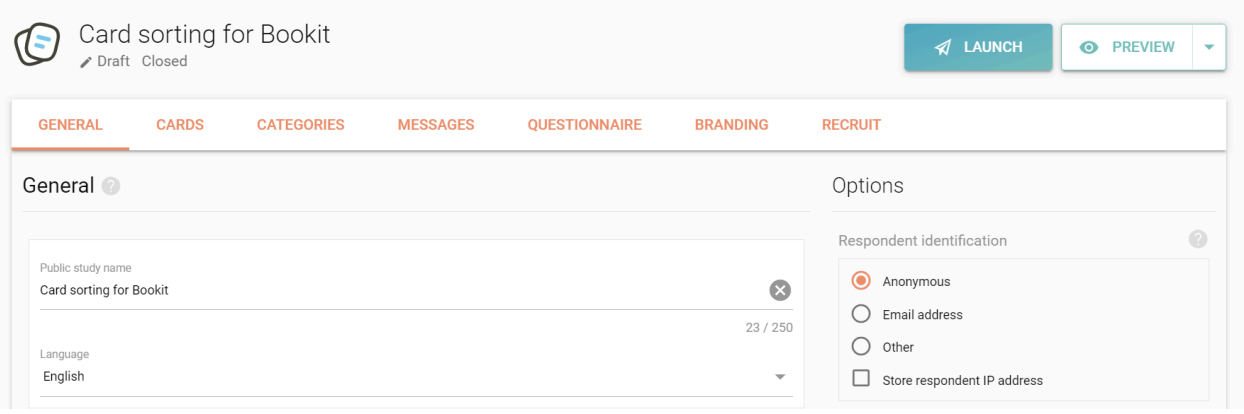
INTERVIEW END

Card sorting resources

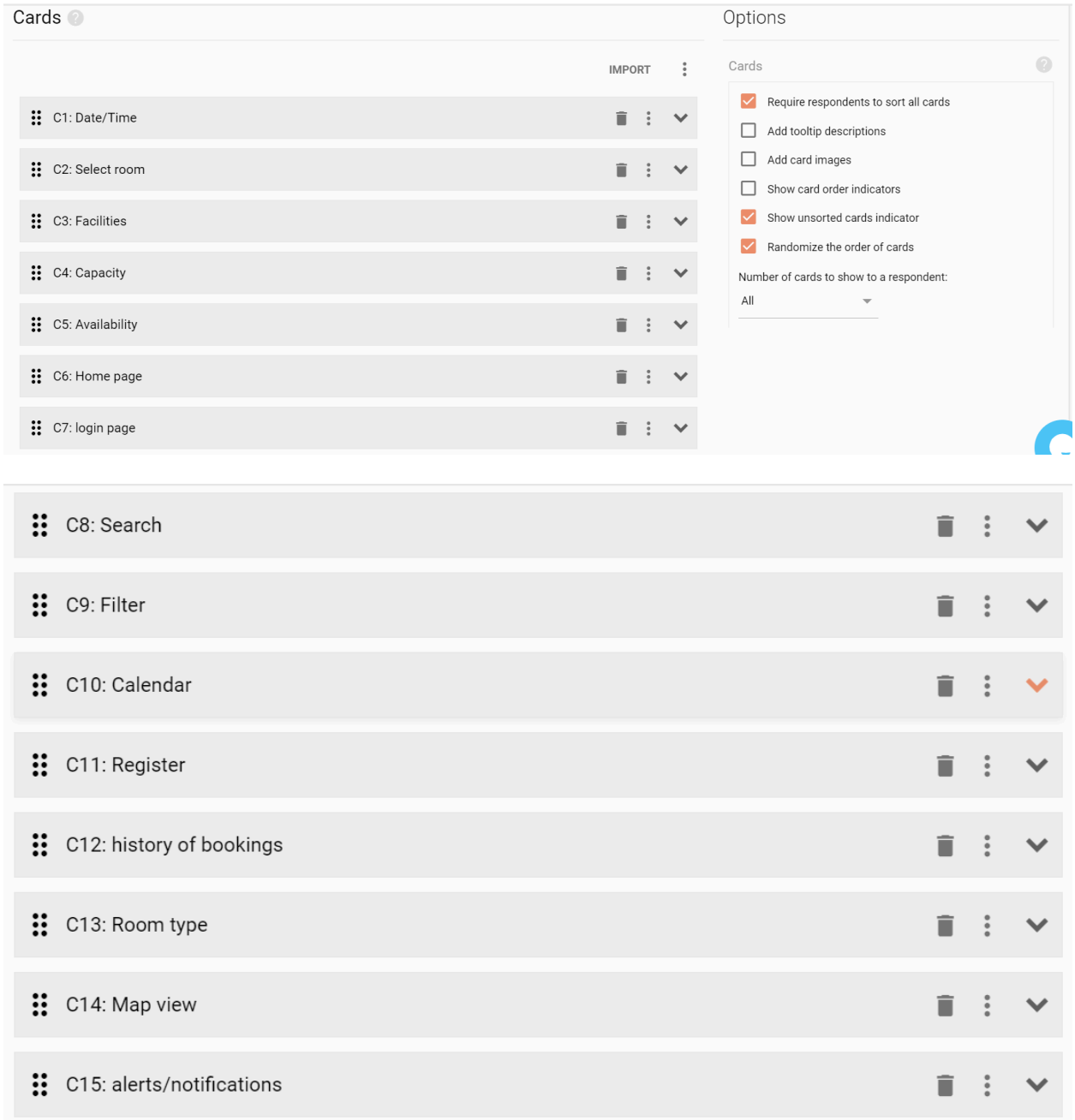
Raw data:

