University of St Andrews

CS4099

SH Project Report

OpDeck: An online card-based game to explore human response to predefined scenarios

Author:
Cameron Allan

Supervisor:
Dr. Ruth Letham

April 12, 2019

University of
St Andrews

Contents

Abstract

There are many situations in which it is desirable to know the opinions of a population. Traditional opinion polls have been used extensively throughout history, with paper surveys giving way to their online counterparts. This has made it possible to reach a much wider audience, while giving the surveyor more control over input restriction and validation. A further benefit of surveys in the digital age, is the possibility of enrichment through various gamification methods. While there do exist tools designed to gamify surveys, these are limited in depth - the interface and the wording of questions can be modified, but at their core the outputs of these products still adhere to the traditional Q&A format. In collaboration with the St Andrews Centre for Exoplanet Science [1], this project aims to build *OpDeck* - a framework that implements gamification at a deeper level, moving away from the standard Q&A format. Using *OpDeck*, games may be created, played, and visualised, with the aim of inferring user's opinions on certain subject matters through gameplay analysis. *OpDeck* is designed to be highly flexible, allowing all aspects of a game's story and development to be decided by an administrator.

0.1 Declaration

I declare that the material submitted for assessment is my own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated.

The main text of this project report is 8,056 words long, including project specification and plan.

In submitting this project report to the University of St Andrews, I give permission for it to be made available for use in accordance with the regulations of the University Library. I also give permission for the title and abstract to be published and for copies of the report to be made and supplied at cost to any bona fide library or research worker, and to be made available on the World Wide Web. I retain the copyright in this work.

Introduction

1.1 Motivation

This project is undertaken in collaboration with the St Andrews Centre for Exoplanet Science, with an aim to answer the question; how would the general human population react to the discovery of alien life? There has been research [2, 3, 4] into this area, but it is certainly not extensive and very little of it is systematic. The topic is vast, and is made more complex by the many factors that may affect the answer:

- How has extra-terrestrial life been discovered?
- What is the nature of the life, is it intelligent?
- Is alien life on our doorstep, or far enough away that it could never reach or harm us?

Anne Smith (Biology) and Christine Helling (Astronomy) of the St Andrews Centre for Exoplanet Science believe it could be beneficial to create a game capable of collecting answers to questions like these. Gamification has been used in many areas, and has been shown to consistently improve engagement [5]. Hopefully, this format of gathering opinions would allow participants to become immersed in the context of the quesiton, leading to more honest responses.

If it was possible to accurately capture players' thoughts and feelings in this manner, this could impact the way opinion data is collected across multiple faculties.

1.2 Achievements

The *OpDeck* framework can be conceptualised into three distinct sections:

- Game Interface
- Game Maker
- Visualisation

The game interface provides a portal through which participants can play games built and stored on the platform. The game maker provides an interface through which an administrator may create and edit games. Finally, the visualisation tools allow administrators to filter and visualise how players interact with the game.

These parts work together to form a pipeline that enbles an administrator to design a game that explores their topic of interest, which can then be played by any number of users. Following this, the admin may explore player's choices using the visualisation and filtering tools provided, or export the data to perform more extensive analysis.

1.3 Document Outline

After outlining the objectives of this *OpDeck*, this document reviews the software and literature surrounding gamification - the primary area of research with which the project is concerned. Covered then are the

processes followed, tools used and considerations taken into account when designing, implementing and testing the framework. This is followed by an analysis of user testing and evaluation. Finally, the software is evaluated with respect to the original objectives.

1.4 Objectives

As part of my Description, Objectives, Ethics and Resources (DOER) document I describe the objectives of this project. Over the course of the project, they remained the same apart from one change - downgrading **SO.2** to being a secondary objective, from being a primary objective. This was done as the focus of the project shifted towards creating an unbiased framework, while this objective was content-focused and therefore became lower priority.

1.4.1 Primary Objectives

PO.1 Devise and implement a game that presents the player with scenarios and allows them to choose from potential responses

PO.2 Devise and implement a flexible infrastructure to model and constrain scenarios and their impacts

PO.3 Devise and implement an infrastructure for capturing and recording player responses

PO.4 Implement basic visualisation of responses

1.4.2 Secondary Objectives

SO.1 Devise and implement an admin centre to allow easy creation of new game content

SO.2 In collaboration with Anne Smith and Christine Helling, devise a sample set of appropriate scenarios with impacts and populate the game

SO.3 Carry out an experiment to assess the effectiveness of the game as a tool to assess people's real world views

SO.4 Create more advanced visualisation and analysis tools

1.4.3 Tertiary Objectives

TO.1 Perform a wider user experiment

1.5 Definitions

Throughout this text I will use the following words defined as so:

- Player/Participant - A user playing the game through the UI
- Administrator - An user that has access to the game maker and visualisation applications. This would be an individual interested in researching the opinions a group holds on a subject matter.
- Game definition - A game within the context of the framework I have created. A unique game definition consists of a set of cards, a set of pillars, and a starting deck

Context Survey

This project aims to develop a framework for capturing the opinions of its users. This closely aligns with the idea of gamifying surveys - a concept that is introduced in Section 2.1. Section 2.2 describes existing products that attempt to fulfil a similar role to OpDeck, analysing their various strengths and weaknesses.

2.1 Survey Gamification

An emerging trend in modern business practice, *gamification* can be defined as 'using game mechanics and game design elements to measure, influence and reward user behaviors.' [6]. There has been some research into gamification as a tool to improve the quality of responses from and engagement with surveys. From this I aim to gain an awareness of the necessary considerations for designing and implementing gamification.

As of yet, most experiments into survey gamification have only gone so far as to change the wording of survey questions, show imagery alongside questions or change the answer input mechanics. The objectives of *OpDeck* represent a considerable departure from the standard survey format, with the goal being that the game can be enjoyed as a standalone experience. As this is somewhat uncharted territory, there are many psychological considerations as to the validity of the results that can be gathered through this new format. There exists a tradeoff between complex levels of gamificaiton that increase enjoyment, and the accuracy of respondent data. It is also difficult to be certain whether players respond the same way in a game as they might in real life. While deep investigation of these issues is not my goal, I deem it important to understand them, in an attempt to minimise any bias that the *OpDeck* framework could impose onto players.

A 2011 paper found that while gamification generally increases participants' enjoyment of surveys, there are three main aspects of gamification that can affect the 'character of the data' [7]:

e.1 Effects caused by changes to the question and how it is interpreted

e.2 Effects caused by changing the attitude and mindset of respondents

e.3 Effects caused by changes to the design and layout of question

While these considerations are targeted toward those gamified surveys that have a fairly standard format, I believe that they are still applicable to *OpDeck*.

A key aspect of *OpDeck*'s gameplay loop is the context that persists between questions, affecting players' choices. Harms *et al.* wrote in a 2015 paper [8] that 'Designers may also implement feedback loops, i.e., dynamics wherein user actions affect the overall state of gameplay. Feedback loops may visualize concepts such as a user's progress, status, wealth, health, points, etc.'. This gives me confidence that adding a persistent context to the game is a valid design decision. As for how this will affect the character of the data, I believe this relates to **e.1** (question/interpretation). This is because changing the context effectively changes the question, as people may respond differently under different circumstances.

e.3 (design/layout) applies not just to *OpDeck*, but to any presentation of a survey - where there exists an interface to interact with, there is potential for it to influence the way the user responds to it. I attempt to minimise this risk - ideally the only bias imposed on a player's decision is from the game definition itself.

Brownell *et al.* [9] summarise the state of research into survey gamification as follows: 'In most cases, their results show that the addition of these game elements increases the length and quantity of responses,

and respondents typically prefer the gamified version to the standard survey version. However, their research does not compare the effectiveness of game elements in gamified surveys. They have also found that that some gamified survey designs can lead to compromised respondent data (Puleston & Sleep, 2011).' [9]. This indicates that there is promise in the notion of gamifying surveys - however it can result in a reduction in accuracy of the data collected. Depending on the extent to which the survey is abstracted from, precise details, both of the question and response, can be lost. Given this, gamification is likely best suited to situations in which the surveyors need broad answers, rather than precise ones.

2.2 Similar Software

2.2.1 Datagame

Datagame [10] is a company that offers services allowing researchers to create and publish gamified surveys. This is done through recognisable interfaces that are heavily game-influenced, such as word searches and decks of cards.

Datagame supplies the tools required to populate one of several template games with custom questions, and export this to an online format that can be played through multiple channels including Facebook [11].

Datagame offer four customisable game types. That most similar to *OpDeck* is referred to as a 'Swiper' game. In this, the player draws cards from a virtual deck, each of which presents a yes/no question. The user swipes the card left or right to answer yes or no respectively. Figure 2.1 shows this interface - it is fairly simple, with no standout features other than those added during configuration. Customisation options include allowing the user to replace card backgrounds with images, as well as change the colour of the text. Figure 2.2 shows an example of the game editing view in which the following aspects of the game can be edited:

- Project title
- Title question
- List of cards
- Card name
- Card text/images
- Toggle card shuffling
- Game UI labels
- Background image

This format excels in allowing participants to answer a high volume of questions in a short period of time. The participant is asked questions directly, therefore there should be no ambiguity in their understanding beyond the wording of individual questions. The limitation of a binary choice reduces the precision with which players can answer a given question - this may be acceptable for only a subset of surveys not requiring precise answers.

While the interface has the appearance of a game, the gameplay elements are shallow. There is no conditional branching in this game, as cards can only be removed by answering them and there is no way to add cards to the deck during a game. Further, the participant's responses have no observable consequences on the future of the game and there is no ultimate goal. This leaves the player with little incentive to continue playing beyond the appearance of the interface, and the content of the questions themselves.

Given the above, while Datagame surveys are presented as games, at their core they are merely a wrapper around a basic binary choice survey format.

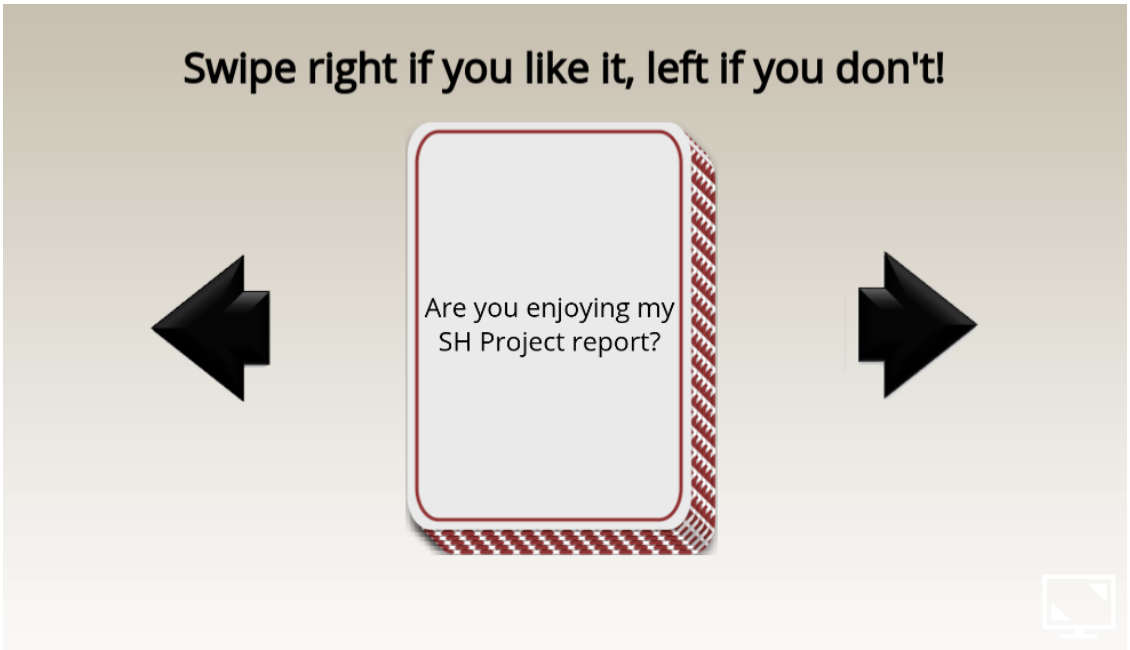

Figure 2.1: Example question in one of the Datagame [10] game types, which involves answering yes or no questions by swiping left or right.

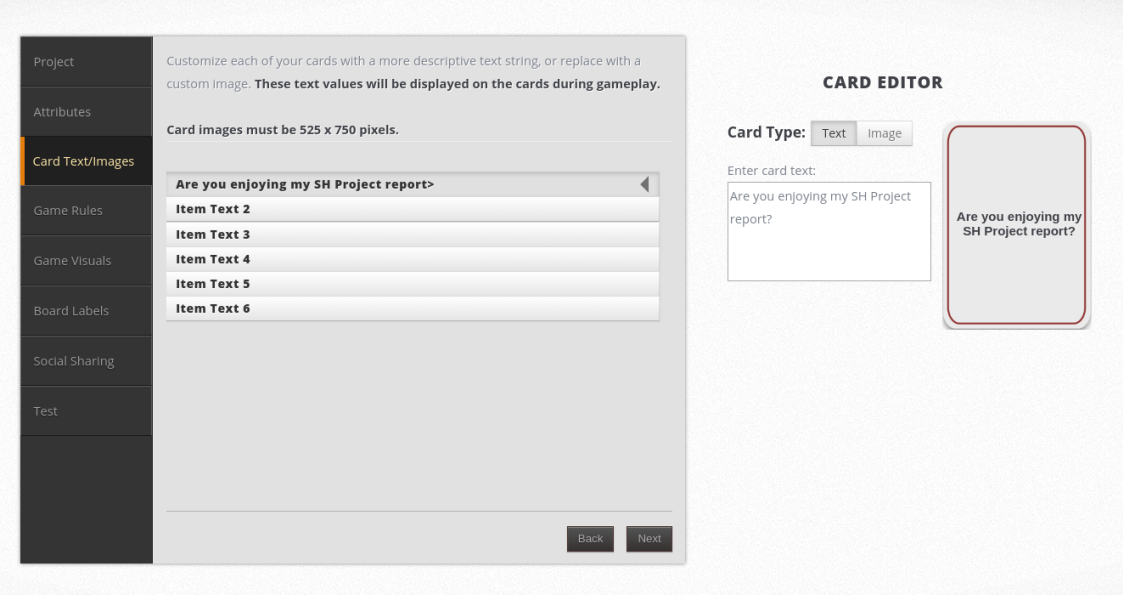

Figure 2.2: Process of editing the Datagame [10] game instance shown in figure 2.1.

