Research

Introduction:

The Research page collects game information related to the project, presents all possible game ideas from the developers, and prepares for the final form of the game. There are four subsections:

Game Formats

This page discusses some possible formats for the project, as well as formats' descriptions, examples, corns, and pros.

Related Games

This page shows some real-world examples of games related to formate for selection and discussion.

Platform

This page describes the potential techniques the development team might use in the game, including the game engine and the game framework.

Proposed gameplay design

This page describes the three proposals the development team designed for the game, with a detailed description of each proposal, and highlights the final one discussed with the client at the beginning of the page.

Dev-Ops Tools

This page discusses tools that will be utilised by the deployment team to achieve tasks such as:

- Building a consistent development environment across developers
- Continuous Integration (CI)
- Continuous Deployment (CD)

To learn more information about the subsections, please follow the links below:

Game Formats

Related Games

Platform

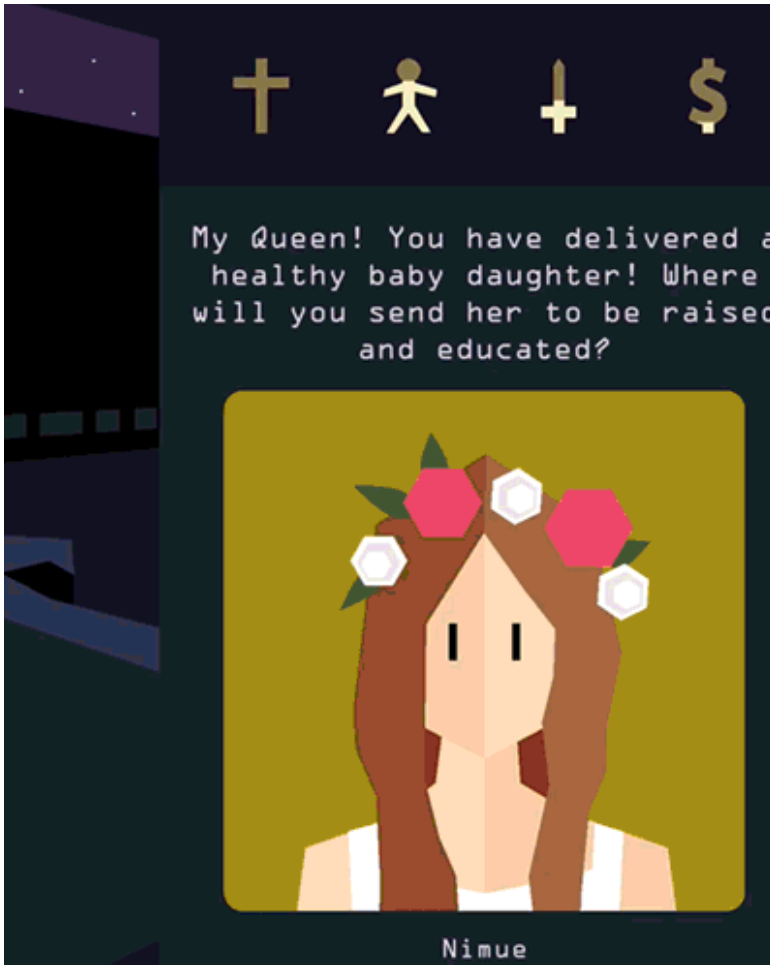
Proposed gameplay designs

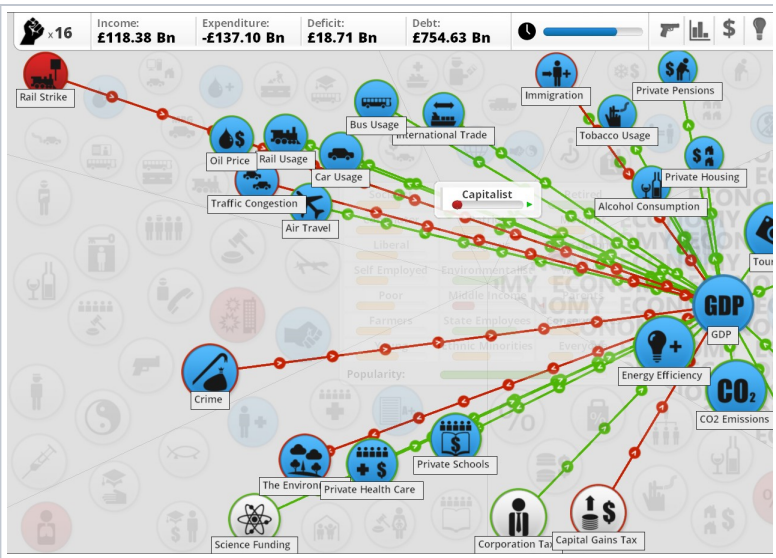
Dev-Ops Tools

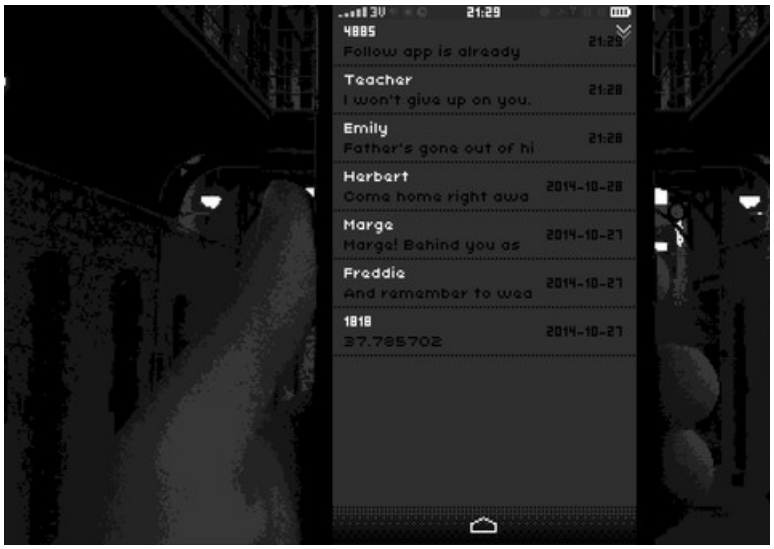
Game Formats

Format Name	Description	Advantages	Disadvantages	Examples
Card Based Game	User is presented with a choice (card or cards) and must either select a card (in the case of multiple cards) or swipe a card (in the case of one card) to make a decision.	• Allows us to focus on a deep story line as development is simplified	• May potentially make decisions seem less important • Format doesn't hide that the game is decision based	• Reigns
Interactive /Adventure	User makes decisions by clicking among choices. Choices will then change the environment	• A dynamic environment can make choices feel more meaningful • Multiple ends • More close to the real world	• Can potentially limit depth of story line as development time will need to be dedicated to visuals • Heavily depend on the story script	• Talk To strangers • Replica • Beholder • Detroit: Become Human