Cards	Booking process	Navigation	User account	Search	Integration	Notifications	Performance	Accessibility	Usability
Date/Time	4	0	1	1	1	2	0	1	0
Select room	6	1	0	1	0	0	0	1	1
Facilities	4	3	0	1	2	0	0	0	0
Capacity	3	0	0	1	1	0	3	0	2
Availability	5	0	0	0	0	1	2	2	0
Home page	0	3	3	0	1	0	0	0	3
login page	0	1	6	0	1	0	0	2	0
Search	1	1	0	8	0	0	0	0	0
Filter	2	0	0	5	1	0	1	0	1
Calendar	4	0	0	0	2	2	0	1	1
Register	1	1	7	0	1	0	0	0	0
history of bookings	6	0	2	0	0	0	0	0	2
Room type	7	0	0	2	1	0	0	0	0
Map view	0	4	0	0	0	0	1	1	4
alerts/notifications	0	0	1	0	2	7	0	0	0
Loading times	0	0	0	0	0	0	8	0	2
response time	0	0	0	0	1	2	4	0	3
scalability	0	0	0	0	3	0	3	1	3
user profiles	0	0	8	0	1	0	0	1	0
Font/colours	0	0	2	0	1	0	0	6	1

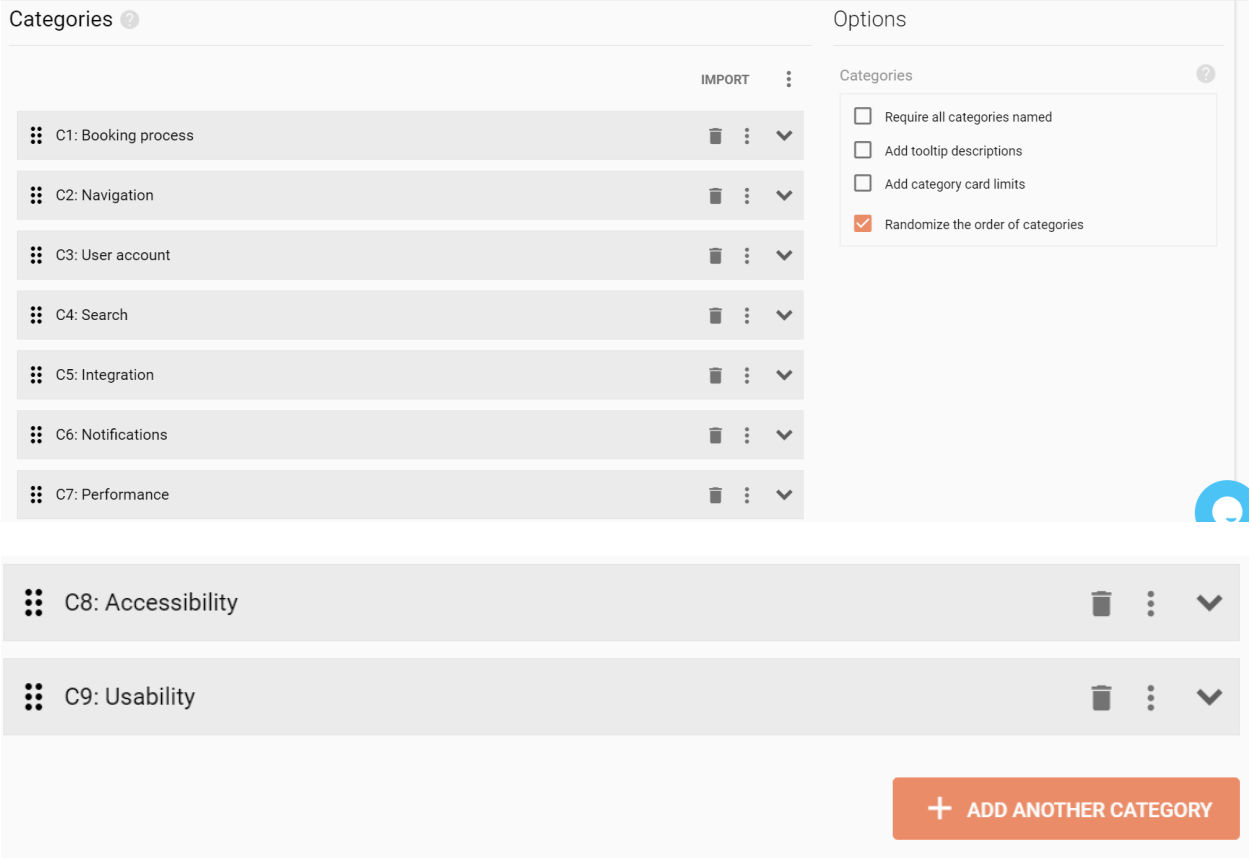
UXtweak overview



Cards


Cards
IMPORT
C1: Date/Time
C2: Select room
C3: Facilities
C4: Capacity
C5: Availability
C6: Home page
C7: login page
Options
Cards
Require respondents to sort all cards
Add tooltip descriptions
Add card images
Show card order indicators
Show unsorted cards indicator
Randomize the order of cards
Number of cards to show to a respondent:
All
C8: Search
C9: Filter
C10: Calendar
C11: Register
C12: history of bookings
C13: Room type
C14: Map view
C15: alerts/notifications

Categories



Group strengths and weaknesses list (semester 1)

List of relevant skills we have as a team:

- Kelvin

    - Experience in JavaScript, Java, Python, HTML 5, CSS, Lua, Haskell
    - OOP
    - Databases – SQL

- Spike
    - Object-Oriented Programming
    - Proficient in JavaScript
    - Limited experience with front-end development (HTML5, CSS)
    - Experience working in a team to a deadline, communication skills, delegation of tasks, critical thinking, problem solving

- Noah
    - 1 year experience programming in JavaScript using Object Oriented Programming – Computing project 1 and Game Project