2.2.2 Qualifio

Qualifio [12] offer a similar service, but provide a larger variety of game formats than Datagame. I was not able to access Qualifio's game creation tools, as this functionality is behind a paywall, however the site does host playable examples. Qualifio's toolset features a 'Swiper' style game similar to that provided by Datagame, which can be seen in figure 2.3. This comes with the same tradeoffs between engagement, accuracy of results and precision of input as the Datagame equivalent.

Figure 2.4 shows another game template offered by Qualifio, named 'Ranking'. Ranking provides a

Figure 2.3: Qualifio [12] swiper demo. This example game has players swipe or click depending on which team they think a given football player belongs to.

different approach to gathering user opinions, where multiple options are ranked in order of preference. This is done through a click and drag interface - more engaging and intuitive than the pen and paper equivalent of labelling each option with a rank number.

Ranking gives users more precision in their answers than is available with Swiper, however the results within a question are all relative to each other. If one user likes all of the options, and another dislikes them all, it's plausible that they will give the same answer - the absolute values are lost in place of relative data.

Figures 2.5 and 2.6 show examples of other game templates that Qualifio offer.

The prediction game in figure 2.6 provides an example of a game format that targets a specific category of questions - what players predict the result of an upcoming competition will be.

2.2.3 Reigns

Reigns is a multi-platform game in which the player takes the role of a medieval ruler, making binary decisions to solve problems their subjects approach them with. These decisions affect which scenarios may later appear, as well as changing how the ruler is perceived by various factions such as their population, army, and church. The player's success is measured by how many decisions they can make without falling out of favour with any of the factions.

Reigns is marketed purely as a game, with no data-collecting functionality. In addition, the framework on which it is built is proprietary, so users are not able to design game scenarios around topics of their own interest. For these reasons, Reigns cannot reasonably be used to collect and analyse data from players' choices.

As of 2019-03-01, Reigns is a well reviewed game, with a rating of 4.7/5 on the Google Play store [13]. Building intuitive creation, play, and analysis platforms for games in the style of Reigns has the potential to make rich gamification of surveys accessible to a wider research audience.

Figure 2.4: Qualifio [12] Ranking demo. This example has players rank football players from top to bottom - best to worst.

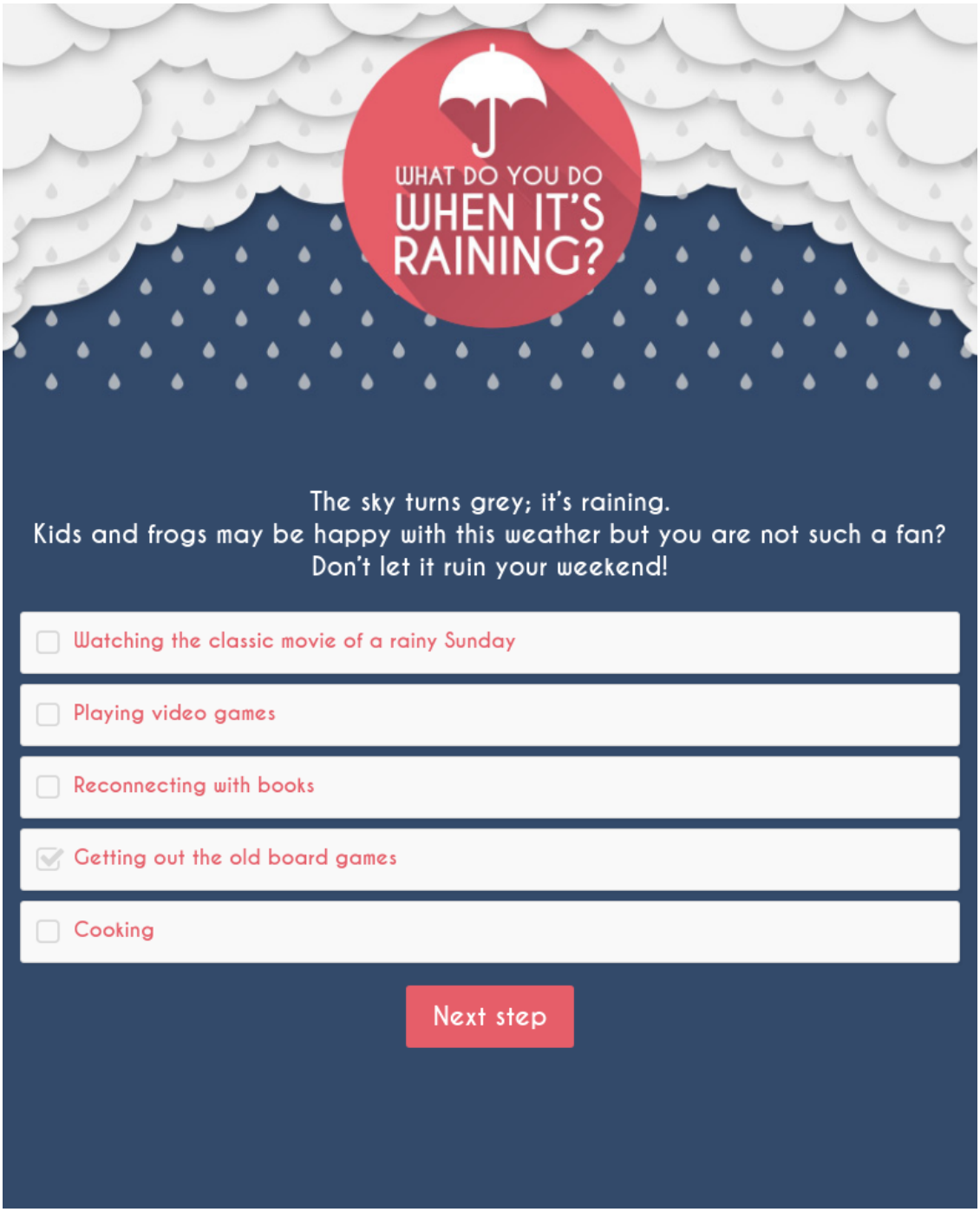

Figure 2.5: Qualifio [12] Checklist demo. Players check answers that they agree with.

Figure 2.6: Qualifio [12] Prediction demo. Players predict the scores of two parties.

Requirements

Below are the software requirements as specified before beginning the implementation of OpDeck.

3.1 Functional Requirements

FR.1 **User** *High Priority*

A user can choose a game to play by entering the game code

FR.2 **User** *High Priority*

A user can read and respond to scenarios presented to them

Depends on: **FR.1**

FR.3 **User** *High Priority*

A user can lose a game, when the value of one of their pillars reaches zero

Depends on: **FR.2**

FR.4 **User** *Medium Priority*

A user can enter a unique id, under which their game data will be recorded

FR.5 **User** *Medium Priority*

A user can see the effect that their responses will have on the pillars

Depends on: **FR.2**

FR.6 **User** *Medium Priority*

A user can lose a game, when there are no more cards in the deck

Depends on: **FR.2**

FR.7 **User** *Low Priority*

A user can view an image related to the current scenario

Depends on: **FR.2**

FR.8 **Admin** *High Priority*

An administrator can add, edit and remove cards from a game definition

FR.9 **Admin** *High Priority*

An administrator can add, edit and remove pillars from a game definition

FR.10 **Admin** *High Priority*

An administrator can save a game they are editing and return to it later, or let users play it

FR.11 **Admin** *Medium Priority*

An administrator can select specific cards and obtain visualisations that summarise how players respond to them

Depends on: **FR.1**, **FR.2**

FR.12 **Admin** *Medium Priority*

An administrator can export and download all game data to an appropriate data processing format

Depends on: **FR.1**, **FR.2**

FR.13 **Admin** *Low Priority*

An administrator can filter turns included in visualisations by the state of the game during those turns

Depends on: **FR.2**

FR.14 **Admin** *Low Priority*

An administrator can save their progress in editing one game and switch to another

FR.15 **Admin** *Low Priority*

An administrator can view a summary of the game they are creating, which contains information on any machine detectable oversights and game balance

Depends on: **FR.8**, **FR.9**

3.2 Non-Functional Requirements

NR.1 All interfaces are clear and intuitive

NR.2 The game interface is responsive to input - there should be minimal delay between an input action and the outcome

NR.3 The game interface does not influence the user's choice in a way that is not customisable

NR.4 User data should be stored securely

Software Engineering Process

As the majority of the work put into this project was in developing the software artifact, I focused from the beginning on following an effective software engineering process. This is detailed in Section 4.1. Section 4.2 is closely related, describing the tools that enabled me to follow the process.

4.1 Process

I employed an agile[14] methodology throughout my work on this project.

One of the key Agile principles is to 'Deliver working software frequently'[15]. I followed this from the beginning by setting myself intermediary deadlines, by which I planned to demo working sections of project. These deadlines were discussed and agreed upon with my supervisor. This helped me stay focused on getting the features I needed working while avoiding premature optimisation.

Another primary principle of the Agile Manifesto is to 'Welcome changing requirements, even late in development'[15]. As an example of this, one of my original requirements involved assessing how accurately *OpDeck* could gather people's real world views. After thinking more about the psychological aspects of this question, my supervisor and I decided that this was outside the scope of the project as it is closer to psychology than computer science. After realising this, we changed this requirement to performing a simple user evaluation.

4.2 Tools

With this in mind, one of the first tools I employed was the version control system Git [16]. Given the size of the project, it was important to be able to revert the project to a previous point in case some functionality broke. Version control software was the obvious solution, and Git is the one of these that I have the most experience with. To complement this, I set up a master GitHub [17] repository to use as a backup.

In addition to serving as a backup, GitHub provides a useful set of project tracking features. I did consider other tools often used to track projects, such as Trello [18], but in the end decided that GitHub's ability to reference aspects of the code (thanks to them being stored in the same place) made it better suited for my needs. Here I will outline the features that I used and describe how they helped my development process:

Issues These are a core element of GitHub; any task that needs completing is documented as an issue. I kept my issues fairly high-level, as I have found previously that too much detail in issues results in spending excessive amounts of time managing them or some inevitably become outdated. Figure 4.1 shows an example issue.

Projects Issues alone can become unorganised, so I made use of GitHub's projects feature. I split my work into four parts - game, game maker, visualisation and backend then made a project for each of these. This allowed me to keep issues relating to different parts of the project separate, making it easier to focus on one section at a time.

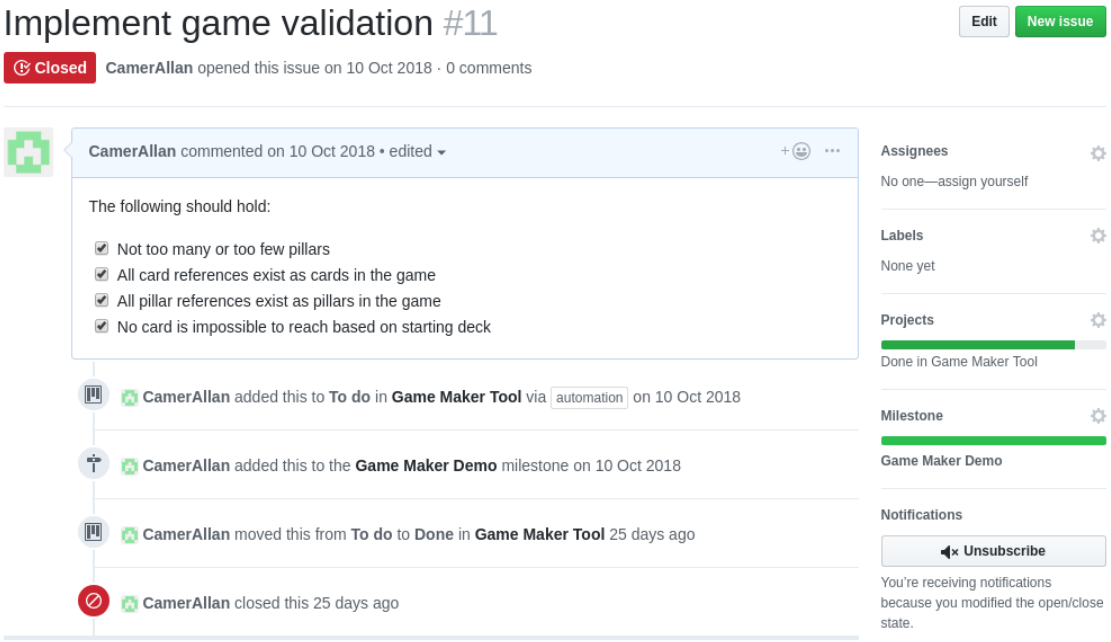

Figure 4.1: An example of an issue, note the assigned description, project, and milestone.