Turn-Based Strategy	Players take turns playing (Making choices). Strategy games tend to have more complicated metrics systems	• Users / Teachers may be able to better quantify the 'results' after a game is played due to this format's tendency to have more metrics	• Learning curve for users to play game compared to other formats • May take a long time for each round	• Democracy 3 • Sid Meier's Civilization VI • Chess • Werewolf
SIM (Simulation Game) **building or managing**	The player will play a role in an aspect of reality and control the development.	• Combine the role play and strategy • Engage players more	• No specific ending and rewards	• SimCity
First Person Perspective	The player will chose a role in the game and promote the episods by making actions or choices. Each episode focuses on a particular character, and each choice will influence all episodes.	• Combine the role play and strategy • The player has a strong sense of substitution • Based on the player's view • The player has overall image of the game structure	• Too much flexibility may add complexity	• Detroit: Become human

Related Games

Game	Format of the game	Game description	Advantages	Disadvantages	Visual
Reigns	Card Based Game	As royalty, make decisions on the kingdom by swiping left or right on cards.	• Clean, Simple interface (players stats are represented by symbols above cards) • Simple operation • Short playing time for each round of the game • Visualized rewards and punishment	• Good UI design needed to attract players	
Talk To strangers	Interactive	**Talk to Strangers** is a short game of everyday survival. Survive door to door conversations ranging from bizarre to mundane (in this order) while trying to sell products to **strangers** **Source:** https://store.steampowered.com/app/963280/Talk_to_Strangers/	• Pixel art is simplified game development • Simple operation • Could have multiple ends.	• Good UI design needed to attract players • Good script needed • May take a long time for multiple users to make decisions turn by turn	

