**Introduction**

You have been asked to specify the requirements and design for a new game under consideration by Boffin Games Ltd – Dundee’s leading developer of games which target Android devices.

The game will eventually incorporate advanced multi-player interaction but what you are being asked to design is a proof of concept demonstrator which focuses on an initial, reduced functionality version of the game. The game will target the latest version of the Android operating system (10.0) so that it can utilize the newest Android features such as gesture navigation and dark mode. The game will be developed using Google’s ‘Flutter’ development toolkit rather than Android Studio – this is because Boffin Games are shifting all of their cross-platform development to Flutter.

**Game description**

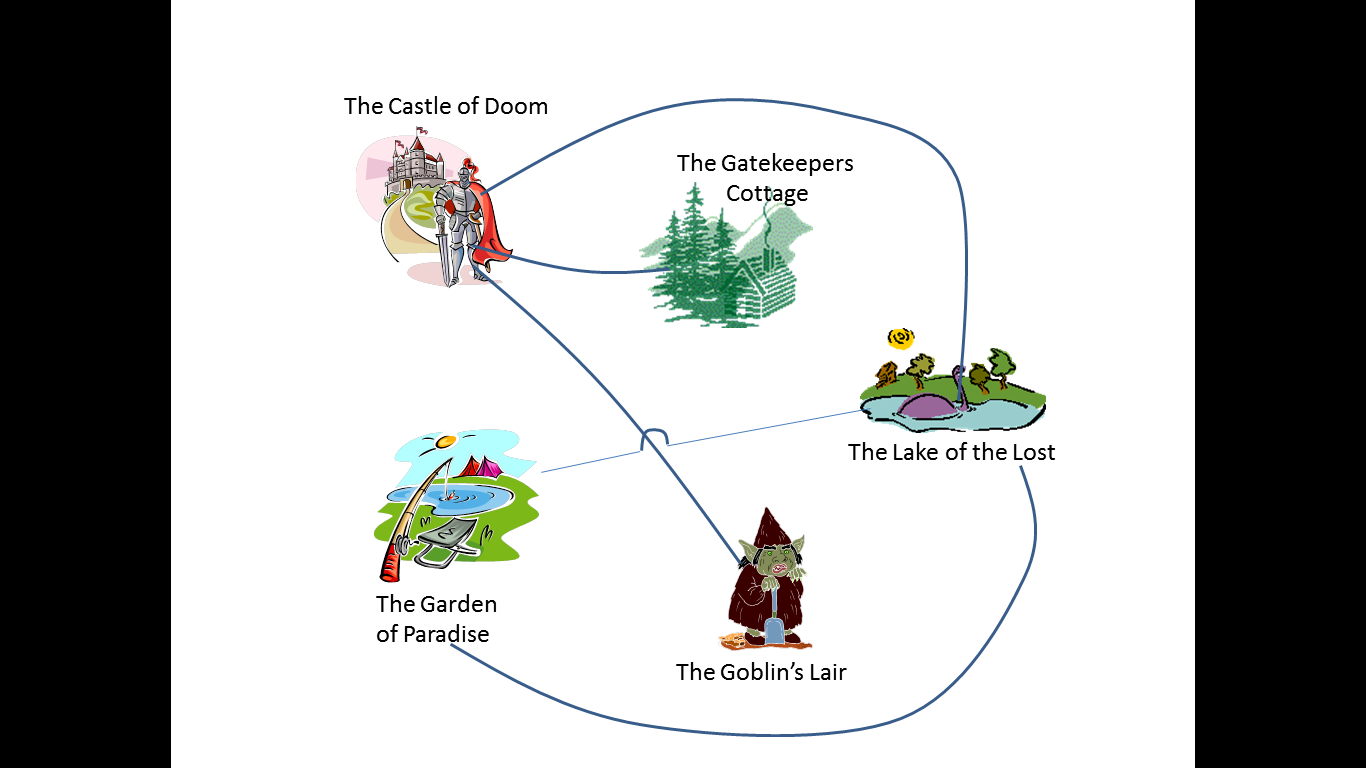
The game is called ‘JADE’. The objective of the game is to navigate a **map** of locations in order to find three virtual **“key stones”**[[1]](#footnote-0). Each keystone will be in a different location. The **player** of the game doesn’t know where the keystones are, they have to find them through a process of discovery. When all key stones have been collected, the player can make their way to a **final location** for the game which is given the inspirational name of **‘The Exit’**. Upon arriving at the Exit, the key stones will be brought together to form a **key** which will allow the player to unlock a **door** and complete the game.