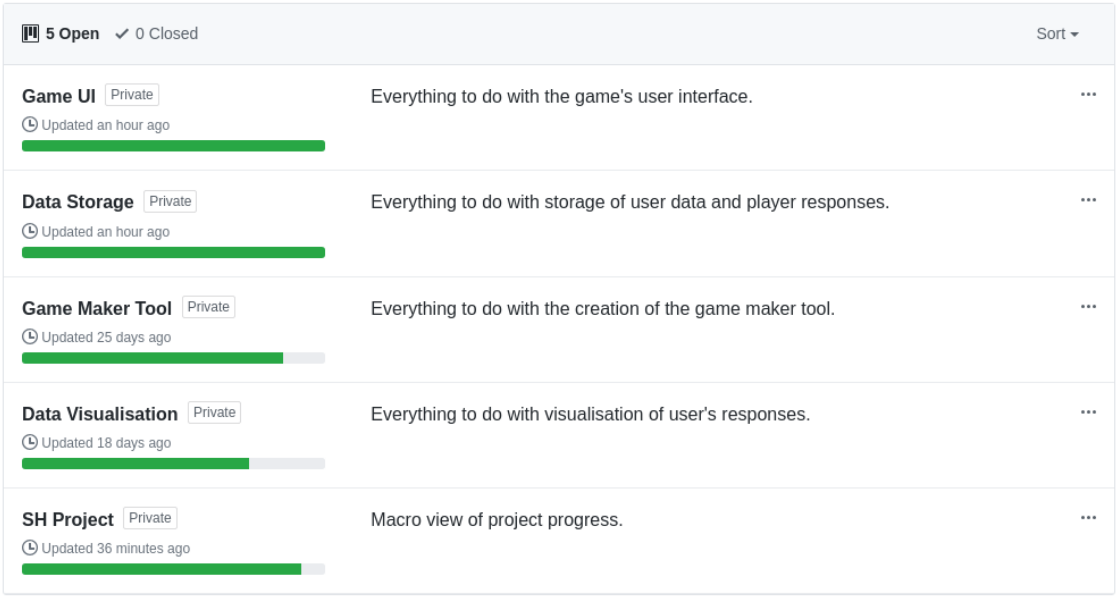

Figure 4.2: Project overview page.

Each project has its own kanban-style [19] board, used to track its progress, an example of which can be seen in figure 4.3. Issues assigned to a project start in the `todo` column. When being worked on, I would move them into `In progress`, and finally when closed they are automatically moved to `Done`.

The project overview visualises the proportion of issues in each column, providing an instant, accurate depiction of progress.

Milestones The first step I took in tracking my project was to set my Milestones. These are an aspect of a GitHub project that serve as a deadline, to which issues can be added. Completing and closing these issues

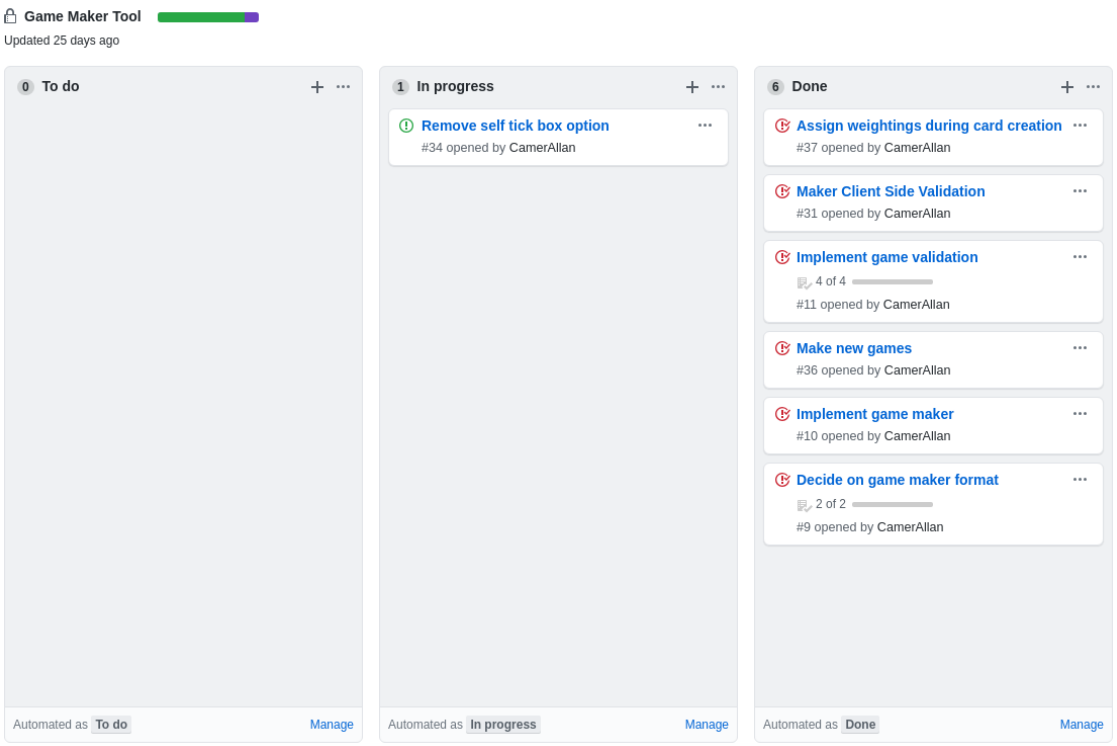

Figure 4.3: Example project board towards the end of the project.

then automatically provides a visualisation of progress towards a milestone, as can be seen in figures 4.4 and 4.5. These made for a helpful overview, which was useful both for myself, and for sharing my progress with my supervisor.

I decided on these milestones early on by estimating the dates by which I could complete core parts of the project. As this was done in advance I was not able to be precise with demo dates, however I completed the vast majority of work for each milestone in advance of the deadline, so I consider this a successful element of my planning.

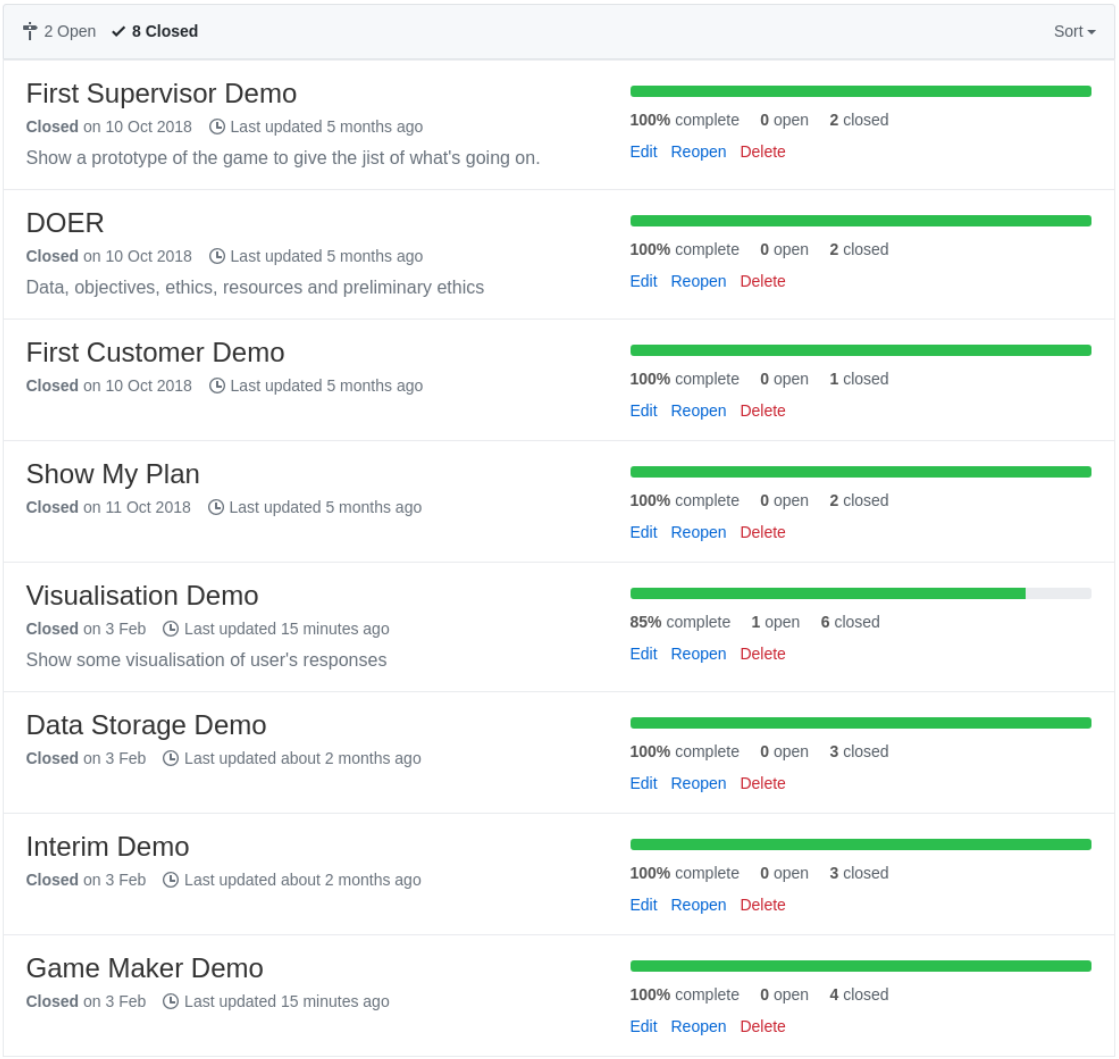

Figure 4.4: Closed (completed) milestones towards the end of the project

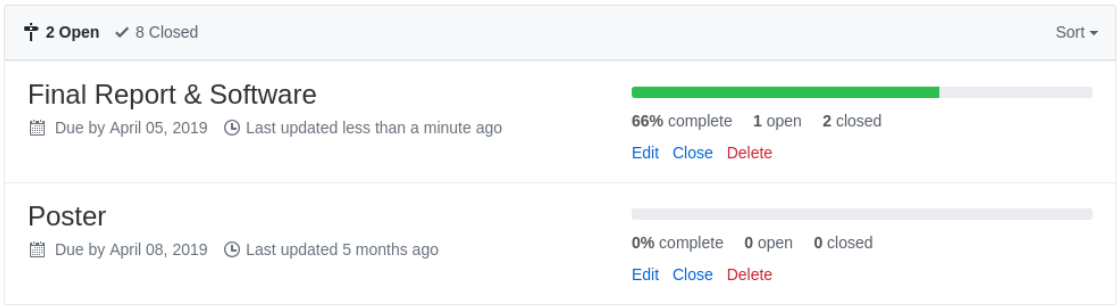

Figure 4.5: Open milestones towards the end of the project

Ethics

All data used for testing and development purposes was fictional. Fictional data was made up by myself, and consists only of user ids. No personal or identifying information was gathered during the user evaluation. Participants' answers were anonymously recorded on paper forms and kept in a secure location. The form contains the code of the ethics application that covered this excercise. The digitised forms were stored in an encrypted location. Both ethics application and user evaluation forms are attached as appendices to this document.

Design

6.1 Gameplay Overview

OpDeck's game UI provides a fairly straightforward experience from the player's point of view, however there is a significant amount of complexity that the user does not see. Conceptually it is a card game, so it will be described as such. The game consists of three decks which I dub 'in play', 'out of play' and 'reserve'. In addition to this, there are n 'pillars' - these are attributes of importance within the context of the game story. Each of these has a minimum, maximum and current value. The 'in play' deck has a defined starting set of cards, with the 'reserve deck' containing all others - 'out of play' starts empty. Both decks are shuffled at the beginning of the game, and each pillar has a predefined starting value.

The player draws and reads a card from the play deck, each one showing the following information:

- Title
- Description
- Choice #1 ('accept')
- Choice #2 ('reject')
- Requirements to draw

Each choice on a card consists of text detailing the response, and the effects of the response. Effects are made up of two parts - pillar changes, and cards added/removed. Pillar changes specify amounts to add or remove from one or more pillars. Cards added/removed defines which cards should be moved between 'out of play' and 'reserve'. 'Requirements to draw' consists of conditions that the current pillar levels must satisfy in order for this card to be drawn, such as the current value being within a certain range.

After each choice made (a turn), any cards in 'out of play' whose pillar requirements are met are moved to 'in play', while any 'in play' whose requirements are not met are moved to 'out of play'.

The game ends when any of the pillars fall to their minimum value.

6.2 UI

6.2.1 Game

As it is described above, the game takes a lot of effort on behalf of the player, having to sort and shuffle cards. Fortunately, this effort can be removed completely through work done by the computer. This leaves a simple game from the user perspective; users are presented with a card, make their choice, and get the next card.

