Building a Platform for the Escape Game Community

COM3610 Dissertation Project

Simon Fish | Supervisor: Andrew Stratton

This report is submitted in partial fulfilment of the requirement for the degree of Computer Science with a Year in Industry by Simon Fish.

2020-04-15

Signed Declaration

All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations that are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

Simon Fish

Abstract

Escape rooms are physical, interactive experiences in which a group of participants must solve puzzles to escape a locked room, solve a mystery, or otherwise meet some goal in a particular timespan. This report uses various studies into applications of escape rooms to discuss the priorities of well-made escape games, particularly with regards to the inclusion of technology. It also marks my current progress in research into my dissertation topic as expressed in the title. It focuses on the difficulties in applying technology in escape rooms and priorities that should be held in order to mitigate these. On this basis, it serves to document universal requirements to bear in mind when developing hardware or software for escape rooms.

Acknowledgements

My deepest thanks to my family, my friends at University of Sheffield Tea Society, and my mentor Dalbinder Kular. All have strongly supported me this year. Thanks also to Andrew Stratton for supervising and providing a guiding hand during the project. Thanks to Liam Woff of the Lockup Escape Room in Sheffield for providing helpful insights and allowing me to bring a group to his escape room. Finally, thank you to those working in Development at UKCloud for giving me the skills needed to build Blacklight - special mentions to Ben Saunders, Chris Couzens, and Oliver Nye.

Contents

Chapter 1

Introduction

Escape rooms are physical, interactive experiences in which a group of participants must solve puzzles to escape a locked room, solve a mystery, or otherwise meet some goal in a particular timespan. They are a phenomenon that has existed since around 2007 (Nicholson 2015), and are a growing industry. Escape rooms are run both by enthusiasts as solo ventures, and as franchises across the country. Nicholson (2015) documents escape rooms as the culmination of a variety of media. He identifies puzzle hunts as team-based problem-solving challenges, which, with treasure hunts, appears most similar to escape rooms. Various others have lent features to escape rooms, such as immersion in a story as the "hero", which DuPlessie (2013) reports as being something participants enjoy. This is embodied by live-action roleplaying, another inspiration for the genre (Nicholson 2015). Nicholson presents the precursors to escape rooms in further depth in his paper - Figure 2.1 summarises these.

Figure 1.1: Nicholson (2019) presents the precursors to, and inspirations for, the escape room phenomenon in this diagram. Adapted with permission from Nicholson, Scott. 2015. "Peeking Behind the Locked Door: A Survey of Escape Room Facilities." *White paper available online at http://scottnicholson.com/pubs/erfacwhite.pdf.*

Escape rooms have grown popular in many locations across the world as a recreational activity (Nicholson 2015; Stasiak 2016), even serving as a tourist attraction (Dilek and Kulakoglu Dilek 2018). Nicholson reports that the *Real Escape Game* by SCRAP was the earliest well-documented activity branded as such. SCRAP has gone on to develop escape rooms at a much larger scale than the typical escape room, which serves teams of an average size of 4.58 people (Nicholson 2015).

My goal in this project was to understand the needs of the commercial escape room industry, such that software or hardware could be developed as appropriate. Such a product would aim to increase efficiency or expand the industry with new capabilities. The escape room industry has a wide variation of scale in application, from enthusiasts running singular escape rooms to major companies such as SCRAP delivering escape games to dozens, if not hundreds, of people. It also has varied contexts - escape rooms on permanent fixtures, portable escape games, applications in education such as EscapED (Clarke et al. 2016). It can also be noted that during the COVID-19 pandemic taking hold at the time of writing, maintainers are moving towards products that enthusiasts can use in their own homes.

With a strong understanding of this in mind, and feedback from the community itself, a product could then be designed and built to target some subset of these needs. These needs may be related to various issues - such as making sure a timer is visible to the participating group and maintainer, or to processes that currently take more time than necessary, such as posting photos of teams to social media (Woff 2019).

Research Questions

Two research questions were identified, which the literature review stage of the project is focused towards answering:

RQ1: What has been reported about concepts used in escape rooms applied in different environments?
RQ2: Based on this, what can be established as the requirements for escape games, independent of their environment?

Relationship between Project and Degree Programme

The project did not directly tie in with any of my modules this year. I acknowledged during research that my solution could potentially build on skills learned from modules such as *COM3505 Internet of Things*, should my solution have incorporated microcontroller hardware. In such an instance, I would have elected to use the ESP32 microcontroller out of familiarity.