The game will make use of the **location** features in Android to track the user’s location as they navigate the map. There are two common ways of obtaining a user’s location in Android: (i) the **GPS Location** Provider which is accurate but slower and consumes battery; (ii) the Network Location Provider which can be faster but dependent upon network connectivity. The ideal is to make use of a combination of these together and a ‘Fused’ Location Provider is available within the Android API for this purpose. Therefore, in the interests of preserving the user’s battery life whilst also retaining location-accuracy in the game, it is proposed that the ‘Fused’ provider should be used by the app.

Game landscape, locations, and paths

The **game** will display a map of locations or **destinations** to the user which are connected via **pathways**- see Figure 1 below. As can be seen in Figure 1, the map presents an imaginary **landscape** which contains **imaginary locations** within it (such as the Castle of Doom or the Garden’s of Paradise). Note: the landscape and the locations will change depending on the theme of the game; different **games** or **themes** can be downloaded for users to play. Figure 1 also shows that each location is connected via a **pathway**. The pathways essentially show how you can get from one location to another.

.



*Figure 1. A game landscape comprising locations to travel between*

What can’t be seen in Figure 1 is that each of the locations on the map corresponds to a **real-world location** in the user’s vicinity. But the real-world location is obscured; it is being presented as an imaginary, ‘virtual’ equivalent instead.

For example, one possible game could be made up of locations which are spread across Dundee City Centre. One location may be Costa Coffee in the Overgate centre. Another location may be Braes Bar, or the fountain in Dundee City Square, or platform 2 of the train station, or the 2nd floor of the University Tower building and so on. In the game, these real-world locations will be given alternative, **imaginary names** and **icons** instead to suit the theme of the game, e.g. Braes Bar becomes ‘The Castle of Doom’, Costa Coffee becomes the ‘The Garden of Paradise’, and so on. Each of the locations has pathways to connect them. These pathways are virtual, e.g. they don’t correspond to real-world roads or pavements. They are there just to create a link between the locations on the map and will be used later when navigating between them.

When viewing the **game map** (Figure 1), the player of the game will not initially be aware of which real world locations that the map corresponds to, e.g. the player won’t know that ‘The Garden of Paradise’ is actually ‘Costa Coffee’. It is only through playing the game that the real world locations will become apparent. The player will select a destination on the map to travel to. Then, using the **location feature**s on their device, the player will be guided towards the real world location that the selected map destination corresponds to. When the player reaches the GPS location that the game has guided them to, the real-world location will become apparent, e.g. they will realize that the ‘Garden of Paradise’ is actually ‘Costa Coffee’.

To achieve correspondence between locations on the game map and the real-world locations that they relate to, each location on the map stores **a collection of GPS coordinates** that define the **boundary** of the real world location that it represents, e.g. ‘The Castle of Doom’ will have coordinates which correspond to the location of Braes Bar in Dundee. Therefore, when the game is being played, it shall be possible to infer, by using the **GPS position** of the **player’s device**, which location in the game map they are positioned at / within. As players physically move to a real world location, their corresponding position on the game map can also be deduced, e.g. they are at location ‘X’ or somewhere in between location ‘Y’ and location ‘Z’. When a given destination is reached (i.e. it is detected that the player is within its real-world GPS boundary) then this may trigger a **game-specific event**. See details of different **location types** later.

Each *destination* on the map will have a name and a **description**. As mentioned, the **destination’s name** may be imaginary and may not correspond to the name of its real world counterpart, e.g. Costa Coffee in Dundee may be given a name of “Dragon’s Lair” in the game. Also, various graphics may be used to represent and render the destinations on the map and which, again, may not bear any resemblance to the destination’s real-world counterpart. E.g. a **graphic** of a castle may be rendered instead of an image of the Costa Coffee store. However, each destination in the game may also have a set of **descriptive hints or clues** associated with it which could assist someone in navigating to it (see **player movement** later).