This simple perspective means that the design for the game UI is quite straightforward. Figure 6.2 shows the first design for this UI alongside the final product. Initially, it was uncertain whether or not the pillars would be visible to the player. It is generally accepted that visual feedback is an important aspect of interface design [20]. An additional, important consideration in this decision was the edge case where a card may

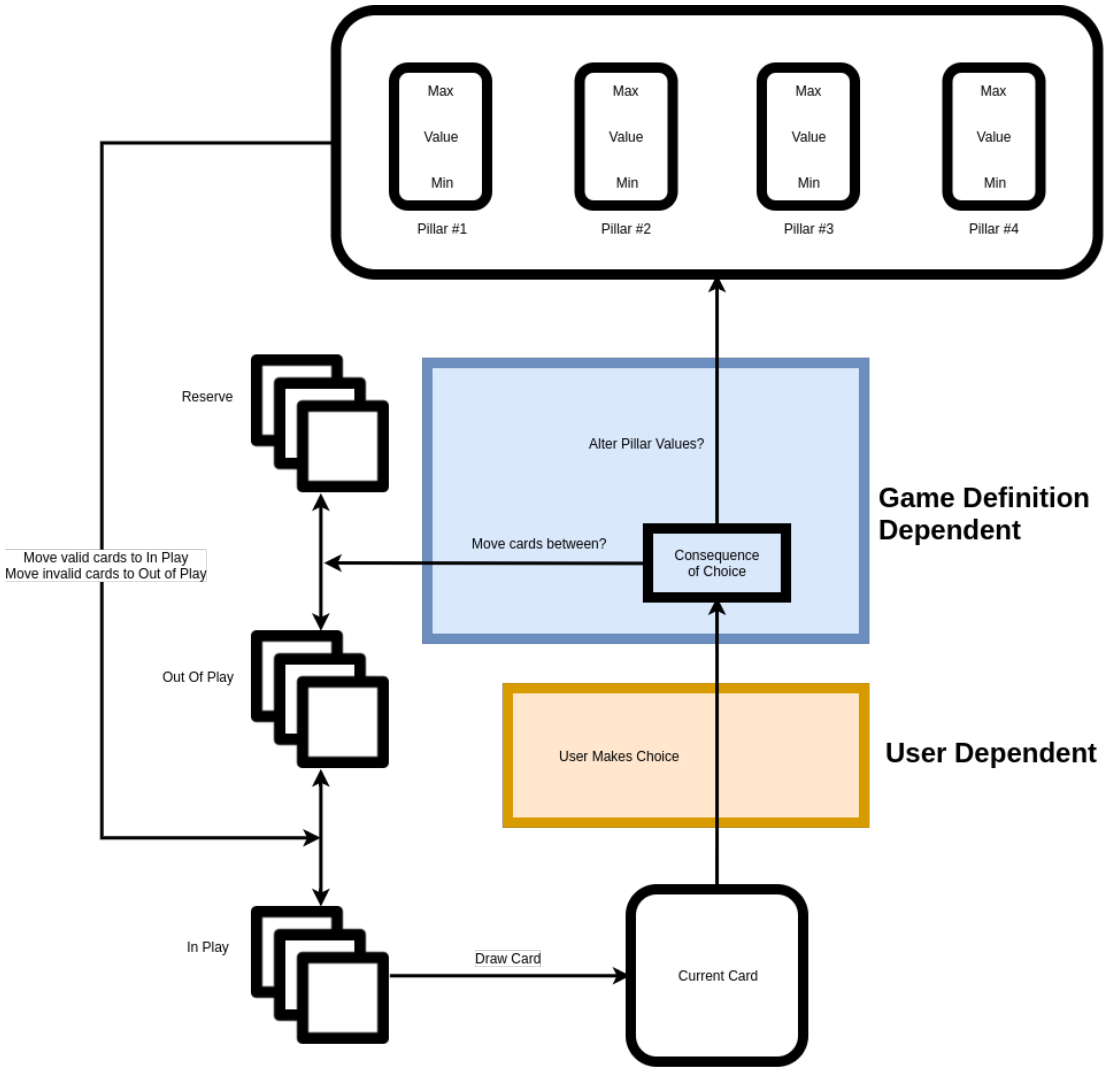

Figure 6.1: Diagram of the main game loop. Actions followed by a question mark are optional, depending on the game definition

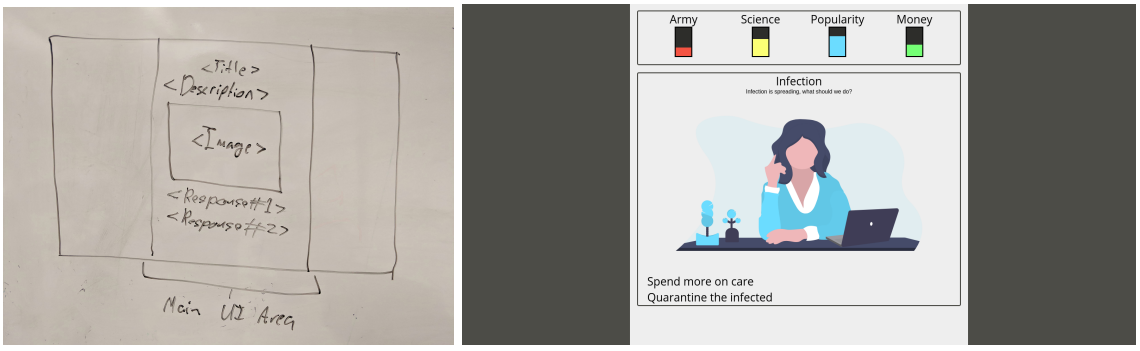

Figure 6.2: Comparison between initial and final designs for the game UI

present itself on two consecutive turns. Taking these points into account, it was decided that visualising pillar updates was important, so this was included at the top of the interface.

Pillars are displayed as vertical bars at the top of the screen. These fill from bottom to top, representing where the current value lies between the predefined min and max. Each pillar has its own fill colour (customisable in the game maker), which was inspired by Datagame [10] who offer the same feature for their Swiper game. This serves to make the game more visually engaging without imposing any particular colour scheme, as doing so could conflict with the desires of game creators.

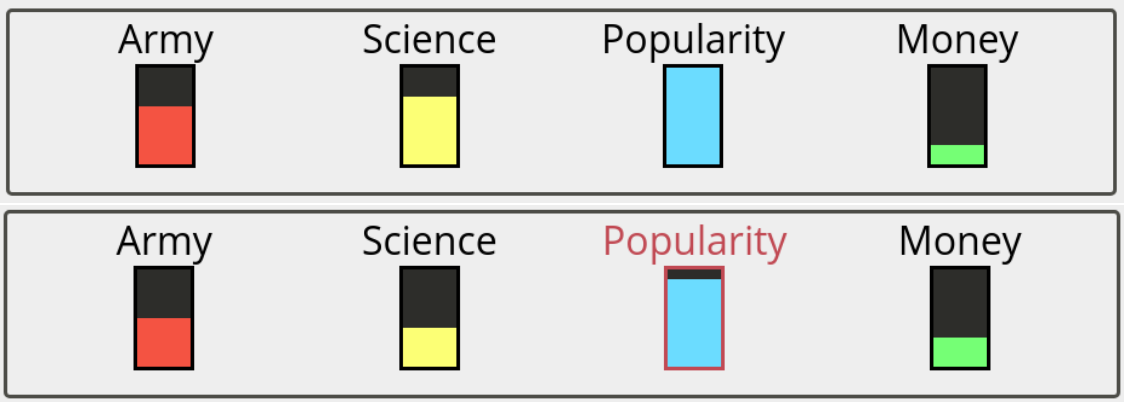

Figure 6.3: Visualisation of a negative outcome; the red outline around `Popularity`

The image shown with a given card is dependent upon the pillars that the card influences, as seen in figure 6.4. In this example game, each pillar was assigned an 'advisor' image, so that the advisor representing the pillar most affected by a card is shown.

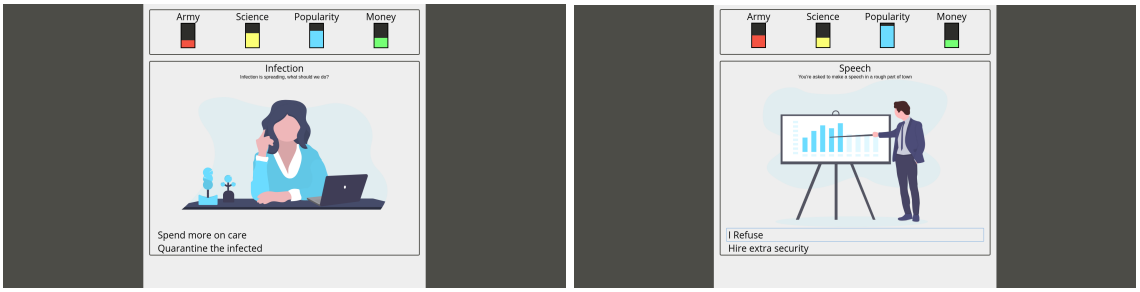

Figure 6.4: These cards primarily affect different pillars, so they have different associated images

As seen in figure 6.5, not only can the image change, but the primary colour of the image may also change. This feature is included as cards may affect multiple pillars, which would otherwise not be represented. With this colour shifting implemented, images can represent secondary pillar effects.

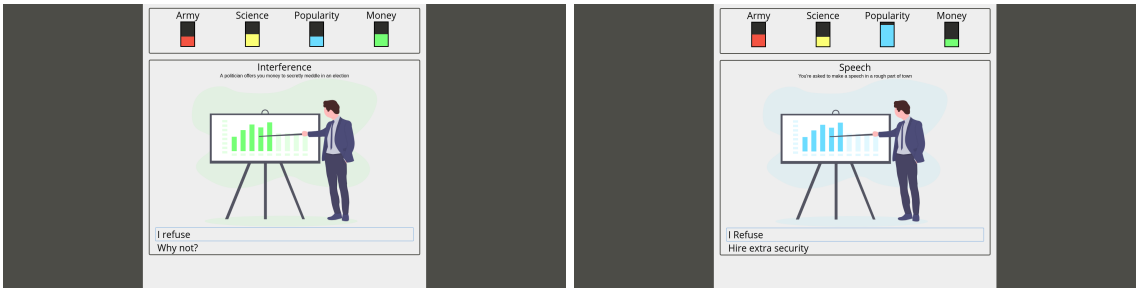

Figure 6.5: Both of these cards primarily affect the `Money` pillar (they are being presented by the `Money` advisor) however the second also affects `Popularity`, so has a blue primary colour

6.2.2 Admin tools

Game Maker

The game maker interface was the most challenging to design, as I wanted the user to be able to maintain a high-level overview of the game while adding and editing pillars and cards. After thinking this through, I initially settled on the design depicted in figure 6.6. The main theme is that editing is done in the left panel, while the right side continues to show an interactive view of the game, including a visualisation of the relationships between cards. The final design is roughly the same, however the card view is not as complex as the connections between cards are not visualised. There are two ways in which cards can be connected (Add to deck/Remove from deck), and further still, a card may be connected to an arbitrary number of others in these ways. For this reason the final design displays cards in a grid instead of a graph.

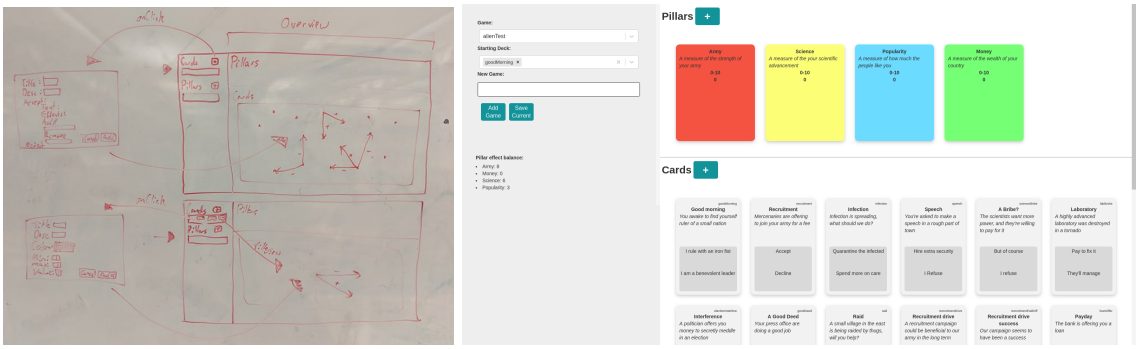

Figure 6.6: Comparison between initial and final designs for the game maker. Shown is the example game I created for demonstration purposes.

In the right hand section of the interface, the user can see the game pillars and cards that they have created. This side is scrollable, while the left section remains in view. If the details of a card are too large for the view, the text fades out. The card view can be expanded by hovering over it, as seen in figure 6.7 - when this is done, other cards shrink and move to account for the extra space needed. New cards and pillars can be added with the large + buttons by the headers, while existing ones can be edited by clicking on them. Both of these options open the edit view in the left hand section.