However, my approach has been influenced by learnings from software development-focused modules such as *COM1001 Introduction to Software Development* and *COM3420 Software Hut*. In the former, the concept of agile processes was introduced, though my experience with *COM390 Year in Industry* presented this in a practical manner that more directly inspired my approach. The limited time available meant that many Rails best practices were not transferred through Software Hut - instead, many of these came through to me from working in Development at UKCloud on my year in industry.

My approach was not reminiscent of the Scrum process used in my team last year. Instead, I elected to move towards use of Kanban, which would allow me to be more flexible with my priorities and adjust my principles in light of time constraints. This created a challenge for me. My work at UKCloud during *COM390 Year In Industry* gave me a strong set of principles regarding software development, particularly as regards programming. I wished to include industry-standard processes in my work, including continuous integration, vulnerability testing, and measurement of test coverage. While I was able to make efforts towards tackling the first two, I lament that I could not commit to any figure on test coverage, let alone track it. In essence, my experience at UKCloud widened the scope of my capabilities and priorities alike, but due to time constraints, I could not apply them to the fullest.

Chapter 2

Literature Survey

Escape room maintainers are able to apply technology to varying degrees. On the outside of the escape room experience, maintainers implement leaderboards, share team photos to social media, and interact with the team via a screen or handheld transceiver in the room. Inside the room, entire puzzles can be based on technology, if the owner has the necessary expertise. Some more unique applications within the room are also possible, such as using a hidden camera to take a photo of the team, apply filters, and display the photo among works of art as demonstrated in The Gallery[1], which I visited in July 2019. However, there is a general aversion to the use of technology in escape rooms, for a variety of reasons. Woff (2019) suggests that the time investment, reliability, and necessary expertise are some of the greatest contributing factors. DuPlessie (2013) approaches this from the perspective of immersion - with 70% of escape room games being purely physical activities (Nicholson 2015), DuPlessie recommends movement away from what he calls the "glowing rectangles" as our medium of choice.

The increasing application of IT in education means that computers are often part of the school environment, and can be used as a tool when building an escape room experience for educational purposes. Several studies cover the use of escape rooms as a means for education (López 2019; Rouse 2017; Peleg et al. 2019; Beguin et al. 2019). Rouse (2017) applied technology to an escape room in the classroom using a game loaded from a memory stick. This application seems understandable, as Rouse's audience was likely to have some basic level of expertise in, and enthusiasm for, handling computers as part of the digital native generation. However, in practice, it brings to mind the image of a small group of people crowding around a screen. Woff (2019) warns against situations like this, saying that visibility of the puzzle to the entire team should be a priority.

However, poor implementation does not mean technology should not be excluded from escape room environments outright. Instead, I feel technology has the potential to inspire change in escape rooms. Escape rooms and technology are inherently linked - digital escape-the-room games such as *Myst* precede and inspire physical escape rooms (Nicholson 2015). These forms of escape room evade one of physical escape room maintainers' greatest anathemas - resetting these rooms is as simple as resetting the game. Whether this is done by restarting an attempt, restarting the game, or removing save files and starting over, it is often trivial compared to how long it takes to reset escape rooms. Woff (2019) explains that resetting rooms can often take as long as 15 minutes, and that it is something escape room maintainers seek to optimise; the shorter a reset takes, the more time is available to welcome customers.

Commercial escape rooms are often permanent fixtures. This means that more immersive environments and more complex physical puzzles can be built. However, puzzles of a physical nature usually need to be reset by the room owner back to their original state, so that more than one group can attend a room in the same day. Contrary to this, escape rooms built in a classroom environment are usually of a temporary nature, and may even try to allow for multiple groups to attempt the room at once, at the cost of immersion and interactivity. A study by López (2019) organised its puzzles in a manner whereby puzzles could be completed in any order, allowing multiple groups to attend the room at once. One example puzzle given in the study was an exercise likely done on paper - exercises in this form make resets trivial, as a fresh worksheet is all that is necessary, but they also demonstrate the lack of immersion.

[1] https://escapist.nl/en/

Commercial escape rooms regularly employ more physical interaction - 78% of escape rooms employ a search for physical objects as part of the experience (Nicholson 2015). Some varieties of commercial escape room visit the other end of this spectrum, such as those developed by Tuzak, an Istanbul company developing portable escape games (Gündüz 2018). Portable or temporary escape games often trade immersion for greater logical challenge.

Application of technology in escape rooms comes with some uncertainty, and a break in the flow of the escape room experience can shatter participants' immersion and lead to negative reviews (Woff 2019). This also creates some difficulty when it comes to visibility - unless monitors are suitably placed and large enough to be viewed by a full party, the whole team may not be able to interface with a puzzle that applies technology. This is particularly an issue if a single typical workstation is set up - Nicholson (2015) warns of the danger of removing just one player from the "mental space" of the team.