Destinations on the game map will be linked to each other via one or more pathways. As noted, the pathways are virtual and may not necessarily correspond to any real pathways which exist between the real-world locations in the game. For example, there may be a pavement which one could use to move directly from the University Tower Building to Braes Bar in Dundee. However, in the game map these two locations (their imaginary equivalents) may not have a direct pathway between them. Instead one may need to move indirectly through other locations on the map to plan a route from one destination to the other during the game.

When they are playing the game, a player can choose any means of travelling to their destination, e.g. on foot or by bike, car or helicopter! The game isn’t concerned about *how* a player moves between locations. Rather, it is concerned with being able to detect where they are in relation to their intended destination and when they have arrived at their **selected destination** (by detecting that the GPS coordinates of the player’s device are within the GPS coordinates that define the boundary of the intended destination).

As will be described later, each location in the game may contain **items** that players can collect and/or will require **activities** to be completed by the player to allow them to proceed.

It is worth noting that there could be many different games to choose from. For example, there could be several Dundee-based games all of which use different locations or different themes. There will be games that map to locations in other **towns, cities, and countries throughout the world**. In the future, it is anticipated that users can design and upload their own games for others to play but, for the time being, a collection of pre-defined games will be made available via a **game server**.

Starting / joining a game

To start a game, the player must search for a game in their **current location**. A dedicated **game server** will exist for JADE which will host a repository of all known JADE games in various countries and which can be queried for information about available games. When provided with the GPS location of a player’s device, the game server will return a list of all games which are within reasonable proximity of the player’s location. For example, if someone standing in Dundee City Centre searched for a game, they may find games available on the University Campus, the Hilltown area, Broughty Ferry, and others. A **list of available games** will be provided to the player, ordered in terms of proximity and containing information about the name of the game, a brief description, and the number of players currently playing it. **NOTE: the game server software is being developed by another team of people, it is their responsibility to design the logic required to answer questions about game availability. Your system simply needs to be able to call these services and receive the data returned.**

Once a list of available games has been retrieved and displayed, the **player must provide a name** they wish to use during the game and select the game they want to play. The **game application** will then issue a request to the game server to download the actual **data for the selected game** onto the player’s device. This **data** will include: the map/landscape for the game which is pre-populated with its various locations/destinations and their contents and paths between them; a collection of **questions or puzzles** which will be used during the game (see later). The devices that are expected to be running the game are likely to have a decent level of RAM (e.g. 4GB or above). However, it is recommended that the memory resources of the game do not exceed 128MB, including the game graphics and **sound files**. This is to ensure optimum compatibility.

Having downloaded the game data, the game will commence and the player will be asked to make their way to the **starting location**. One of the locations on the game map will have been designated as the starting location by the **game designer** and this will be highlighted to the player on the game map. The game will wait until the player has physically moved to the starting location. This will be achieved by monitoring the GPS location of their device and waiting for them to move within the known boundaries of the intended destination. When they reach the **starting point**, the player will be offered the option to take a turn. At the start of a game a player will be provided with a 100% **health level** and will also be provided with a **'bag'** - this will be used during the game to hold items that the player can pick up, including key stones.

Player movement

Players move around the game using what is known as the **‘Oracle’**. The oracle is a **virtual cup** which when shaken will dispense a **set of stones** which will form the **shape of a number.** This number dictates the number of locations a player may move around the landscape.

For example, assume a player is currently positioned on the starting location. The player shakes the Oracle and receives the number ‘3’. This means the player can move to a destination which is up to three **path lengths** away from their current location. The game will automatically search pathways from the current location and highlight all of those which the player can move to. The player may then select the destination they wish to visit. Once selected, the game will then wait for the player to enter that destination by monitoring the GPS location of their device and waiting for them to move within the known boundaries of the **intended destination**. NOTE: the player can move directly to the c**hosen location**; they don’t have to visit intermediate locations along the way.