Democracy 3	**Turn-Based Strategy**	In **Democracy 3**you are a country leader who just won the election, and you need to take a series of measures to make sure your people live good lives so that you could win in the next election.	• see the influences of decisions directly • Visualized rewards and punishment	• Complex User Interface • Long time for each round • Complex play rules • Complex decision tree • Limited outcomes • Single player	
Sid Meier's Civilization VI	**Turn-Based Strategy**	You are a country leader, and your aim is to build your empire and make it become the strongest country in the whole world.	• Multiple players	• Long time for each round • 2 outcomes: Succeed or Fail • Complex play rules	

Replica	**Interactive** **First Person Perspective** **Role Play** **Detective**	You are given a cellphone of an unknow owner. You must look for evidences of terrorism by hacking into the cellphone owner's account, under governmental coercion by inspecting the cellphone usage history and social media activity records (Stream game description). Otherside you will be accused of terrorism.	• Multiple ends • Simple interface • Simple play rules • Real situation stimulation (experience moral dilemma when making choices)	• Single player • Good UI design needed • Good story script needed • No visualized rewards or punishment	  https://steamcdn-a.akamaihd.net/steam/apps/256668436/movie480.webm?t=1470670633
Beholder (1 & 2)	**Interactive** **Third Person Perspective** **Role Play** **Strategy**	You're a Stat-installed Landlord in a totalitarian State. You must spy on tenants, peep, eavesdrop and profile! You must report on anyone capable of plotting subversion against the State. You MUST! But WILL you? (Stream game description) You will experience moral dilemma when you spy other's life. You have to make choice whether to offer a hand or not when other face problems in the risk of your own life.	• Multiple ends • Visualized rewards and punishment • Simple play rules • Real situation stimulation and have time limited for some actions • Visualized rewards and punishment	• Single player • Require long time for each round • Good story script needed	 https://steamcdn-a.akamaihd.net/steam/apps/256672664/movie480.webm?t=1476703328

Werewolf	**Card game** **Turn-Based Strategy**	**Werewolf** is a multiplayer game. Use special abilities to uncover the roles of other players.	• Simple UI	• Multiplayer game interaction (requires sharing data and parallel interaction problems during turn-based games)	
SimCity (2013)	**SIM (Simulation Game)** **Role Play**	Player(s) will act as the leader(s) of a city. They can make the rank of the city grow by making decisions of policy, culture, transport, industry and dealing with the crisis	• Managing a city is close to managing a company. So we can reference it	• Need complicated design of situations and issues in the game. • Good UI needed	
Detroit: Become human	**First Person Perspective**	*Detroit*: *Become Human* is an adventure game played from a *third-person* view. There are multiple playable characters. The game is broken into epsidodes. Source: https://en.wikipedia.org/wiki/Detroit:_Become_Human Detroit 2038. Technology has evolved to a point where human like androids are everywhere. They speak, move and behave like human beings, but they are only machines serving humans.	• Episodic, the game is broken in to smaller pieces (may want to structure game like this). • Each episode focuses on a particular character, however choices made by one character can influence all (This allows simplicity by avoiding interactions between characters but also keeps the choices meaningful)	• Graphics are complex (not within time and skill constraints)	

Proposed gameplay designs

Introduction:

There are three proposals for the project, first is Individual+group-based decision making, second is Round-based group decision making and the last one is Episodic game. The finial proposal is the second one, Round-based group decision making game format.

