Spell Slinger

Project Report

Student: David Darigan
Student ID: C00263218
Supervisor: Dr Joseph Kehoe

Submission: 19/04/2024

Table of Contents

Table of Figures

1. Introduction

The purpose of this report is to provide a detailed overview of the completed student project to the reader.

This document contains a description of the “SpellSlinger” augmented reality android mobile game.The reader will be briefed on the technical technologies used throughout, the final screens of the project will be displayed, the test cases used to test the backend server logic will be presented to the reader. The challenges and learning outcomes will also be explained in detail.

2. Project Description

Spell Slinger is a multiplayer android game that utilises augmented reality for an enhanced player experience. Players travel the real world while gaining ‘Skill Points’ for every new distance walked. Players may spend these skill points to draw spells from Spell Wells, which are located at real world landmarks. Players may use these spells in battle against creatures or against other players in their surrounding vicinity. All of these use cases have been met.

3. Technical Description

3.1 Jetpack Compose

Spell Slinger was developed using Jetpack Compose which is a relatively new reactive functional player interface framework for Android applications which uses Kotlin. This framework was chosen because it aligned player interface code and logical code into a single shared code space.

3.2 Firebase

Google's Firebase Realtime Database was the database utilised by this project. It provided authentication which proved useful in securing data. Google's Cloud Functions also integrate with Firebase through Firebase Functions which integrated with this project significantly.

3.3 Google Maps

The project used Google Maps Platform as the basis for the World Map screen. This allowed the project to create a styled map based on real world locations.

3.4 Bluetooth

Bluetooth enabled location-based multiplayer without breaching GDPR by pairing with only devices locally within the area before starting a player versus player battle on the server.

3.5 SceneView

SceneView is a Jetpack Compose wrapper over Google's ARCore library. It provides Spell Slinger with augmented reality features.

4. Final Screens

The Application starts at the Splash Screen which is a LoginActivity.xml file and allows players to login or sign up by invoking the Firebase Authentication UI

Fig 1 - Splash Screen

This is the premade FirebaseAuthenticationUI package that is involved from the splash screen. players have the choices to sign up or sign in with their email and password OR their google account. If successful, the LoginActivity will finish and will start the Applications MainActivity.

Fig 2 - Firebase Auth UI

The World Map screen is the first composable the player will encounter upon logging in. A composable is a function that provides Kotlin code to define the player interface of a Jetpack Compose application and manage interactions between itself and the screen state.

The World Map contains a sprite representing the player located at their approximate location in the real world.

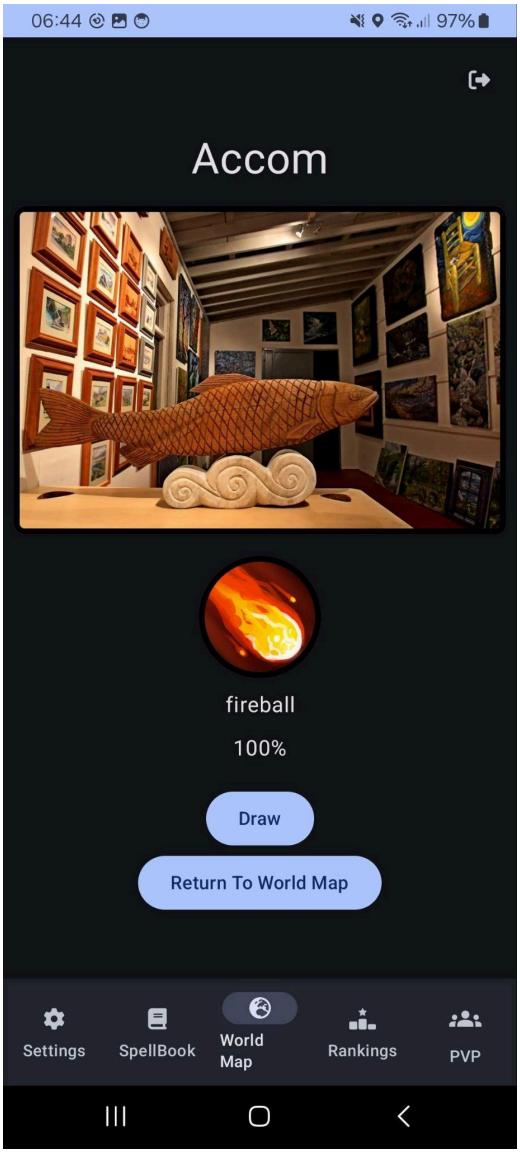
Creatures and Spell Wells are also populated at locations in the real world. The World Map observes the players location in order to get all creatures in the players geohash and surrounding geohash neighbours

This is also the introduction to the scaffold that hosts the top app bar, which includes an action to logout of the Application, the bottom app bar that contains all primary navigation options.