    - Some experience in Front End Web Development – using HTML, CSS, and JavaScript
    - Writing, analysis, and creative writing skills from an A-level in English Lit and Language
    - Long-term Portfolio Development Skills (Drama portfolio, Film portfolio)
    - Team Working and Project Management (from A-levels group working)
    - Team leading/training (Senior role at my Barista job, often taking over managerial duties for extended periods of time)
    - Time management (splitting time between studies and part-time working)
    - IT Tech Support skills – building computers from components, installing software packages, troubleshooting, floor walking.
    - Familiarity with both Windows OS and Linux (Ubuntu)

- Sahaf
    - Experience with front-end coding – JavaScript, HTML, CSS
    - OOP
    - Writing and documentation experience from A-Level Comp Sci and History
    - Experience with Lua and Python
    - IT Support skills
    - Team worker

- Jake
    - Java
    - Java with Android Studio (Android Mobile Apps) OOP
    - Databases: SQL. MySQL. Also, Google Firebase for NoSQL cloud databases.
    - Web application security. OWASP.
    - JavaScript, HTML5, CSS

- Ben
    - Decent experience with web-based coding in JavaScript – as well as knowledge of HTML5 and CSS (Cascading Stylesheet).
    - Java – understanding of OOP.
    - Python.
    - Writing and documentation.

- Shaquille M.
    - Has experience with some JS (Uni Year 1)
    - Some experiences in front-end web dev with HTML and CSS (Uni Year 1)
    - Writing, documenting & planning (Computing Year 11&12)
    - Hardware knowledge (Computing Year 11&12)
    - Co-operative, team worker

Identification of matches and mismatches within our listed skill sets as a team are as follows:

- Matches
    - Javascript
    - HTML
    - CSS
    - Writing + Documentation
- Mismatches
    - Very little of us work with databases