The details of three proposals:

1. Individual + group-based decision making

- Live multi-player game, each player accessing the game with ther laptop, phone or tablet, while also sitting around a table face-to-face
- Players are assigned one of the five roles at the start of the game. Each role is assigned a particular set of goals that he/she is to fulfill by game's end.
- Player are presented with the next part of the plot, and is faced with a set of individual decisions. They also learn certain information relevant to an upcoming group decision.
- At certain times, the group is tasked with making a group decision - they are to discuss the decision face-to-face as a group, and then enter the decision on device(s). Depending on the nature of the decision, sometimes the players might all get an equal vote for the decision. Other times, some roles may have more power, or certain roles may even be exclusively given the final say (creating power assymmetry).
- Once the decision is made, the plot advances accordingly depending on the decision, and a new round starts/

This is similar to the round-based group decision making, however, players are tasked with making individual decisions on their device (and possibly evem interacting with particular players one-on-one) in addition to the group-based decisions discussed face-to-face.

This would provide a very rich, realistic experience, however may be too complex to achieve as the plot would have to be able to handle many different combinations of decisions.

2. Round-based group decision making

This model is the same as Possibility #1 but simplifed by removing individual decisions affecting the plot - instead the game is driven by rounds of group-based decisions.

In this proposal, the game is a live, multi-player game. Each player access the game using their own laptop/phone/tablet, while also sitting around a table face-to-face.

Players are assigned one of the five roles at the start of the game.

The game takes place in rounds. At the start of each round, each player is presented with the next part of the plot, and they also learn certain information relevant to an upcoming decision. The information given to each player depends on their role (creating information assymmetry between players; potential conflict!).

Then, the group is tasked with making **one** decision for the round - they are to discuss the decision face-to-face as a group, and then enter the decision on device(s). Depending on the nature of the decision, sometimes the players might all get an equal vote for the decision. Other times, some roles may have more power, or certain roles may even be exclusively given the final say (creating power assymmetry).

Once the decision is made, the plot advances accordingly depending on the decision, and a new round starts.

Although this idea is a bit more of an abstraction compared to real life, it is comparable to how a board game simplifies real life for the purpose of gameplay.

This would simplify the complexity of the game, compared to our original model where players making individual decisions, as it results in fewer combinations of decisions in the plot (compared to having 5 different roles making different decisions) - while still retaining the element of interactivity and group dynamics.

Example round:

- **Context:** (to be incorporated into each player's plot and information):
    - Shared context: Everyone is working hard to drive the development of the 737 MAX, in time to fulfill the first order from Lion Air, which will be a huge win for everyone. The work looks to be on track.
    - **Boeing executive:** Lion Air's order will be a huge financial gain. Huge pressure to get everyone working hard and meet the deadline; otherwise huge $ loss and face the Board
    - **Aeronautical Engineer**: also desperately needs to meet deadline for their own KPIs. But has identified a potentially critical issue regarding the reliance on one single AoA sensor.
    - **Software Developer:** wants to maintain a good pace on the software development; doesn't want some issue to disrupt their progress and cause everyone to pull long hours
    - **Pilot:** has info to share on what an AoA sensor does (+ it isn't normally catastrophic if it fails)
    - **FAA:** also has some info on why an AoA sensor might be important, past cases of it failing
- **Group discussion + decision:**
    - Should the Boeing Executive halt development in order to investigate the AoA sensor further?

- For this decision: all players are instructed to vote, however the **Boeing Executive** has the ultimate say (while taking input from the team). However the development can also be stopped, if both **Aeronautical Engineer** and **Software Developer** agree to veto the Executive's decision

3. Episodic game

- Game is played on either a single or multiple devices (depending on number of players)
- Players are not assigned to characters, instead they are assigned goals (increase company revenue, increase safety, etc)
- All players vote on all descisions
- Game is structures in an Episodic way. We include particular key events leading up to the Boeing 737 maxi incidents. Consequences from past episodes will affect future ones
- Focus on one character at a time and each character will have an ending (not necessarily at the same time).

Example Episodes:

- **Boeing Executive** finding out about new generations of planes being developed by a competitor
- **Aeronautical Engineer** designing plane
- **Software Developer** developing flight automation software
- **Airline Pilot** goes to a flight simulator to learn about the new plane
- **Airline Pilot** flying the new plane
- **FAA Official** investigating crash: (this would be an example of an episode that is dependent on what occurs in previous episodes)

Example

Assumptions:

1. Built with the structure of the game "Talk to strangers"

This means the storyline is first person and conversational based between one or many NPCs and the player. Of course this can be changed.