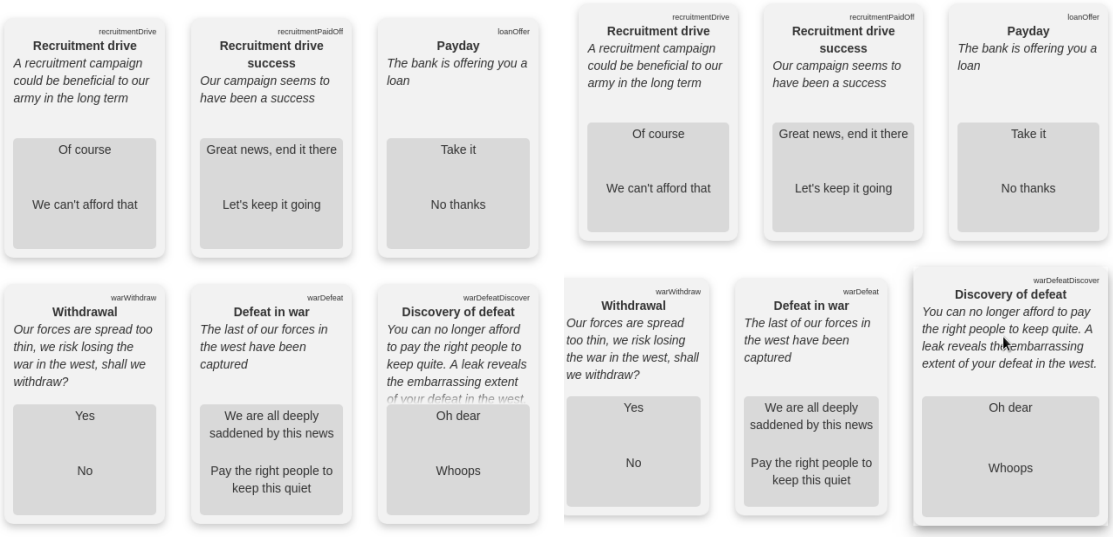

Figure 6.7: Demonstration of the hovering expanded view (note that the full text is cut off on the left, readable on the right)

The left hand section provides contextual menus. When no pillar or card is selected, it provides high-level functionality, such as saving, switching, or creating new game definitions. Also present here is starting deck selection, game balance information and any warnings.

'Pillar effect balance' provides the admin user with an idea of how balanced their game is in its current state. This is done by summing all of the effects that are applied to each pillar for any given response to all cards. This acts as an indicator of how well a user is likely to do if they randomly pick their responses; the higher the balance values, the easier the game. In addition, these figures serve to notify the user of any unintentional bias within their game. Understanding and manipulating the relationship between cards and pillars is key in gathering unbiased data.

Warnings appear when a card has no chance of being added to the game. This means that a card has been created, but it is neither in the starting deck, nor is it added by any other card. This is only a shallow check - if card A is added by another card B, which itself is never added, the warning will only appear for card B. I consider this acceptable, as once card B is added or deleted, the situation will be resolved.

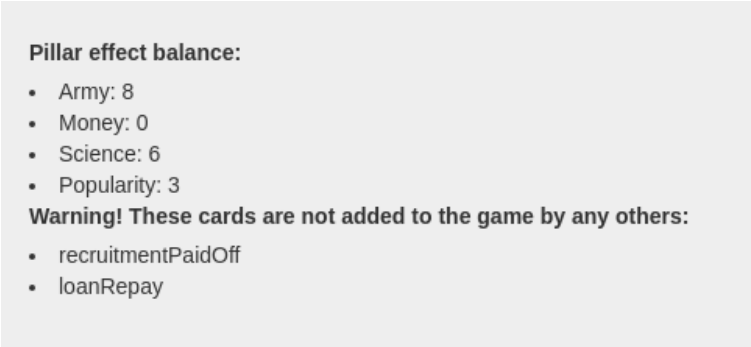

Figure 6.8: Info on left panel. Note the two warnings, which appeared on removing `recruitmentPaidOff` and `loanRepay` from all 'cards added' lists

When editing a card, a live preview of its contents can be seen at the bottom of the panel, alongside buttons to cancel changes, delete, or submit the card. When editing the consequences of a response, adding and removing cards is done through a dropdown list that allows multiple cards to be selected.

Figure 6.11 shows the pillar editing view, which is very similar to the card editing view. The colour of the pillar can be entered as a hex value, which is immediately reflected in the preview below.

Clicking `Save Current` will save the updated game definition to the configured game definition database, provided it is running. Once saved, a game can immediately be played from the game UI by entering the corresponding game ID.

Visualisation

The visualisation application is of similar form to the game maker, with data selection and filtering happening in a left panel while the visualisations are updated on the right. It is possible to adjust the data used in the visualisations by filtering the pillars by value. This limits the data shown only to cards that appeared while pillar values match the specified criteria. The visualisations I chose to show are as follows:

- Accept/Reject balance

  This is a horizontal bar chart showing the proportion of players that choose one option over the other for a given set of cards. Each card has a value between -1 and 1, where -1 indicates that players choose the reject option every time, while 1 indicates accept is chosen every time.

- Total times drawn

  This is a vertical bar chart showing the total number of times each card has been drawn and responded to.

- Pillar average

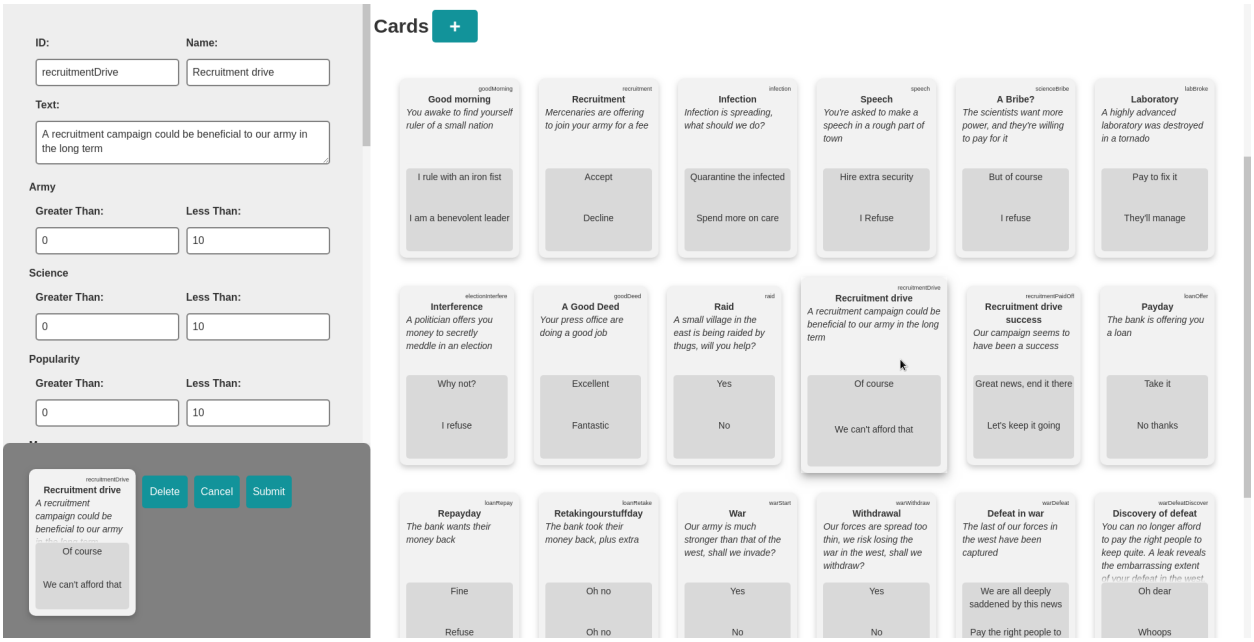

Figure 6.9: Card editing view, reached by selecting a card by clicking it.

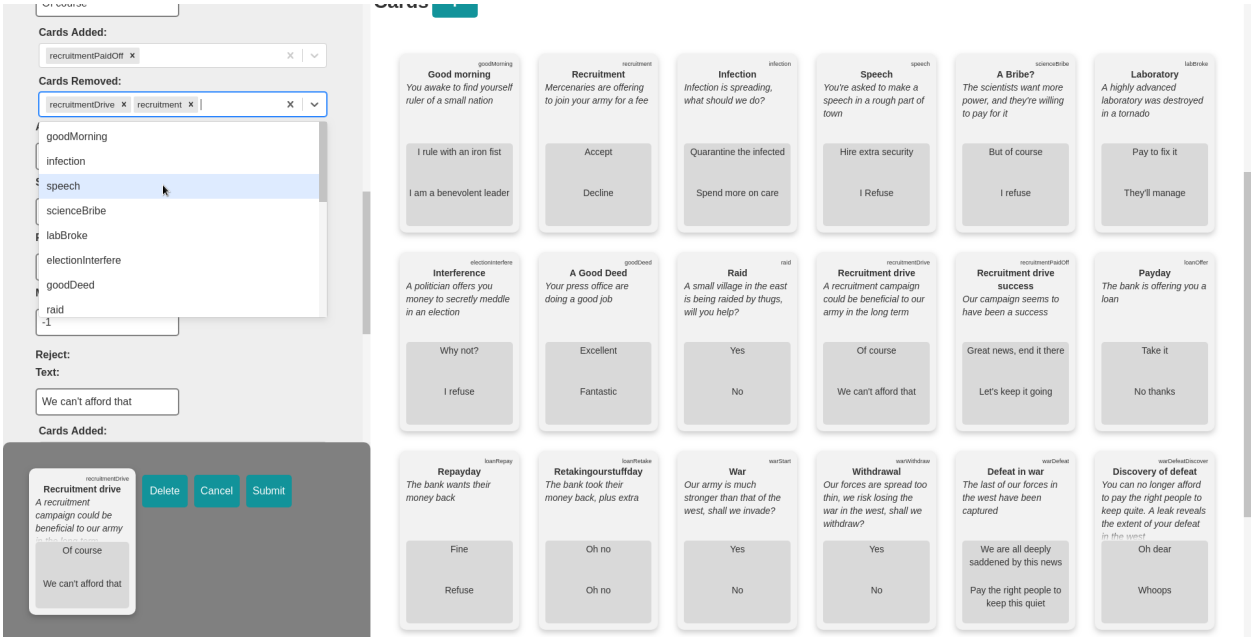

Figure 6.10: Dropdown showing all other cards that can be added or removed

This shows the average pillar levels over all turns.

The intention of these visualisations is to allow an admin user to 'play' with the data. The live updating charts allow for the user to rapidly identify relationships in the data, for example between pillar levels and accept/reject balance.

The other function offered by this page is the ability to export data to CSV. This effectively repackages the contents of the user database as a csv file, where each row represents the turn of a user. Individual games can be uniquely identified by sets of rows with the same user id and game id.

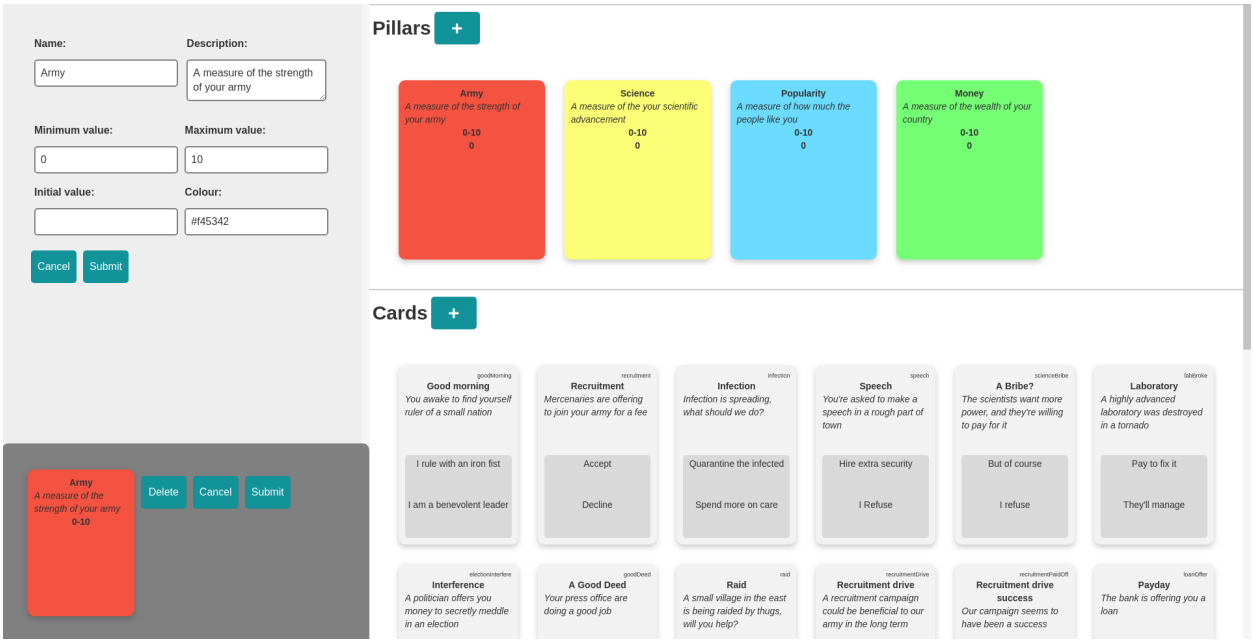

Figure 6.11: Pillar editing view

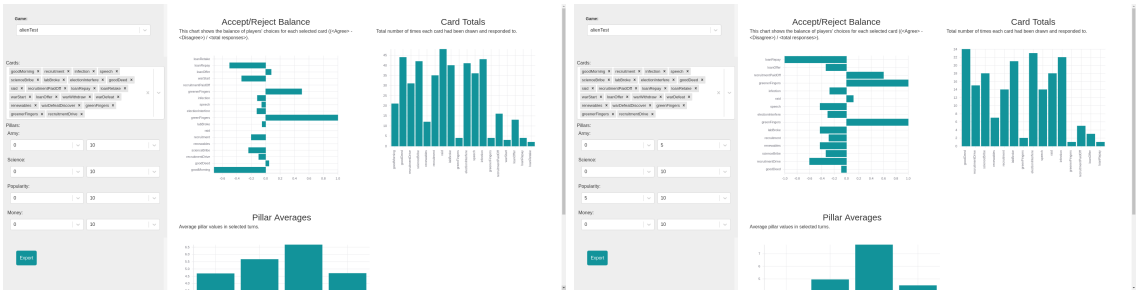

Figure 6.12: Visualisation screen effects of filtering data by cards and pillar values. This example filters responses to cards during turns where the player's `Army` was between 0 and 5, and their `Popularity` was between 5 and 10.