Woff (2019) theorises that VR escape rooms such as EXIT VR[2] may be the next stage for the industry, which allow immersive rooms to be created while effectively eliminating the issue of resetting the room as above. This would bring the escape room cycle full circle, reincarnating the modern wave of physical escape rooms in the digital form that inspired them.

While the escape room industry cautiously explores the "glowing rectangles" DuPlessie (2013) warns against, the video games industry that lent it inspiration sometimes takes small strides to recede from them. This has resulted in concepts that escape rooms, and interactive experiences of all kinds, can learn from. *Keep Talking and Nobody Explodes*[3] focuses on asynchronous gameplay, in which one player must defuse a bomb while the others guide them using the bomb's manual[4]. Conceptually, the game shares fundamentals with escape rooms, but what is of most importance to the point is that the foremost task in the game is communication. Escape rooms already apply asynchronous gameplay - The Lockup Escape Rooms' *Meltdown* (Woff 2019) begins with most of the party in individual chambers, with the designated 'leader' communicating from outside - but *Keep Talking and Nobody Explodes* demonstrates that with technology, it is possible in almost any location. *1-2-Switch*[5] similarly pulls away from the screen, with the majority of its minigames relying on what is done physically with the Joy-Con controller and instructing players to face their opponent directly. It is an effective example of how to apply technology in a way that does not lock singular players into staring at a screen. Such examples as these can serve as positive influences in escape room development.

In summary, technology brings a variety of benefits, from quick resets when used as part of a puzzle to interesting and reactive ideas that may not otherwise be possible. Many ideas and inspirations have been discussed here. However, care must be taken to ensure that if the implementation separates one player from the group, it is applied in an engaging manner. Escape room maintainers value reliability, with negative reviews being the consequence for ill implementation (Woff 2019). The strength of escape rooms as an educational tool has also been demonstrated here, which is worth consideration when building for the escape room industry.

Keywords

I identified the following keywords for use in my search. The search was conducted using the Google Scholar search engine.

- escape room / puzzle hunt
- owner / host
- implementation
- software
- virtual reality
- education / classroom

This led to the following searches.

- escape room owner
- escape room host

[2] https://exit-vr.de/en/
[3] https://keeptalkinggame.com/
[4] https://bombmanual.com
[5] https://www.nintendo.co.uk/Games/Nintendo-Switch/1-2-Switch-1173186.html

- escape room software
- escape room education
- escape room classroom
- virtual reality escape room
- portable escape room

Additional searches were made based on discoveries such as escapED (Clarke et al. 2016) and Breakout EDU ("Breakout Edu," n.d.) to see where they had been applied.

Chapter 3

Requirements and Analysis

The objective of the project was to build a tool for the escape room community. The exact form in which this would come was to be dictated by the nature of the problem - if the scenario called for a tool that would be active inside escape rooms themselves, it would have been more likely to be hardware. At the initial stage, this was the intent - a set of networked microcontrollers/SoCs that would unite to track the escape room experience - but this was abandoned. The idea of building an escape room, even in the form of a prototype, was out of scope. Additionally, Woff (2019) warned that digital uptake in traditional escape rooms may be middling due to the inherent risk and time involvement, as discussed in the literature survey.

I met with Liam on the 20th November to discuss the challenges he faces as an escape room owner, and his philosophy in developing the rooms he offers at The Lockup. Liam and I discussed some areas that could be targeted, which inspired a survey I sent to the Facebook group created by Nicholson (2015). This group is for escape room enthusiasts, encompassing both maintainers and participants, to which I sought and gained access as part of my work. The majority of posts during research were from enthusiasts who would report back from rooms they have attended, though I have seen posts about types of puzzles that can be implemented. I chose to survey this group, as it appeared to be a central hub for what seemed, to an outsider, to be a sparse online community.

From the seven responses received, the following conclusions were drawn:

- The majority of the group already shares photos online, but those that do not are all interested by the prospect of it
- Memberships, in-character communications, advertisement among other escape rooms, and participant metrics are all of interest
- Memberships are agreed upon by those interested as something that should not be done without the involvement of technology

I was able to elicit a direction from these conclusions - the idea of a social network shared by escape room maintainers and enthusiasts allowed me to potentially target several of these factors at once. In particular, the purpose of this social network would be to allow escape rooms to advertise among others of their ilk and share photos. I concluded my analysis of the results by setting a goal statement.