Conventions

Characters

NPC (Non playable characters): represented by *italicised text*

Player: Represented by **bold text**

- An example of the player dialog is a situation where the player has 1 choice like this situation

Types of choices

Impactful: represented by **bold underlined**

- leads to unlocking of future choices, influences ending

unimpactful: represented by underlined text

- may change some dialog but doesn't change future choices or ending
- These essentially give the player a sense of complexity without actually making the story complex

An Impactful choice will be denoted as follows

Choice <name_of_choice>: "Choice 1"(choice_value) OR "Choice 2" (choice_value) OR ...

Example an Impactful choice

Choice (make_storyline_for_game): "Yeah sure, how bad can it be" (TRUE) OR "I'm a software developer, why am I doing this" (FALSE)

To simplify I believe we should stick to binary (two) choices, however we can allow for many choices with this convention by using values other than TRUE or FALSE

Conditional Statement

Used to represent changes based on previous choices

IF (condition1 == value OR condition2 == value2) AND (condition3 == value3):

THEN

<Something here>

ELSE

<Something here>

Note: ELSE statement is optional

Example:

IF software_students_make_storyline_for_game == TRUE:

THEN

client: Why did I enlist software engineering students to write a story

ELSE

client: This is a good story

Endings

- In the episodes we will reference Endings by ID **<< Ending_ID >>**
- We can also concatenate endings like this **<< Ending_ID1 >>,<< Ending_ID2 >>**,...
  - **Example: Pilot dies, software engineer goes to jail, CEO is fired**

- We will then refer to the Ids with a table
- Endings can be placed at the **end of the episode** OR **in a conditional statement**

Episode: Training day

You are a pilot for <name of airline> and are learning about the new mk8 jumbo jet.

Instructor: Hi, <name of pilot> I trust you read the training briefing for the this new mk8 jet

Choice admit_to_not_reading_manual: “I haven’t had the time” (TRUE) OR “Yes I have” (FALSE)

IF admit_to_not_reading_manual == TRUE:

THEN

Instructor: No worries, this plane is very similar to the mk7, with a couple of tweaks to make the pilot's life easier.

ELSE

Instructor: That's great, since I have many pilots to train today I will skim over the details.

Instructor: <more generic dialog with unimpactful choices>

IF company_adds_autopilot_changes_documentation* == TRUE

THEN

Instructor: Here are some important changes I need to mention. Due to changes in the plane's aerodynamics systems a new autopilot system has been installed. This system makes the system behave as the MK7 jet so there should be no noticeable changes when flying. However, if you need to you can disable the feature by pressing this button (points to button)

Walking out of the session you bump into an old work colleague you have lost contact with.

<Some dialogue with unimpactful choices>

Work colleague: How about we go grab a drink for old time's sake?

Pilot: Thanks but I better not, I have to being flying one those things tomorrow

Work colleague: ahh, you know how it is, these things practically fly themselves. Don’t worry it’ll still be 8 hours from bottle to throttle.

Choice go_for_drinks: “go-to the bar” (TRUE) OR “go home” (FALSE)

*__note__: the **company_adds_autopilot_changes_documentation** would be a choice made in an earlier episode whether or not to inform pilots of changes to the autopilot system.

Platform

Introduction:

This page aims to discuss the related technique, such as language for backend and frontend, for game design and their advantages and disadvantages.