Implementation

7.1 Tools & Technologies

7.1.1 Node.js

It was decided early on that this application should be easily accessible on a range of devices, particularly the game interface. For this reason the system was built as a responsive web-application, running on a node server. Node[21] is an excellent framework for getting sizeable projects up and running very quickly, which I was able to do with this project. This is in part due to NPM[22], which provides access to countless packages which I was able to use, ranging from Victory[23] for visualisations, to Thunk[24] for managing asynchronous side effects.

7.1.2 TypeScript

According to Microsoft who maintain it, 'TypeScript is a typed superset of JavaScript that compiles to plain JavaScript'[25]. I decided to use TypeScript (TS) to combat some of the problems that can occur when working on a large JavaScript codebase. Being statically typed, TS prevents type errors and instantly makes code more navigable and readable. This helps both during development and maintenance, which is particularly important if others are to continue to work on this project.

7.1.3 TSLint

Alongside TypeScript, I set up TSLint[26], a linter for TypeScript. This was a helpful for development, as TSLint highlights issues that can affect the performance of the application. For example, using arrow functions in JSX results in the function being recreated each time the component renders. This can then lead to the component re-rendering due to the shallow comparison of the old and new functions. This can give rise to performance issues, which could have been troublesome for the more responsive parts of *OpDeck*, such as the visualisation page. TSLint also provides general code cleanup features such as organising imports.

React

React[27] is a UI library that simplifies the creation of responsive interface components. I knew that I wanted *OpDeck* to be highly responsive, with each section effectively being split into a single page application. React made this fairly trivial once the structure of the application was settled, as components are automatically updated as information flows down through them.

React also encourages the use of inline styles. This makes styles composable, offering improved reuse of and maintainability over plain CSS.

Redux

Redux [28] naturally complements React by providing state management. Using only React, each component stores its own state. This can lead to confusion, as components that should be reflecting the same information can fall out of sync if due care is not taken. Redux encourages moving the state of the whole application

into one top-level serializable JS object. This then acts as a 'single source of truth', which takes care of component synchronisation. Another benefit offered by Redux is the debugging tools, which take the form of a chrome extension [29]. When Redux is correctly implemented, this tool makes it possible to view and manipulate the application state, and even 'time travel' the page to a previous state, such as before a button was clicked. This made debugging very straightforward, particularly for game logic.

7.1.4 NeDB

To store both the game definitions and the gameplay data, a form of server-side persistance was necessary. I had decide here between designing and maintaining a rigid relational database, or opting for a NoSQL [30] solution. After consideration, I chose to use a document-based database. The main factor in this decision was that it was unknown what data the users of the final product would desire. While a relational database requires strict definitions of data and their types, document-based databases are much more free form, meaning properties can be added and removed with no alterations to the database required.

Specifically, I chose to use NeDB[31] - a lightweight JS database library that proved quick and easy to set up and use. The NeDB API is a subset of that of MongoDB[32], and was therefore well documented despite this being a smaller, more lightweight tool. Game definitions and user data are stored in two separate NeDB instances, to provide separation of concerns.

This database implementation does have some downsides in terms of long term maintenance - ideally further along in the project's life cycle this would probably be converted to a relational database, once the data requirements were more strictly defined. Also, regrettably NeDB does not expose encryption as part of its API. This was something I only became aware of towards the end of the project, I might have chosen a different library if this had been immediately obvious. It would still be possible to encrypt fields individually It did however suit the needs of the project, and I believe I made the right choice in getting started with NoSQL.

7.2 Data Persistance

The structure of the Redux store is an important factor to consider when beginning, as it can affect many other aspects of implementation. The store is a JavaScript object that holds the whole application state. Figures 7.2 and 7.3 show a visual representation of the store for the game site. The game logic runs client-side, so to enable this, the entire game definition is requested from the database server once the player requests to play it. Depending on the use case, this could be considered a downside as technically a player could 'cheat' the game through using console commands. This choice did however greatly simplify the API between the client and sever, as once the game begins the client only needs to send the outcome of each turn.

Access to the NeDB databases is provided through a backend server, which exposes a RESTful [33] API. This is a lightweight server that directly accesses and manipulates the databases. One database stores game definitions, while the other stores user turn data. The documentation for this API is located in a `README.md` in the top level of the `backend` folder.

7.3 Game Interface

7.3.1 Logic

Most of the game logic happens in the redux reducers. This is the part of the code where actions arrive and the payload gets processed, resulting in some change to the game state which is then propagated through the page.

Reducers should technically be pure functions, as this makes them reliably testable and allows the time travel aspect of the debug tools to work correctly. I have followed this convention for all of my reducers, except for `CHOOSE`, which is called when the player makes a choice. This is because of the shuffle function that gets called, which has a random element.

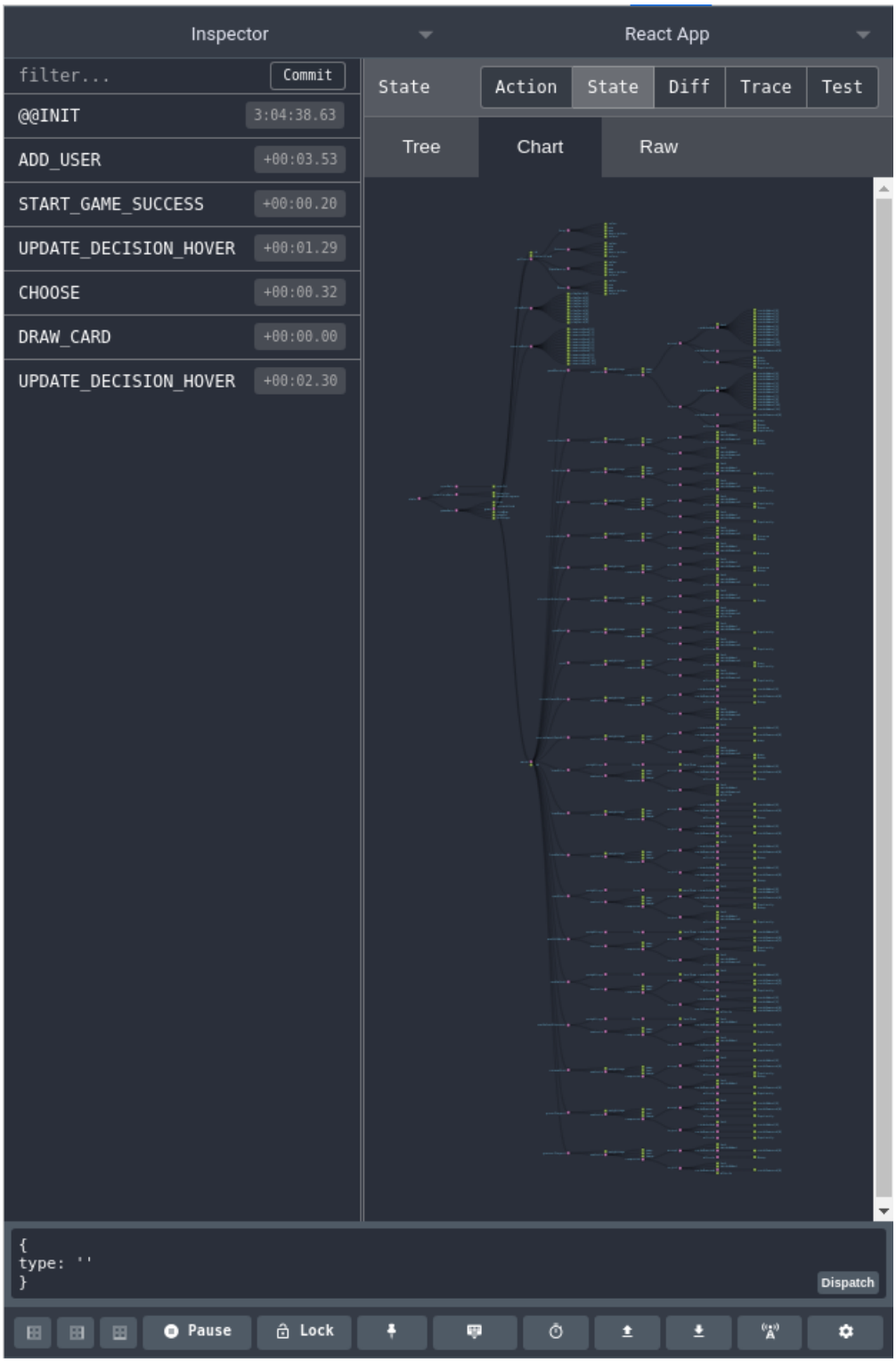

Figure 7.1: Redux DevTools [29] while in game view. This shows what actions have occurred while the page has been open, as well as visualising the application state.

The shuffle function used is an implementation of the Fisher-Yates Shuffle[34]. This was chosen to ensure that the game remained unpredictable across replays, improving replayability.

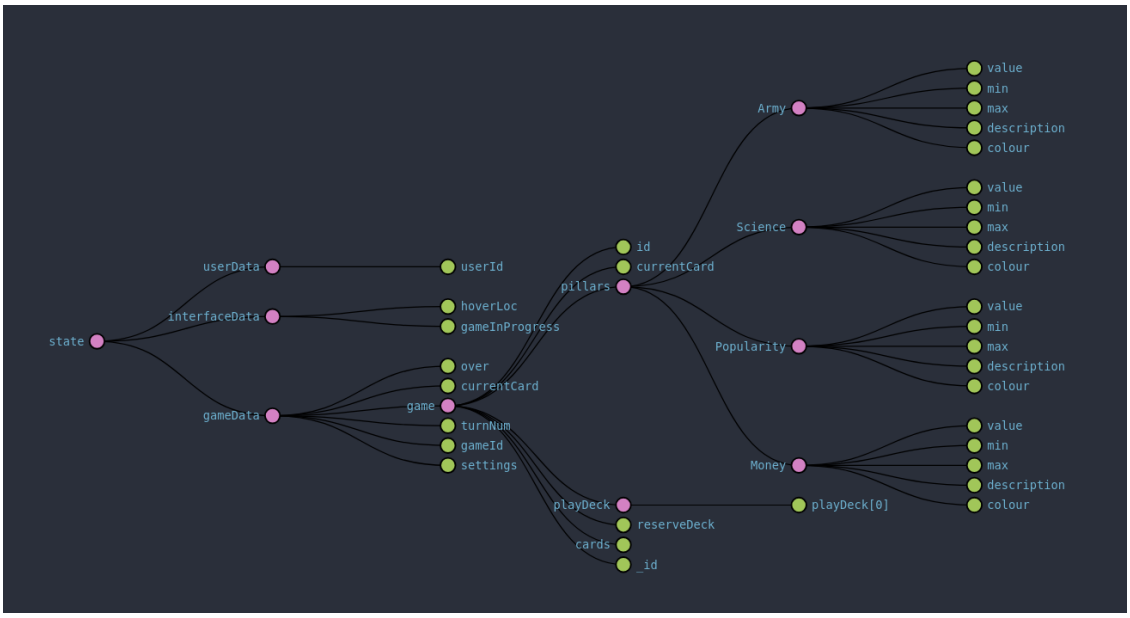

Figure 7.2: Redux store object structure. Note that the pillars (Army, Science, Popularity, Money) are game definition dependent. Cards object is omitted due to large size.

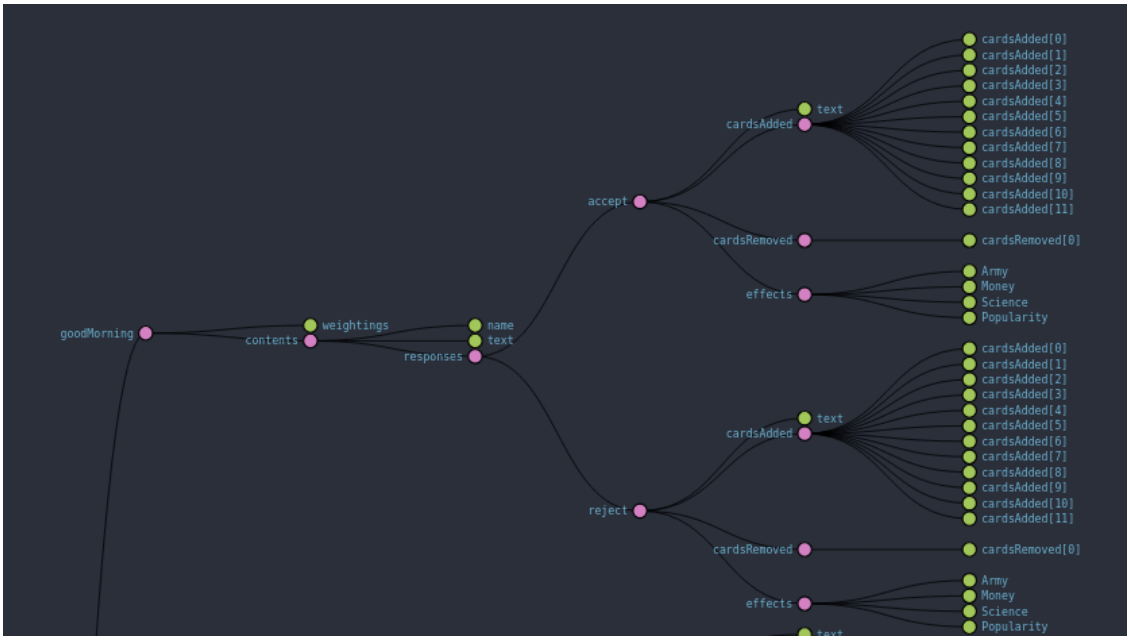

Figure 7.3: Example of a card object in the store. This is a starting card that adds many other cards to the game no matter which option is chosen.