Goal Statement

To design and build a system that:

- allows escape room owners to share photos with their community
- allows escape room owners to advertise among other escape rooms
- allows escape room enthusiasts to discover new escape rooms to tackle
- allows escape room enthusiasts to track which escape rooms they have cleared

---

With these as an initial guide, I began to set requirements using the MSCW system, aiming to complete all defined as **Must**- and **Should**-Have by the end of the project. Though these were not formally defined

and did not dictate the order in which I completed tasks, I thought on what I had learned during my year in industry - while haste was of the essence, I made strides to emulate the software development process as I had seen it firsthand.

Stories were written with parts of the goal statement ('epics') in mind and estimated according to their complexity using a modified Fibonacci scale. In agile software development, complexity estimates are preferred to time estimates as the former are easier to settle upon (Karlesky and Voord 2008). Stories with estimates any greater than 13 would need to be broken down into smaller stories. This was a principle I kept in mind from my year in industry. Atlassian suggests a differing scale and limit, but the basic principle of keeping stories small and manageable is there. Velocity is of the essence, and smaller tickets assist with that.

Another factor that was kept in mind when creating stories was keeping them open. Stories, as often as possible, would need to describe what the user would wish to achieve, as opposed to what the developer working on them should aim to do. Framing stories from this perspective allowed their implementation to remain open to change and interpretation.

As mentioned in the previous chapter, I have had more hands-on experience with Scrum than with Kanban. Kanban is employed when more flexibility is desired. It prioritises throughput and encourages a "culture of 'done' " by enforcing work-in-progress limits.With this in mind, it would be difficult to set more than two development milestones. Treating each MSCW category as an epic and taking the amount of time necessary for work on this report and my other modules into account, I aimed to work on the project for a month, devoting about two weeks to each milestone.

I figured that each milestone would take a similar amount of time - while Rails' built-in generators would likely ease the burden of scaffolding the initial functionality, setting that groundwork in the right way would take slow and careful decisions. Scaffolds would dictate the flow of the rest of the project to a degree.

- **Dev commenced** Mar 12th
- **Must done** Mar 26th (actual 24th)
- **Should done** Apr 16th (actual 13th)

After setting my focus, I decided my solution would take the form of a web application. Ruby on Rails was chosen as the development platform. The QOC analysis below was the source of this decision, in which I weighed up other potential candidates such as Iron (Rust), ASP.NET MVC 5 (C#), and Next.js.

Table 3.1: QOC chart for choice of stack, with priority values and totals revealed.

	Priority	4	5	4	2	4	3	3	5
	Criteria	Familiarity	Documentation	Stability	Linux support	Available Docker resources	Developer tools (generators)	Community	Development cycle
145	Rails (ERB)	5	5	5	5	5	5	5	4
108	Next.js	3	4	3	5	5	1	2	5
109	ASP.NET MVC 5	2	5	5	1	5	5	3	2
66	Iron	1	2	1	5	3	1	1	4
130	Rails (React)	4	4	5	5	5	3	5	4

Originally, I weighed ERB and React against one another as view engines. This estimation was taken on the assumption that I would use React even where interactivity and state management were not required. In hindsight, this approach would have slowed development. Instead, my final approach of applying React only where functional improved the quality of the application without hampering development time. I would have liked to have combiend the two further and made the site entirely usable without the need for JS, "embrac[ing] HTML over the wire" as Rails creator David Heinemer-Hansson encourages. I was not aware of the concept of progressive enhancement at the time.

Chapter 4

Design

This chapter discusses choices I made with regards to design methodology and choice of approach.

React Components: Atomic Design

I selectively applied Atomic Design[1] in creation of React components for Blacklight. I was introduced to this by an internal development team during my time at UKCloud. Personal projects I have taken on in React thus far were all in motion before I was introduced to this, so I saw Blacklight as an excellent opportunity to apply it.

In summary, Atomic Design serves as *"not a linear process, but rather a mental model to help us think of our user interfaces as both a cohesive whole and a collection of parts* at the same time*"*. While Atomic Design in full uses five different levels of hierarchy - atoms, molecules, organisms, templates, and pages - I employed only three of these and expressed only two in React.

Atoms were designed with the intent of taking in one or several Rails objects as props, and representing those. They would not use asynchronous calls to APIs. Molecules would comprise atoms, and in some specific cases use API calls to retrieve the objects to display. Both of these would be used for rendering as part of a page by the Rails templating engine.

My justification for use of Atomic Design is that it helped me to define a purpose and a set of rules for components that I created. I could limit the scope of a component with quicker self-justification if I could express that its capabilities should sit within certain bounds I had defined on principle. However, while it served useful in that regard, it was at times difficult to ascertain what should be an atom and what should be a molecule. In the final codebase, some components are considered atoms where they should be molecules.