1. Use Engines: Game engines can help us to render so that we can concentrate on the logic and design parts.

Engines	Language Skill	Price	Pros	Cons	Libraries	Example	Link
Construct 3	No language skill needed; operate with GUI	free	• Powerful event-based system • Easy to operate				
GameMaker Studio 2	Dragging (no language skill needed) OR C	free for development paid for publishing	• Extensive integration toolkit	• paid for publishing			
Unity	C#	free	• Full-featured • Much information and tutorials • Extensible (Customizable)	• Unfriendly to a beginner • More resource needed (CPU, GPU...)			
Godot	GDScript (similar to python) OR C++ (can be imported as GDScript)	free	• 3D and 2D, better in 2D • Multi-platform • Visual editor • Lightweight • Easier animation and scene system	• less tutorialdemo plugin template	asset lib (Godot Asset Library)	• Spooky Ghost Dotosts Dot Com (Steam) • Grimante (Steam)	
Unreal Engine 4 (UE4)	C++ (Encapsulated via Epic, less difficult than C++)	free for studying purpose	• 3D 2D • better for multithreading • better effects	• Hard to start • More complex logic needed			
cocos2d	C++, Lua or JavaScript	free	• Lightweight and efficient cross-platform • Develop for multiple platforms: OSAndroidHTML5 PC • Open-source • 2D	• Long compilation time • Need time to learn			
Phaser	JavaScript	free	• Friendly for beginners • Opensource codabase • Keep together WebGL and Canvas components				https://phaser.io

2. Develop with pure code: Suitable for a small-scale project, it is efficient to develop.

- Including front-end, which uses HTML5css3 and JavaScript as the main language and server-end, which we can use java, python, C, C++.
- Pros:
    - There is no need to download and install the engine.
    - Totally free
    - The game can be published or played on multiple kinds of devices and OSs. e.g. App Store, integrated with a webpage, Google play
    - No compilation process when developing. Save time and lower computer configuration required
- Cons:
    - Excellent skill of front-language needed.
    - Callback function nesting
    - A huge amount of code
    - Need to consider rendering process
- Some GUI development tools can be used, for example, QT.

Front-End Tools

Introduction

Having decided our project is going to be a web application, on this page we will compare different options for front-end technology to use, and document the decision made and why.

Template engine vs framework

The first decision was whether a simpler template engine would suffice, or if a more powerful front-end framework would be needed.

Templating engines (e.g. Pug, Handlebars) would allows us to use static HTML-like templates with the content filled in based on data coming from the back-end. It would be a viable option.

However a framework (e.g. React, Angular) is capable of running the application on a single page, sending and receiving data to the back-end and updating accordingly - it would enable us to provide a much more fluid and interactive experience. Using a framework may be more challenging (especially as there is not much direct experience on the team) but would also be a good learning opportunity.

On this basis, we have decided at this stage that we will go with a framework for the front-end.

Front-end framework comparison

We looked at the more frequently used frameworks, and selected Vue and React to compare and contrast. These are two widely-used frameworks that are somewhat similar in terms of their capabilities.

Tool	Language	Description	Pros	Cons
Vue.js	Javascript	Vue.js is an open-source Model–view–viewmodel JavaScript framework for building user interfaces and single-page applications.	• Easy to learn and use for beginners • Tiny size • Virtual DOM rendering and performance • Reactive two-way data binding • single file component and readability • Concise documentation	• Lack of support for large-scale projects • Reactivity complexity
React	Javascript	React is a JavaScript library for building user interfaces. React can be used as a base in the development of single-page or mobile applications.	• Powerful framework • Virtual DOM • Open source facebook library • Widely-used, in-demand skill	• Difficult to set-up and use • Documentation is only adequate

In general, Vue would be easier to use while still capable of doing what we need for the project, given that our front-end needs are not that complex. React on the other hand is powerful and widely-used, and a good in-demand tool to learn. But it is hard to justify using React if it is much harder to use for little benefit for the project.

Conclusion

On this basis, we have decided to proceed with Vue.js as a front-end framework for our project.

Back-end Tools

For back-end technology, we narrowed our search space to two main contenders, Java and Node.js.

Technology	Description	Advantages	Disadvantages
Java	Java is an Object-Oriented, general-purpose programming language and class-based. Developers can use the principle – “write once, run anywhere” with Java. Java is portable, means a program written for any platform must run similarly on a combination of hardware and operating system.	▪ Extensive concurrency, networking and GUI support ▪ As a low-level programming language, it is extremely flexible and allows us to customise in almost any conceivable way ▪ Extensive libraries and documentation for building back-end systems with Java exist	▪ Using this means that there will be a disconnect between front-end language and back-end language, which complicates integration ▪ More difficult to build systems, since lower-level languages lack the constructs necessary to speed development
Node.js	Node JS is a runtime library and environment which is cross-platform and used for creating running Javascript applications outside the browser. It is a free and open source and utilized for creating server-side JS applications. Node JS allows developers to execute their code on the server-side. It provides a faster way to write scripts that are scalable and light.	▪ Can be utilised to write BOTH front-end and back-end code, it allows for use of the same Javascript code for both ends ▪ Server-side capabilities are extensively provided, Node.js was originally meant for back-end development ▪ An event-based model to address scalability, and rich libraries to simplify coding ▪ Much faster and much more scalable than Java	▪ Less flexibility, we can only work within the capabilities Node.js provides ▪ Heavy reliance on third-party libraries leads to reliability risk, what if a library is bugged? ▪ Weaker concurrency constructs than Java