As noted previously, players won’t initially know which real-world location it is that they may be travelling towards. For example, if they select ‘The Castle of Doom’ as a destination to travel to on the map, they won’t know that it is actually Braes Bar that they will be travelling to. Therefore, the game needs to provide **navigational guidance** to the player to help them get to the **correct destination**. This will primarily be in the form of a **navigational arrow** which will be displayed to show the **direction** in which the player must travel and the **approximate distance to the destination.** The JADE game will make use of the **Google Maps web service** to assist it during this process. It can periodically send the current GPS location of the player to Google Maps along with the GPS coordinates of their selected destination in order retrieve the **navigational information** required, e.g. **the direction to travel** in (north, north-east, etc.) and the **distance to the destination**. In addition, each destination may have **descriptive hints / clues** associated with it which the player can use to help them determine where they need to go. An option to view **travel hints** will be presented to the player when they have selected the destination to travel to but it will come at a **cost**: each clue may incur a **percentage decrease in the player’s health** as a **trade-off** for being given information that helps them get to their destination more quickly.

Regarding player movement: consideration should be made for the **frame-rate** of the game. Because the game doesn’t rely on ultra-realism or **performance** in its graphics, it will be possible to play it at a lower frame rate. More generally, in terms of the **User Interface** the App must comply with **Google’s Material Design guidelines**[[2]](#footnote-1).

Once the player is detected as having reached / entered the destination that they are travelling to they may be subjected to various **virtual actions or events** depending upon the destination’s type. Each destination on the map will be classified as one of the following: an **‘intermediate’ location**; a **‘question’ location**; a **‘key stone’ location;** a **‘wild card’ location**; and **‘The Exit’ location**. For example, the destination on the game map which corresponds to Costa Coffee may have been classified as a ‘question’ location within the game. When the player is detected as entering Costa Coffee, the game will present the player with virtual actions or outcomes to perform which are befitting of a ‘question’ location (described below).

The various destination types and their intended virtual outcomes are described below.

*Intermediate Locations*

Some locations are intermediate destination types where nothing of interest happens. Intermediate locations may take the form of **navigational points** on the game map (such as bridge to cross or a cross-roads to move in different subsequent directions from). If a player ends up on an intermediate location during their turn, then nothing will occur. The **player** will simply begin their next **‘turn’** of the game in order to move to another location that they are trying to get to. a

*Question Locations*

Some locations are “question” destination types. These are locations where a player will be asked to **answer** a **question** or solve a **puzzle** before they can progress with the game. The question will be presented and the user will be able to type or select their answer. When a game of JADE is downloaded to play it will contain a **collection of pre-defined questions** which can be used in these locations. Some questions may just be **general knowledge**; some questions may relate specifically to the real world location that the player is currently situated within. For example, if the player happens to be in Costa Coffee, examples of **location-specific questions** could be: “what colour of shirt is the 3rd person from the left wearing in the poster on the Costa Coffee counter?”; “What is the last item on the menu?” etc. The rationale for this is that it helps to prove that the player is actually in the location they are supposed to be. When retrieving questions from the question collection, a **location’s id** can be specified in order to receive questions which relate specifically to that location.

If a player answers a question correctly they can use the Oracle to proceed to the next location in the game. If they answer incorrectly, they won’t be allowed to proceed until a **correct answer** has been provided. The same question will either be repeated over and over until the correct answer is provided. Or a different question entirely will be presented. The **location settings** will determine which **option** to use. When a player **answers a question incorrectly**, they will lose a percentage of their health, e.g. 10%. If a player’s health falls below 15% they will be automatically be requested to move to a **‘regeneration’ location** within the game where their health will be replenished back to 100%. The regeneration location is typically the starting location for the game.

‘Trade health for question’: in certain game locations, when a player is presented with a question that they don’t know the answer to, they may be provided with an **option** to give up a percentage of their health instead of answering the question. In this case, the player can bypass the question and continue moving through the game.