Authentication: OmniAuth/Auth0

Authentication, as one of the first and most fundamental areas tackled on the project, was of high importance to get right. Personal familiarity lies with Keycloak in this instance. Other options explored included Dex, ORY Hydra, FreeIPA and Auth0. Auth0 differs from the others in that it is authentication as a service - it hosts its own authentication instance for one to use and manage through their platform, with similar flexibility to the other options. The rest are self-hosted. Self-hosting was a trade-off I almost immediately acknowleged - control and convenience were in the balance.

In principle, I would always choose control over convenience. Personally, I am strongly in favour of self-hosting, though during my time at the University, I have made necessary compromises here to avoid the pitfalls of self-hosting from interfering with my work. This would be another situation in which I would choose to compromise, as will be explained.

Again, my time at UKCloud drew me in the direction of Keycloak and its Red Hat-supported twin, Red Hat SSO. Both are OAuth authentication platforms, and have provisions for setting policies and

[1] https://atomicdesign.bradfrost.com/chapter-2/

permissions and creating groups. Red Hat SSO was running in production at UKCloud while I was there, and I personally built up a lot of responsibility for implementations in that area. My familiarity with it and trust in its production-readiness made it one potential candidate for authentication.

One of Keycloak's primary problems, from a software architecture perspective, is that throughput is limited by the server. This is something of a guarantee with any authentication server, but particularly in Keycloak's instance, much overhead comes from its responsibility to serve the frontend.

Hydra presents itself as being a thinner and faster alternative which does not serve its own frontend. This is positive, but demands that the developer create their own frontend - combined with my unfamiliarity towards Hydra, this would use more time than necessary. Were time not an issue, Hydra would have taken my interest and had far stronger consideration.

Dex seems more accessible than Keycloak to one with knowledge of the OAuth2 protocol. Keycloak has extensive documentation, but much is left to the user to create and discover. Dex gave a strong impression as regards user-friendly documentation. As with Hydra, if time were not a constraint, Dex would have also been explored and considered to a far greater degree - most likely, it would win out over Hydra for developer-friendly documentation.

Of course, Rails' own option of Devise with database authentication was ever-present. However, I would not be as comfortable using this as a potential SSO provider - in the instance that vulnerabilities were revealed and patched in Ruby that affected Devise database authentication, I would need to upgrade all of Blacklight in line with this. OmniAuth in Devise would serve as a thinner layer over the top and allow me to easily migrate authentication providers, should I wish to.

Auth0 was chosen as my authentication provider, albeit as a compromise in the name of stability and convenience. One benefit is that it is externally hosted with a served login page, saving further frontend development and creating one less self-hosting dependency. Deployment also became much easier as a result of this choice - I was able to deploy Blacklight through Heroku buildpacks without needing to run an authentication service in a separate container, or additional networking configuration. Developer documentation is additionally very strong with Auth0 - I am glad to report that my experience in implementing Auth0 was mostly straightforward.

In this instance, the choice to use Auth0 as an authentication provider was valid for several reasons. Creating a dependency of large size would mean that in the event of a situation that called for further development, an extortionate amount of time would need to be expended. Such situations could include, for example, a security flaw or a significant update. Centralising this on two major dependencies - Auth0 and the `omniauth-auth0` gem, which are both managed by Auth0 Inc. - puts it in far safer hands.

Rich Text: Markdown

My intent from the offset was to allow escape game maintainers a strong degree of control over their profile, with the intent that much of the data could be used by enthusiasts for filtering. This would allow them to find escape games to suit them. Part of this was allowing them to submit a rich-text description for their room, including text formatting and links. The aim here was to take inspiration from Kickstarter[2], which allows rich-text descriptions that campaign runners use to good effect with images acting as subtitles.

Ovadia (2014) thoroughly describes use and benefits of Markdown in a variety of contexts. In selection of Markdown as the engine to drive rich text for Blacklight, it was strongly acknowledged that users may find learning Markdown "prohibitively difficult". However, Markdown is employed on blogging sites such as Tumblr, Reddit and WordPress (Ovadia 2014), with which I expected escape room maintainers may have some familiarity. Though I could not devote time to building a fully-fledged Markdown editor as a React component, I endeavoured to build a usable approach with a live preview and a link to a Markdown guide for those unfamiliar with it.