Dev-Ops Tools

Ensuring that developers are able to use a consistent environment.

In order to reduce issues in taking code from a development to a staging or production environment, we can simulate these environments.

Tool	Description	Advantages	Disadvantages
No Tool	Instead of using a tool, we provide instructions to set up the development environment	• Developers don't need to learn a new tool	• Ability to have a consistent environment across all developers is limited even with documentation
Docker	Use Containers (essentially a lightweight virtual machine) during development, developers use the containerised services	• Developers will need to learn how to set up the software and run containers	• Allows for fine-grained control of the development environment (version of tools)
Virtual Machine	A virtual machine is an emulation of a computer system.	• Full control over the environment (more so than docker)	• Higher performance overhead • More complex to set up and use than other solutions

Continuous Integration / Continuous Deployment

Continuous Integration (CI): The practice of merging developers code into a shared mainline (shared branch) on a constant basis. This helps us to avoid integration issues when developers need to merge all their features into one product.

Continuous Deployment (CD): A software release process whereby new code is tested in an automated fashion to ensure that the changes made are suitable for a production environment. This allows us to ensure that at any time we have a stable product that can be deployed.

Upon Researching It was found that there are a plethora of CI/CD tools, To limit the scope we looked at tools that are:

1. Well known
2. Have a large community backing

Tool	Description	Advantages	Disadvantages
Bamboo	Atlassian Tool designed for CI/CD	• Strong Integration with Bitbucket (since they are developed by the same company) • Most likely freely available for the project (will have to check if Melbourne Uni provides us with the tool)	• Unsure if unimelb provides hosted bamboo (the service appears to have no recent activity)

Jenkins	Open source automation server designed for CI/CD processes	• Has a large community backing
		  • Many plugins
		  • Well Documented
			• Requires us to set up our own infrastructure for compiling and testing builds

CircleCi	CI/CD tool, Closed source.	• Has "orbs" (Preconfigured templates for integrating with many services)	• Limited feature set on the free tier
			  • Can only run 1 job at a time
			  • Has a credits system (limited number of builds per month)

BuildKite	A platform for running fast, secure, and scalable continuous integration pipelines on your own infrastructure.	• Unlimited number of builds and number of concurrent jobs on the free tier	• Requires us to set up our own infrastructure for compiling and testing builds
Travis CI	Travis CI is a hosted continuous integration service used to build and test software projects hosted at GitHub and Bitbucket.	• Easy setup (compared to other tools)	• Free only for public repositories, closed repositories have a 1-month free trial
Gitlab	GitLab is a web-based DevOps lifecycle tool that provides a Git-repository manager providing wiki, issue-tracking and continuous	• Has an integration with Bitbucket	

	integration/continuous deployment pipeline features	• Offers a free tier cloud-hosted solution	• limited to 2,000 CI pipeline minutes per month