*Keystone locations*

Some locations are “key stone” destination types; they contain one of the virtual key stones that players are expected to collect during the game. Key stone locations are similar to Question locations, i.e. the player will be presented with a question. The only difference is that, if the player gets the answer correct they will win a key stone and it will be placed into their bag. The player can then proceed to the next location in the game. NOTE: players won’t know in advance where the keystone locations are. Indeed, this is true of all location types. During the game, the player simply selects a location to travel to without knowing what may happen when they get there. It is only upon entering the location that its purpose becomes clear and player is presented with a question or other task to perform.

*Wildcard locations*

Some locations are **wild card destination types**. When a player enters one of these they will find themselves subject to a virtual action or outcome which will either be beneficial or detrimental to their progress within the game. This is intended to inject and element of surprise into the game. When a wildcard destination is entered, a wildcard will be retrieved at random from a wildcard collection and presented to the player. The following wildcards will be provided:

* **“Energy boost**”: the player’s energy level will be increased by 10%
* **“Question Immunity Spell**”: the player will be awarded a question immunity spell which they can add to their bag. They can use this **spell** later to help them bypass questions that they don’t know the answer to without incurring any health penalty.
* **“Give It Up”**: the player must relinquish something from their bag such as a key stone or question immunity spell, or give up 5% of their health.

A range of additional wildcards are planned for the future such as encountering various **virtual creatures** within game locations (e.g. serpents, trolls, Discipline of Computing lecturers ☺) and being able to pick up **virtual weapons**, **shields**, or **spells** (e.g. invisibility cloaks, hypnotic trances, smoke screens). However, these will not be implemented in the current version of the game.

*“The Exit” location*

One of the locations in the game is the final destination: “The Exit”. This is the location which players will move to once they have gathered all three virtual key stones. Once the player has collected the final keystone, the Exit location will be revealed to them. When the player enters the Exit location, the game will bring the key stones together to unlock a **virtual door** into “The Exit” and inform the player that they have completed the game.

Player **messaging** and **status updates**

In the version of JADE that you are designing, players simply make their own way around the game landscape without any direct **interaction** with other players. However, player ‘messaging’ will be supported, i.e. the ability for one player to broadcast a message to any other players who may be playing the same game at the same time. **Messaging** will primarily be handled by the **JADE game server**.

At the start of a **player’s turn** (i.e. any time options are being presented to a player to select from) the player will be provided with an **option** to broadcast a message to other players. After they enter the **message** that they want to broadcast, the game application will send the **message text** to the JADE game server along with the **player’s name** and **the id of the game** they are currently playing. The game server will then handle the logic for recording the message and ensuring it is relayed to other players (the game server development team will be handling this). At the start of a player’s turn (i.e. any time options are being presented to select from), the game application can issue a **request** to the game server to retrieve any messages which have been broadcast by other players in the same game, if available. It can do so by providing the **id of the game** being played to the server. Any messages retrieved can then be displayed to the player alongside the other **game options**, e.g. “Player ‘x’ says: ‘Boo!’ “

The game publishers are also keen to explore integration with **social networking sites** in order to promote awareness of the game and to build up a large **user base**. Therefore, during a player’s turn or during specific game events an option will be provided to allow the player to post a status update to Facebook, e.g. ‘Craig is Grumpy because he lost 5% of his health’. The application will integrate via the **Facebook web API** in order to achieve this. **Integration** with additional social media sites may be considered in the future. It is expected that some **personal data** for the user will have to be stored during the game to achieve this, e.g. their location or **social media credentials**.

Game Packs

Finally, it will be possible for players to purchase and download game packs to enhance the version of the game they are playing. Game packs may contain additional map layouts for their game, different **‘skins’** for them to use, different actions that occur on the game locations, and so on. Game Packs won’t be available through the Google Play Store. Instead, these will be purchased directly from Boffin Games and will require the user to provide their **payment information** (details of a credit card) via an **in-game form**. Boffin Games have a **payment server** which handles payments and will be used to process the **credit card details** which are provided.

1. The virtual ‘key stones’ are made of the precious stone ‘Jade’ – hence the name of the game. [↑](#footnote-ref-0)
2. <https://developer.android.com/docs/quality-guidelines/core-app-quality> [↑](#footnote-ref-1)