Building the play deck is simply a case of iterating through each card that is out of play and adding those that match the requirements, as well as iterating through the previous play deck and removing those that do not.

7.3.2 Images

The colouring of images was one of the more technically involved parts of the project. In order to avoid heavily pixel processing I knew I would have to use SVGs, which can be edited with comparably insignificant computation (also having the advantage of being infinitely scalable). I then had to acquire some example images to use for my game definition. To maintain a similar art style I sourced them all from the same artist, Katerina Limpitsouni, on her site unDraw[35]. These images are open-source and usable for free and without attribution. The website has the advantage of allowing the main colour of each image downloaded to be set through a colour picker. After failing to find a solution to edit specific colours of an SVG in JavaScript, I resorted to wrapping each image in a React component, which passes through and injects the desired colour. This was not a favourable solution, as it's not possible to do this through the game maker tool. Considering this a prototypical feature (and not part of requirements), I believe the implementation is satisfactory.

Alternatively to this solution, another, less technically challenging but more time consuming possibility would be to fill the image URL property of each card, which could then be downloaded and rendered client-side.

7.3.3 Hover

When the user hovers over an answer with the mouse, the pillars affected by that response are outlined with a border, the colour of which depends on whether the effect is positive or negative. There were multiple ways in which this could have been implemented; I decided to use an enum property in the global state. Doing so makes it extensible - other effects or other hovering elements could be added easily.

7.4 Game Maker

Much of the time I spent working with CSS was on the game maker page, particularly the card and pillar views.

Within the cards, if any text overflows beyond the limit of its section, it slowly fades out towards the bottom. This indicates to the user that there is more text to read, avoiding the situation where text is cut off but this is disguised by the gap between two lines of text. This effect is achieved by adding a fixed-height, partially transparent gradient matching the background colour to the bottom of each text `div`.

Cards are displayed in a responsive grid, which was done using CSS-flexboxes. I first used a combination of the `flex-basis`, `min-width` and `max-width` properties to allow each card to expand and contract slightly beyond its initial size. These were then placed inside a `div` with `display:flex`, allowing the cards to dynamically size and arrange themselves according to the size of this parent container. Using the Emotion[36] library, I was able to inject inline styles into the `hover` attribute of the card component, which allowed me to create the card expansion effect on hovering over them. The responseive nature of the site can be seen in figure 7.4.

Removing a card or pillar removes all references to it held by other cards/pillars. This ensures that games remain consistent.

7.5 Testing

7.5.1 Unit Tests

The unit tests are all written with Jest [37] - a simplistic JavaScript testing framework.

Stemming from the structure enforced by Redux, the application logic is entirely contained within one file - `reducers.tsx`. Tests are written for the reducer helper functions that handle and manipulate the game state. I wrote these tests alongside the functions as this was easier than manually testing edge cases through the game UI.

I was not able to unit test the shuffling algorithm, due to its random nature, however I manually verified this by playing through several games and confirming that different cards appeared under the same player input.

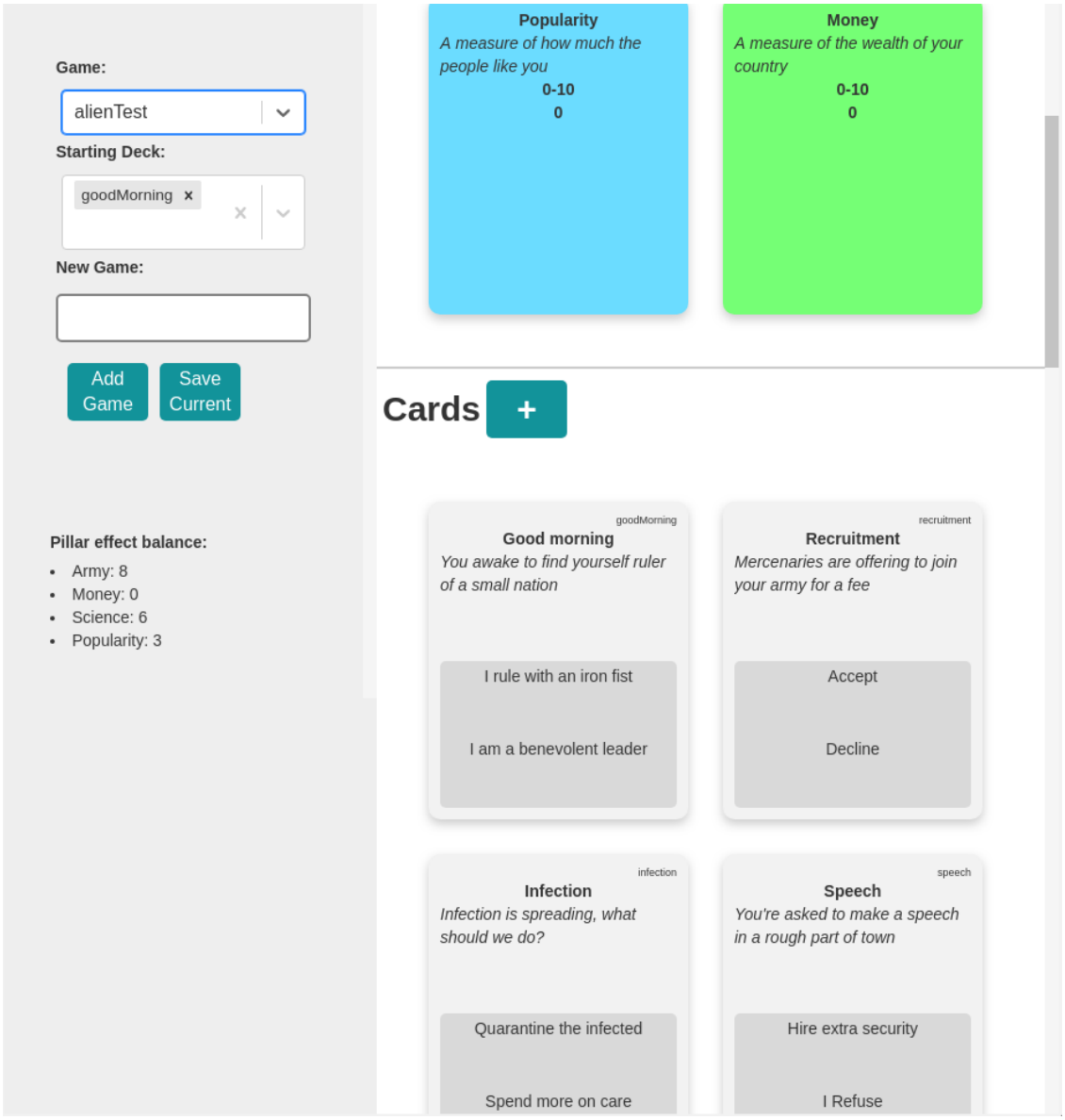

Figure 7.4: Game maker at 960x1080 - note that all infomation is still visible.

7.5.2 Manual Testing

While there exist libraries capable of testing UI, I found the *OpDeck* interface was simple enough that manual testing was both possible and sufficient. This testing consisted of myself simulating 'normal player usage', as well as intentionally attempting to break the app. This was successful in revealing several bugs, which I was able to fix, however as with most forms of testing it is impossible to be certain that no bugs remain.

Evaluation

8.1 User Evaluation

8.1.1 Data Collection

Towards the end of this project, I presented *OpDeck* to members [38] of the St Andrews Exoplanet Research Society. This provided an excellent opportunity to get feedback on the software in its current state from its potential users, as well as feature requests and inspiration for future work. To collect this information, I designed a brief artifact evaluation form which is attached as Appendix .3. This form covers the main non-functional, user facing requirements of my system. It consists of seven statements with which participants express their agreement by circling the appropriate point on a seven point Likert scale.

The items on the read as follows:

it.1 The game is entertaining

it.2 The game interface is intuitive

it.3 The game interface is attractive

it.4 The visualisation interface provides useful controls and visualisations

it.5 The game maker interface is intuitive

it.6 The game maker provides useful feedback when making a game[1]

it.7 This software would be useful in evaluating human responses to predefined scenarios

Shortcomings

The survey is not perfect, here I describe some of its shortcomings:

sc.1 I did not take sufficient measures to avoid acquiescence bias[39] when designing the form. This is the tendency for users to generally agree with statements, even if this results in two conflicting answers. Avoiding this would have required adding additional, negatively phrased questions in order to establish a baseline level of respondents' agreeableness.

sc.2 The form is brief, with only seven questions. This was a design decision, made with the aim of increasing engagement. This however, limited the amount of quantitative feedback I received.

sc.3 Participants did not have an opportunity to personally interact with the software, answers provided are based upon a presentation that lasted roughly thirty minutes. I feel that this could particularly impact the questions regarding the intuitiveness of the interface; watching another perform a task they are familiar with can make it seem easier, which may affect one's perception of how intuitive it would be to use themselves.

[1] Due to time restrictions, I was unable to sufficiently present this aspect of the game maker, therefore participants were told to ignore this question and will not include it in my analysis

sc.4 Responses pertaining to the game UI may have been swayed by the quality of the example game definition I was using. Improving this was not an item of high priority therefore it may not have provided the best demonstration of the framework.

8.1.2 Quantitative Analysis

Since they are categorical, it is not appropriate to average Likert scale data [40]. For this reason, I have visualised the results as an aggregated stacked bar chart in figure 8.1.

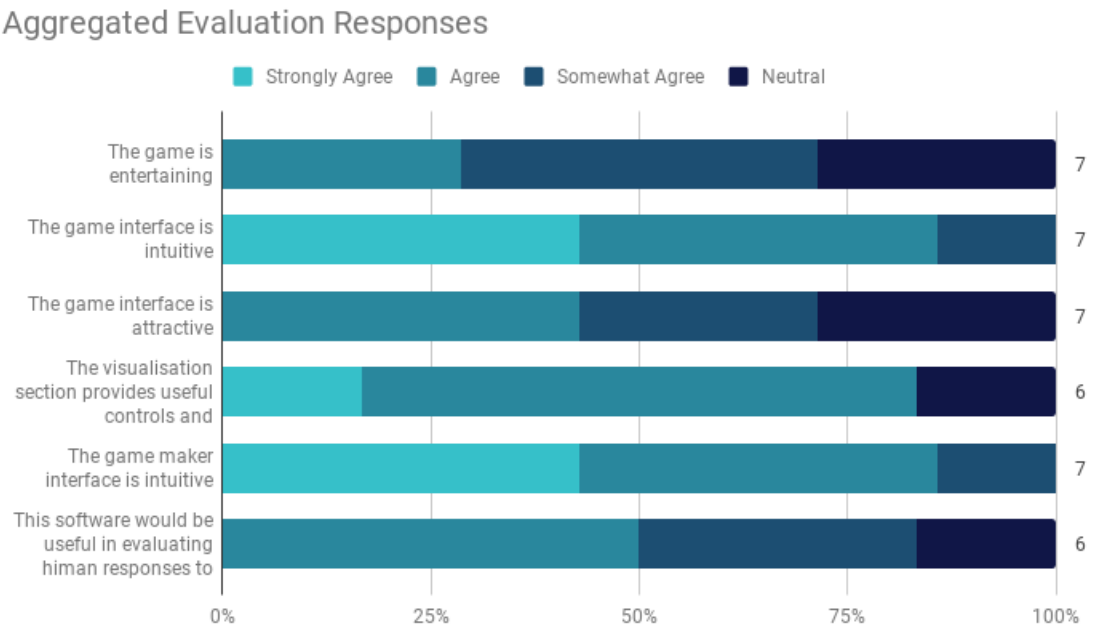

Figure 8.1: Visualisation of artifact evaluation likert items. Number to the right of each item indicates the number of responses.

From the data, it is immediately striking that none of the responses to any of the items were above 4, meaning that none of them 'disagree' to any extent. It is impossible to judge how much of this could be due to acquiescence bias; but I interpret this as a minor effect, due to the extent to which the responses are positive.

Items **it.1** (entertainment), **it.3** (attractiveness) and **it.7** (usefulness) received the lowest levels of agreement. The response to **it.1** could have been influenced by **sc.4**, or simply be a deeper comment on the game framework itself.

it.3 (attractiveness) could be influenced by the images used in this game definition (**sc.4**), however this aside, in my effort to avoid imposing a bias onto the framework, perhaps achieving a neutral response is desirable here.

it.7 (usefulness) is intended to provide an assessment of the software as a whole, being used for its intended purpose. This sees agreement, which is a strong sign that the project has been a success. The reasoning for this seeing less agreement than the other items which constitute it could be explained by general thought of doubt around the concept of a game being used to gather player's opinions that were vocally expressed by some participants.

One piece of written feedback was submitted, which stated that it was 'very intuitive to create the game'

8.1.3 Qualitative Feedback

The half hour that followed the presentation consisted of discussion around the software and its potential uses.

Feature Requests

There were several suggestions to improve various aspects of the software pipeline:

- Addition of 'sub-decks' - sets of cards which can be easily added or removed in one click, rather than having to select each card manually, possibly multiple times.
- Addition of more than one choice per card.
- Addition of an option for pillars to be 'invisible'. This would add depth to the games that could be created, allowing for hidden factors that the player can not predict, such as how they are perceived by an enemy.
- Addition of game-end condition customisation, for example two pillars must be empty before the game ends, or one must fill.
- Addition of different endings, some considered winning and others losing.