Cloud Hosting

Service	Description	Ease of use	Flexibility	Cost/Limitations of free tier
AWS	Amazon cloud hosting services.	Offers GitHub integration through a tool called "Cloud Deploy". We would need to further investigate how we deploy the application (docker image on ec2 instance)	Extremely Flexible. AWS offers services for many technologies	AWS offers both always free and 12 month free tiers for there services. The Limitations depend on the service. For example, EC2 instances offer 750 hours free per month for 12 months. For info here
Heroku	Platform as a service. Built on top of AWS.	Yes, Has GitHub integration built-in. Heroku will automatically install dependencies and run the application.	Low flexibility. Heroku is designed to 'do one thing and do it well' as opposed to 'doing everything'. For example, there doesn't seem to be an easy way to add load balancing.	Heroku offers free 'dynos' however only on personal accounts. Therefore we need to create 1 account and share the credentials
Digital Ocean	Cloud infrastructure provider	No. Requires a third-party CI/CD tool	Very Flexible. Offers many cloud services. More info here.	Offer $50 in free credit which lasts for 12 months.
Google cloud	Cloud infrastructure provider	Yes, GCloud has a tool called "Cloud Build to achieve this".	Very Flexible. It offers many cloud services. More info here	$300 credit for 12 months. In addition, they offer free tiers for their services. More info here. Based on personal use I believe this is enough for the purposes of the project.
Mongo Atlas	MongoDB service provider	Yes, setup of the cluster is done through UI. No code required.	No. Only offers MongoDB service.	Free MongoDB cluster with limitation of 500MB.

Decisions
Development environment

If team members need help setting up their development environments we will use **Docker** to achieve this.

CI/CD

For CI/CD we will use **Gitlab** since it has a free tier cloud-hosted solution with limitations that should not impact our project.

Cloud Hosting

For Staging, we will use **Heroku** and **Mongo Atlas** due to ease of use.
For Production, we may use **AWS** since it adds flexibility. This will depend on the production infrastructure we will need.

Database

Below lists a few alternative database options that we have considered, featuring a description, some advantages and disadvantages, and a price point.

Database Type	Database	Description	Advantages	Disadvantages	Price
SQL (RDBMS)	MySQL	MySQL stores its data in tables and uses the structured query language (SQL) to access the data. MySQL uses schemas to define the database structure, requiring that all rows within a table have the same structure with values being represented by a specific data type.	• Supports features like Master-Slave Replication, Scale-Out • Query Cache for repeatedly used statements • You can easily learn and troubleshoot MySQL from different sources like blogs, white papers, and books.	• Transactions related to system catalog are not ACID compliant • Sometimes a server crash can corrupt the system catalog • Stored procedures are not cacheable • MYSQL tables which is used for the procedure or trigger are most pre-locked.	
	SQL Azure	Microsoft Azure SQL Database is a managed cloud database provided as part of Microsoft Azure.	• High Availability • Data Security • Scalability	• Requires Management • Requires Platform Expertise	
NoSQL	MongoDb	In MongoDB, data is stored in JSON-like documents that can have varied structures. To improve query speed, MongoDB can store related data together, which is accessed using the MongoDB query language. MongoDB is schema-free, allowing you to create documents without having to define the structure of the document first. These documents can be easily changed by adding or deleting fields.	• No schema • Fast	• Transactions using MongoDB are complex • In MongoDB, there is no provision for Stored Procedure or functions, so you can't implement any business logic in the database level, which you can do in any RDBMS systems.	

Testing Frameworks

JavaScript testing frameworks	Introduction	Advantage
MochaJS	Mocha is a feature-rich JavaScript test framework running on Node.js and in the browser, making asynchronous testing simple and fun. Mocha tests run serially, allowing for flexible and accurate reporting, while mapping uncaught exceptions to the correct test cases. Hosted on GitHub.	• Provides compatibility for both frontend and backend testing • NodeJS debugger is supported which makes error tracing easier • Accurate reporting • Provides support for all browsers including the headless Chrome library • Very convenient framework for the developers to write test cases
JEST	Jest is a delightful JavaScript Testing Framework with a focus on simplicity. It works with projects using: Babel, TypeScript, Node, React, Angular, Vue and more.	• Compatible with NodeJS, React, Angular, VueJS and other Babel based projects • Standard syntax with documentation support • Very fast and highly performant • Managing tests with larger objects is possible using Live Snapshots
Jasmine	Jasmine is a behavior-driven development framework for testing JavaScript code. It does not depend on any other JavaScript frameworks. It does not require a DOM. And it has a clean, obvious syntax so that you can easily write tests.	• Provides small, clean and straightforward syntax for easy testing • Does not require any Document Object Model (DOM) • Provides support for both frontend and backend tests • Ease in coding as the syntax used is very similar to a natural language • Strong documentation and community support