[2] https://kickstarter.com

Chapter 5

Implementation and Testing

My commitment to emulating industry best practices carried through to implementation. I felt it vital to use and work with such things as continuous integration servers, containerisation, and linting tools to maintain code quality and readability. Of course, none of these decisions were taken lightly.

While there is still some discourse over whether containerisation in development environments is entirely appropriate, I chose to use it to maintain some degree of parity between running locally and running in CI. Another reason why I made this choice was that popular production platforms such as Amazon Web Services are backed by containers - running the app using containers sets a good precedent to running it on a platform like Kubernetes with the right expertise.

Continuous integration tests code at each commit. I was able to configure CircleCI to build the Docker image used locally, and lint and test the code - this combined well with my use of the GitHub Flow[1], in which I would aim to get both tests and linting passing before merging. Towards the end of my development cycle, when I had deployed to Heroku, I would also run through a set of checkout testing steps I devised as a manual end-to-end regression test.

Automated Testing

I aimed to use test-driven development wherever it was most critical. Primarily, this meant securing controller routes and making sure that novel routes outside the standard Rails scaffold functioned as expected. This included:

- additional routes created beyond a standard Rails scaffold, such as those that make the current user a maintainer/enthusiast or those that remove image associations from a `Clear` or `EscapeGame` object
- secured routes - tests were created that challenged whether some attacking user could change a user ID in a form to steal ownership of a record
- some validations, such as those relating to limits on the number of image uploads for `EscapeGame`s and `Clear`s.
- some queries, particularly those relating to user privacy and escape room hiding.

SimpleCov was the standard at UKCloud with regards to coverage testing. I would have employed this and aimed for at least 80% test coverage, had time allowed for it.

The RSpec test suite ran at each commit on CircleCI's platform, halting in the instance of a failed Docker build, linting failure, or failed test. Though the environment in which I ran tests locally was incredibly similar, continuous integration served to ensure that the `master` branch continued to pass and was stable.

There were instances in which merging was delayed. CircleCI limits usage and fails all builds that fall outside a given quota. In these instances, I would either wait until my credits with CircleCI had reset and rerun the build, or trust that running the tests locally qualified instead and merge. In the event of a CI failure in an industry scenario, this would be discussed in the department - generally, CI serves as a strict

[1] https://guides.github.com/introduction/flow/

gatekeep to `master`, so in the majority of cases, development departments would resolve not to merge to `master` until their CI server functioned again. Due to time constraints, I was not quite so strict around it.

Frontend testing

I omitted frontend testing that strayed too far beyond scaffolded tests due to time constraints. However, my use of React components for major features like the Explore page means that swathes of the app are untested.

`react-rails` documents component testing here[2]. Stateless components such as `Avatar`s would be better tested this way - all that would need to be asserted is that they are rendered with the correct props. Or, to state that more generically, the correct arguments are passed to them at page render.

Some components are dependent on asynchronous functionality - the Explore view makes a request to the app's `/explore.json` endpoint on first render, and again afterwards. In these situations, directly employing Facebook's Jest[3] would be preferable.

Automated regression and checkout testing against production

During my year in industry, this was in high demand. Deployment of the company's primary products would take some quite time due to the number of manual steps involved - checkout testing played no small part in this. I have documented checkout testing steps in Blacklight's README, with clear indication that if the project were to continue, checkout testing would be automated. If possible, I would also follow up deployment with automated checkout testing at every instance.

On user acceptance testing

User acceptance testing became difficult to conduct due to the COVID-19 pandemic. This event endangered the very purpose of Blacklight itself, and made it difficult to justify revealing Blacklight. The industry is already experiencing great turbulence worldwide due to government orders to stay at home, and would not have a use for Blacklight in its current form.

In order to make Blacklight suitable for this situation, the `EscapeGame` model would need to make some allowances for "in-home" escape games - sets of locks, puzzles, and props, which are distributed by escape game maintainers. These allow enthusiasts to run their own escape game experience in the home. I would likely do this by creating additional fields on `EscapeGame` and create another model through Rails' single-table inheritance[4] mechanism. The idea behind this would be that both kinds of escape game could be shown side-by-side in views such as the Explore view.

Security concerns

I aimed to make as complete a software product as possible, given the time. Part and parcel of this was ensuring security. In Rails, this tends to mean enforcing restraints on which users can edit models, and which fields a user is able to submit. There are two parts to this:

1. ensuring that users not associated with the object cannot edit it
2. ensuring that the object's associated user cannot be changed

In Blacklight, this is done as follows:

1. retrieving the object by way of a query that asks for it among only the current user's escape games; returning a 404 status code otherwise
2. setting the object's associated user to the current user

On its own, the second part of my solution would be insecure. The more standard way of doing this on update would be to strip out the incoming user ID from the hidden field in the form. However, there is

[2] https://github.com/reactjs/react-rails/blob/d5da11129459cd75fd003c75319b1f7440c37322/README.md#test-component

[3] https://jestjs.io

[4] https://api.rubyonrails.org/classes/ActiveRecord/Inheritance.html

good reasoning behind it. This methodology is used on create as well as on update, which means that users cannot create records in other users' names.