8.2 Objective Evaluation

Priority	Objective	Complete?	Comments
Primary	Devise and implement a game that presents the player with scenarios and allows them to choose from potential responses	✓	
	Devise and implement a flexible infrastructure to model and constrain scenarios and their impacts	✓	
	In collaboration with Anne Smith and Christine Helling, devise a sample set of appropriate scenarios with impacts and populate the game	✗	Due to the lateness of my first meeting with Anne and Christine, there was not enough time for them to become familiar with the framework and create test scenarios. I did however create my own example scenario.
	Devise and implement an infrastructure for capturing and recording player responses	✓	
	Implement basic visualisation of responses	✓	
Secondary	Devise and implement an admin centre to allow easy creation of new game content	✓	This became a more primary objective, as it became obvious that without the game maker tool, the software would be much less accessible, as technical experience would be required to create new games
	Carry out an experiment to assess the effectiveness of the game as a tool to assess people's real world views	✗	It became clear early on that this was a more psychological question, and that I had neither the time or knowledge required to answer this question
	Create more advanced visualisation and analysis tools	✗	After basic visualisations were complete, I prioritised the quality of the game maker tool over more advanced visualisations. The reasoning behind this was that I could not be certain I was providing useful visualisations, and in that case, external tools could be used with the exportable data.
Tertiary	Perform a wider user experiment	✓	This was achieved in the form of the user evaluation forms handed back from members of the Centre for Exoplanet Research.

8.3 Future Work

All of the feature requests raised by participants in the study would make for worthwhile future work, in addition to many other possible quality of life and aesthetic updates. Here I will elaborate on future technical work, as implementation details weren't much discussed in the presentation:

- The response system could be expanded and generalised to support different numbers of responses for different cards, rather than two for each.

- Depending on the final use case, it may be desirable to add support for collecting user demographic data, such as age and gender. This would require frontend work in making the appropriate forms, as well as increased security for communication (SSL) and encrypted storage on the backend.
- Currently the application only deploys locally, some work would be required in implementing a permentant hosting solution. Fortunately this wouldn't be too challenging as this is well documented and supported for Node.
- The visualisation tool could be endlessly expanded to become a more capable analysis suite. If more properties to were added to the user data field, there exists the possibility of filtering by age bracket and various other factors.
- The game is not currently playabe through any channels other than the site itself. Future work could consist of migrating to a more portable technology, which could be shared and played directly through other channels, such as social media.
- Deeper analysis and verification could take place in the game maker. For example, detecting cards that will never be playable given the pillar consequences of choices that must precede them.

Conclusion

Comparing my project to Datagame [10] and Qualifio [12], I find that *OpDeck* holds up well. I believe the game I have implemented is at least as engaging as the games available on these other platforms. *OpDeck* and Datagame offer similar game creation capabilities, and while I cannot speak for the analysis tools provided by these other services, I believe the functionality provided by *OpDeck*, combined with its capability to export data, is sufficient. As previously stated, further research is required to evaluate the accuracy with which this tool can infer player's opinions. Despite this, *OpDeck* has achieved or surpassed the majority of the original objectives, and I feel confident that it could provide the grounds for further research into the world of gamification and opinion gathering.

Bibliography

[1] Centre for Exoplanet Science. *St Andrews Centre for Exoplanet Science*. 2019. URL: https://www.st-andrews.ac.uk/exoplanets/ (visited on 03/17/2019).

[2] Douglas Vakoch and Yuh-shiow Lee. "Reactions to receipt of a message from extraterrestrial intelligence: a cross-cultural empirical study". In: *Acta Astronautica* 46 (June 2000), pp. 737–744. DOI: 10.1016/S0094-5765(00)00041-2.

[3] Jung Yul Kwon et al. "How Will We React to the Discovery of Extraterrestrial Life?" In: *Frontiers in psychology* (). DOI: 10.3389/fpsyg.2017.02308. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5767786/.

[4] A A Harrison. "Fear, pandemonium, equanimity and delight: human responses to extra-terrestrial life". In: (). DOI: 10.3389/fpsyg.2017.02308. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5767786/.

[5] J Looyestyn et al. "Does gamification increase engagement with online programs? A systematic review". In: (2017). URL: https://www.ncbi.nlm.nih.gov/pubmed/28362821.

[6] Á. Tóth and S. Tóvölgyi. "The introduction of gamification: A review paper about the applied gamification in the smartphone applications". In: *2016 7th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*. Oct. 2016, pp. 000213–000218. DOI: 10.1109/CogInfoCom.2016.7804551.

[7] D. Sleep and J. Puleston. "The Game Experiments: Researching how gaming techniques can be used to improve the quality of feedback from online research". In: *ESOMAR Congress 2011* (2011). URL: http://www.websm.org/db/12/16748/Web%20Survey%20Bibliography/The_Game_Experiments_Researching_how_gaming_techniques_can_be_used_to_improve_the_quality_of_feedback_from_online_research/.

[8] Johannes Harms et al. "Gamification of Online Surveys: Design Process, Case Study, and Evaluation". In: *Human-Computer Interaction – INTERACT 2015*. Cham: Springer International Publishing, 2015, pp. 219–236. ISBN: 978-3-319-22701-6. DOI: 10.1007/978-3-319-22701-6_16.

[9] Briana Brownell, Jared Cechanowicz, and Carl Gutwin. "Gamification of Survey Research: Empirical Results from Gamifying a Conjoint Experiment". In: Oct. 2015, pp. 569–591. ISBN: 978-3-319-10207-8. DOI: 10.1007/978-3-319-10208-5_29.

[10] Datagame. *Datagame*. 2017. URL: http://datagame.io/ (visited on 03/26/2019).

[11] Facebook. *Facebook*. 2004. URL: https://www.facebook.com (visited on 03/26/2019).

[12] Qualifio. *Qualifio*. 2011. URL: https://qualifio.com/ (visited on 03/26/2019).

[13] DevolverDigital. *Reigns*. 2019. URL: https://play.google.com/store/apps/details?id=com.devolver.reigns&hl=en_GB (visited on 03/18/2019).

[14] Ward Cunningham. *Agile Manifesto*. 2019. URL: https://agilemanifesto.org/ (visited on 03/29/2019).

[15] Ward Cunningham. *Principles behind the Agile Manifesto*. 2019. URL: https://agilemanifesto.org/iso/en/principles.html (visited on 03/29/2019).

[16] Git. *Git*. 2019. URL: https://git-scm.com/ (visited on 03/23/2019).

[17] GitHub. *GitHub*. 2019. URL: https://github.com/ (visited on 03/23/2019).

[18] Trello. *Trello*. 2019. URL: https://trello.com/en (visited on 03/23/2019).

[19] Atlassian. *What is Kanban?* 2015. URL: https://www.atlassian.com/agile/kanban.

[20] Ryan C. *Feedback in interface design*. 2011. URL: http://www.ryandc.co.uk/feedback-in-interface-design/ (visited on 04/10/2019).

[21] Linux Foundation. *node*. 2019. URL: https://nodejs.org/en/ (visited on 03/24/2019).

[22] Inc. npm. *NPM*. 2019. URL: https://www.npmjs.com/ (visited on 03/24/2019).

[23] Formidable. *Victory*. 2019. URL: https://formidable.com/open-source/victory/ (visited on 03/24/2019).

[24] reduxjs. *Redux-Thunk*. 2019. URL: https://github.com/reduxjs/redux-thunk (visited on 03/24/2019).

[25] Microsoft. *Typescript*. 2019. URL: https://www.typescriptlang.org/ (visited on 03/24/2019).

[26] Palantir. *TSLint*. 2019. URL: https://palantir.github.io/tslint/ (visited on 03/29/2019).

[27] Facebook. *React*. 2019. URL: https://reactjs.org/ (visited on 03/24/2019).

[28] reduxjs. *Redux*. 2019. URL: https://redux.js.org/ (visited on 03/24/2019).

[29] remotedevio. *Redux DevTools*. 2019. URL: https://chrome.google.com/webstore/detail/redux-devtools/lmhkpmbekcpmknklioeibfkpmmfibljd?hl=en (visited on 03/24/2019).

[30] mongoDB. *NoSQL Databases Explained*. 2019. URL: https://www.mongodb.com/nosql-explained (visited on 03/24/2019).

[31] Louis Chatriot. *NeDB*. 2019. URL: https://github.com/louischatriot/nedbz (visited on 03/24/2019).

[32] mongoDB. *MongoDB*. 2019. URL: https://www.mongodb.com (visited on 03/24/2019).

[33] Shirgill Farhan. *What exactly is RESTful programming?* 2015. URL: https://stackoverflow.com/a/29648972.

[34] Wilipedia. *Fisher-Yates Shuffle*. 2019. URL: https://en.wikipedia.org/wiki/Fisher%E2%80%93Yates_shuffle#The_modern_algorithm (visited on 03/28/2019).

[35] Katerina Limpitsouni. *Undraw*. 2018. URL: https://undraw.co/illustrations (visited on 03/28/2019).

[36] emotion-js. *emotion*. 2019. URL: https://github.com/emotion-js/emotion (visited on 03/29/2019).

[37] Jest. *Jest*. 2019. URL: https://jestjs.io/en/ (visited on 03/29/2019).

[38] Centre for Exoplanet Science. *Members*. 2019. URL: https://www.st-andrews.ac.uk/exoplanets/members.html (visited on 03/30/2019).

[39] Paul J. Lavrakas. "Acquiescence Response Bias". In: *Encyclopedia of Survey Research Methods* (2008). DOI: 10.4135/9781412963947.

[40] Dwight Barry. *Do not use averages witht Likert scale data*. 2019. URL: https://bookdown.org/Rmadillo/likert/ (visited on 03/30/2019).

[41] Juho Hamari, Jonna Koivisto, and Harri Sarsa. "Does Gamification Work? — A Literature Review of Empirical Studies on Gamification". In: Jan. 2014. DOI: 10.1109/HICSS.2014.377.

[42] Francois Alliot. *Reigns*. 2019. URL: http://reignsgame.com/reigns/ (visited on 03/18/2019).

[43] L. Dorcec et al. "Exploring Willingness to Pay for Electric Vehicle Charging with Gamified Survey". In: *2018 3rd International Conference on Smart and Sustainable Technologies (SpliTech)*. June 2018, pp. 1–8.

[44] Raed S. Alsawaier. "The effect of gamification on motivation and engagement". In: *International Journal of Information and Learning Technology* 35.1 (2018), pp. 56–79. DOI: 10.1108/IJILT-02-2017-0009. URL: https://www.emeraldinsight.com/doi/abs/10.1108/IJILT-02-2017-0009.

Appendices

.1 User Manual

.1.1 Start the backend

This step is required for the frontend (game) or admin sections to run correctly.

1. Download *OpDeck* from MMS
2. Ensure that Node and NPM are installed on you machine
3. Navigate to the project root directory
4. `cd backend`
5. `npm install`
6. `node server.js`

Once this server is running, admin and game applications can be run simultaneously

.1.2 Start the game

These steps should be performed in a new terminal window, while the backend server is running.

1. Navigate to the project root directory
2. `cd ../frontend`
3. `npm install`
4. `npm start`
5. Navigate to `localhost:3001`

.1.3 Start the admin tools

These steps should be performed in a new terminal window, while the backend server is running.

Note that no user data is uploaded in the submission, so the game must be played before the visualisation app is populated.

1. Navigate to the project root directory
2. `cd ../admin`
3. `npm install`
4. `npm start`
5. Navigate to `localhost:3000/maker` for game maker tools
6. Navigate to `localhost:3000/data` for visualisation tools

.2 Artifact Evaluation Ethics Form

UNIVERSITY OF ST ANDREWS
TEACHING AND RESEARCH ETHICS COMMITTEE (UTREC)
SCHOOL OF COMPUTER SCIENCE
ARTIFACT EVALUATION FORM

Title of project

An online card-based game to explore human response to predefined scenarios

Name of researcher(s)

Cameron Allan

Name of supervisor

Dr Ruth Letham

Self audit has been conducted **YES** ☒ **NO** ☐

This project is covered by the ethical application CS12476

Signature Student or Researcher

Print Name

CAMERON ALLAN

Date

12/03/2019

Signature Lead Researcher or Supervisor

Print Name

RUTH LETHAM

Date

12/03/2019

.3 User Evaluation Form

Artifact Evaluation Form

Artifact title: *An online card-based game to explore human response to predefined scenarios*
Please circle the response that you feel most aligns with your own.
(1 = strongly agree, 7 = strongly disagree)

1. The game is entertaining

Strongly Agree			Neutral			Strongly Disagree
1	2	3	4	5	6	7

2. The game interface is intuitive

Strongly Agree			Neutral			Strongly Disagree
1	2	3	4	5	6	7

3. The game interface is attractive

Strongly Agree			Neutral			Strongly Disagree
1	2	3	4	5	6	7

4. The visualisation section provides useful controls and visualisations

Strongly Agree			Neutral			Strongly Disagree
1	2	3	4	5	6	7

5. The game maker interface is intuitive

Strongly Agree			Neutral			Strongly Disagree
1	2	3	4	5	6	7

6. The game maker provides useful feedback when making a game

Strongly Agree			Neutral			Strongly Disagree
1	2	3	4	5	6	7

7. This software would be useful in evaluating human responses to predefined scenarios

Strongly Agree			Neutral			Strongly Disagree
1	2	3	4	5	6	7

Please write any other comments or feedback on the reverse

Covered by ethical application CS12476