Chapter 6

Results and Discussion

The product of my development process is a web application written in Ruby on Rails, with React.js used to generate and drive some areas of the frontend. This is titled Blacklight - its namesake is the kind of UV light which is employed in puzzles in escape rooms all too regularly. Blacklight serves to emulate its namesake, being:

- ... present in the majority of escape rooms
- ... used by maintainers as an escape room tool
- ... used by enthusiasts to discover something new

At the time of writing, Blacklight can be accessed at https://blacklight-dev.herokuapp.com.

Functionality Overview

Blacklight allows users to log in through the Auth0 authentication service only, though this allows for plenty of expansion in and of itself. Users are prompted to define themselves as an escape game maintainer, enthusiast, or both. This only affects the user's experience, not their level of access.

Escape game maintainers have options revealed to them to create and manage escape games. These escape games have an array of associated data that can be added, including the expected time to complete them, their location (by way of Google Maps integration), images, and a Markdown-compatible extended description. Listings can also be hidden from view and access via a checkbox on the editing form.

Enthusiasts can browse these escape games and mark them as cleared by selecting the lock icon on any listing. This is consistent across the site. Doing so adds the escape game to the 'My Cleared Games' section of the site for that user. There, users can upload photos to associate with each escape room, which also appear to the left of the list in a singular photo gallery.

Users have profiles where they can write a bio, set their location, and link to their own website. Here, those interested can see all escape games by a single maintainer, and all escape games cleared by a user. Which of these is shown depends on whether they have self-assigned as a maintainer, enthusiast, or both.

Forgoing the later milestones, and many ideas that might even give Blacklight commercial viability with them, was a difficult decision to make. However, I am happy with the base functionality and usability that came to be.

Discussion

To give a summary answer, the goals of the project have been met. A tool has been developed that allows escape room maintainers to advertise their rooms and upload photos, and allows enthusiasts to do the same in relation to escape rooms that they have cleared. Measures have been taken to ensure that it is secure, functionally consistent, well-designed and feature-complete. Despite this, I cannot shake the notion that with more time available, Blacklight would feel whole, and potentially viable for public rollout. Of course, in the current climate, Blacklight may serve little purpose unless it were repurposed - in any other world, my personal feelings towards it would be justified.

My implementation of continuous integration and containerisation surpasses previous attempts I have made in personal projects. I feel as though I have refined skills related to testing, and have particularly made strides in building more interactive frontends with React. Room for improvement still remains - late in development of Blacklight, I encountered limits around my use of CircleCI, and I might have been able to capitalise on the premium feature of Docker layer caching. I was alerted to a potentially more suitable option in GitHub Actions, which would have alleviated both woes - with 3,000 free minutes of running available under my current GitHub plan, I would have been able to run somewhere over 400 builds at the average rate per month, which I expect would have been more than enough. For comparison, I have run 201 builds against CircleCI at the time of writing.

As regards further work on the project, the initial scope included the below as ambitions. The existing framework should serve to make these straightforward to implement, but I lament that they could not be part of the final product.

- improvements to factors relating to authentication, such as:
  - introducing two-factor authentication
  - permitting the use of social network accounts such as Facebook and Twitter for login.
- a 'friends' feature, that would allow users to track other users' activity on their timeline
- a timeline for logged-in users that would show the aforementioned, alongside introducing new escape games that had opened in the vicinity of the user
- the ability to reorder photos on escape games and clear records
- toggling of privacy on a user's profile per-field, e.g. visible to friends only or private
- metrics for maintainers - feedback on how many times their escape games and profile had been viewed recently
- time recording and milestones for escape games - design and functionality would take strong inspiration from LiveSplit[1], and escape game maintainers would have a method of completing these for enthusiasts who attend their rooms
- leaderboards for each escape game driven by the above
- pagination of outputs

Ideas further beyond this included:

- an extensive tagging system and increased depth of filtering in the Explore view, with a view to allowing those with accessibility issues to quickly find rooms suitable for them
  - It is of note that maintainers can outline this kind of information in the description of their escape rooms. However, this cannot be searched or filtered.
- refinement of the API to be more standard
  - This would include OpenAPI documentation[2], which could then be verified against the app using a tool such as Apivore[3].
- Integration with social media, allowing Blacklight to consume more data from Google Maps and display Facebook embeds for escape room pages
- A review system specific to Blacklight (i.e. not backed by existing Google Maps data), with some way of weighing more experienced users' feedback against newer users
- maintainer groups, particularly in the instance of large franchises where multiple people may be responsible for the franchise's online image - this would allow multiple people to have ownership of an escape game
- as previously mentioned, measures to include "in-home" escape games, which would entail:
  - an additional filter
  - further fields available to the `EscapeGame` model, such as a definitive `price` - this was omitted as a field in its own right in Blacklight, as 39% of escape rooms charge per team and 55% charge per person, with a further 5% charging in other ways (Nicholson 2015). Such a difference calls for at least two unique approaches to both collecting the data from, and representing the data to, the user.

[1] https://github.com/LiveSplit/LiveSplit
[2] https://swagger.io/specification/
[3] https://github.com/westfieldlabs/apivore

Chapter 7

Conclusion

The escape room industry is wary in its use of technology, making a priority of stability. The experience delivered by escape rooms regularly relies on the physical, as opposed to the digital, in order to build excitement and immersion. For those reasons, it is difficult to develop technology for use inside the escape room itself. This was explored in the literature review.

The needs of escape room maintainers, both internally to the escape room experience and externally of the community, were investigated. This concluded in a decision to build a social network, which both allowed maintainers to advertise among other escape rooms and share photos. Enthusiasts would also gain a captivating experience and useful tool in the pursuit of new rooms to tackle.

The product of this was Blacklight, a web application that meets needs expressed by maintainers and suggested by enthusiasts. Blacklight was engineered through thorough application of industry practices such as use of Kanban and automated testing, with some allowances made for time constraints. The result is a secured and usable web application using Ruby on Rails and React. This was able to meet the basic requirements, but I believe that with more time available, I might have been able to give it enough features for it to be a fully compelling product in its own right.

Bibliography

Beguin, Erwan, Solal Besnard, Adrien Cros, Barbara Joannes, Ombeline Leclerc-Istria, Alexa Noel, Nicolas Roels, et al. 2019. "Computer-Security-Oriented Escape Room." *IEEE Security & Privacy* 17 (4): 78–83.

"Breakout Edu." n.d. *Breakout EDU*. https://www.breakoutedu.com/.

Clarke, Samantha, Sylvester Arnab, Helen Keegan, Luca Morini, and Oliver Wood. 2016. "EscapED: Adapting Live-Action, Interactive Games to Support Higher Education Teaching and Learning Practices." In *International Conference on Games and Learning Alliance*, 144–53. Springer.

Dilek, Sebahattin Emre, and Nur Kulakoglu Dilek. 2018. "Real-Life Escape Rooms as a New Recreational Attraction: The Case of Turkey." *Anatolia* 29 (4): 495–506.

DuPlessie, Matthew. 2013. "Go Analogue." YouTube. July 11, 2013. https://youtu.be/tTcl5I0Wbzk?t=668.

Gündüz, Şafak. 2018. "Preventing Blue Ocean from Turning into Red Ocean: A Case Study of a Room Escape Game." *Journal of Human Sciences* 15 (1): 1–7.

Karlesky, Michael, and Mark Voord. 2008. "Agile Project Management," October.

López, Ángela Gómez. 2019. "The Use of Escape Rooms to Teach and Learn English at University." *Research, Technology and Best Practices in Education*, 94–102.

Nicholson, Scott. 2015. "Peeking Behind the Locked Door: A Survey of Escape Room Facilities." *White Paper Available Online at Http://Scottnicholson.com/Pubs/Erfacwhite.pdf*.

Ovadia, Steven. 2014. "Markdown for Librarians and Academics." *Behavioral & Social Sciences Librarian* 33 (2): 120–24. https://doi.org/10.1080/01639269.2014.904696.

Peleg, Ran, Malka Yayon, Dvora Katchevich, Mor Moria-Shipony, and Ron Blonder. 2019. "A Lab-Based Chemical Escape Room: Educational, Mobile, and Fun!" *Journal of Chemical Education* 96 (5): 955–60.

Rouse, Wendy. 2017. "Lessons Learned While Escaping from a Zombie: Designing a Breakout Edu Game." *The History Teacher* 50 (4).

Stasiak, Andrzej. 2016. "Escape Rooms: A New Offer in the Recreation Sector in Poland." *Turyzm* 26 (1): 31–47.

Woff, Liam. 2019. Personal communication.