In the centre is the ‘content’ that displays the content of the current screen.

The player’s skill points, which are earned by walking real world distance, are displayed in the top right corner.

Fig 3 - World Map Screen

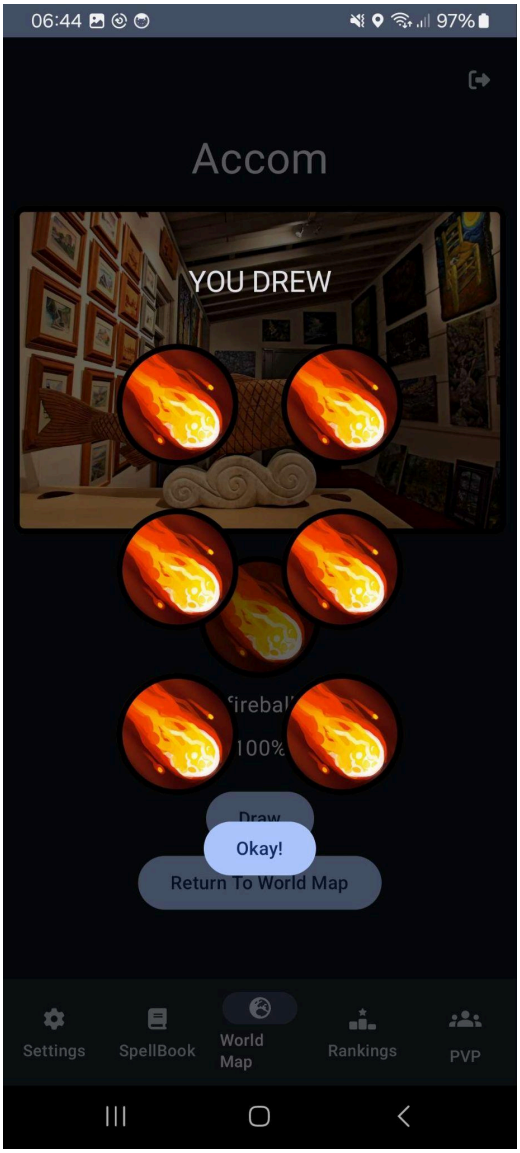


The Spell Well Screen details the title of the Spell Well being interacted with, and has an image that displays a picture from the location.

Underneath the image there is a row of spells available at the Spell Well with the chance of drawing them from this spell well.

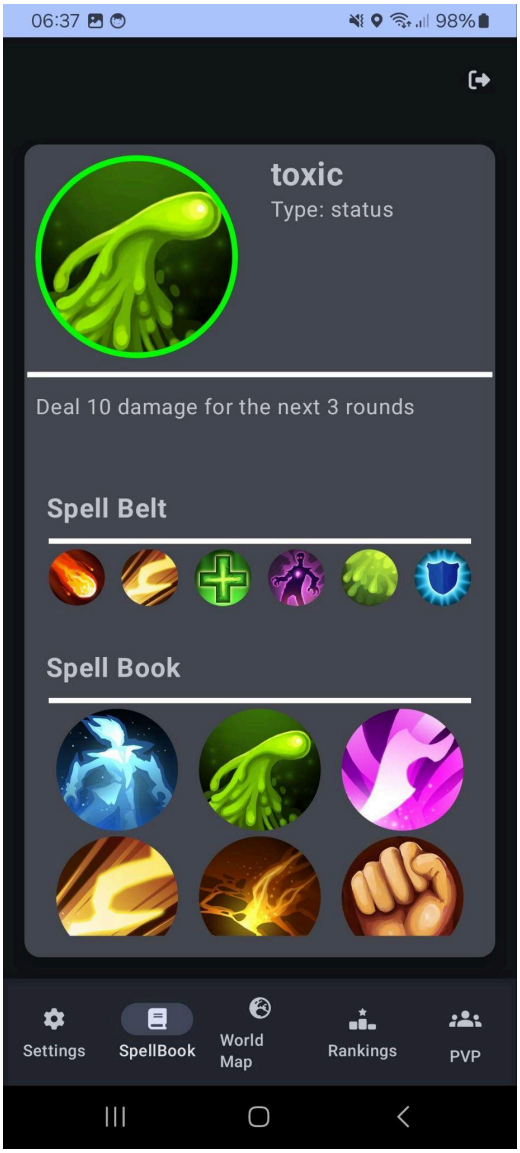
There are two options to draw from the Spell Well which will invoke a write command to the database and an option to return to the world map which will navigate the player back to the world map.

Fig 4 - SpellWell Screen



When a player draws from a spell well, their skill points will be reduced and they will draw six spells from the well, which are then displayed to the player on the client. The player can dismiss this dialog by pressing the ‘okay’ button.

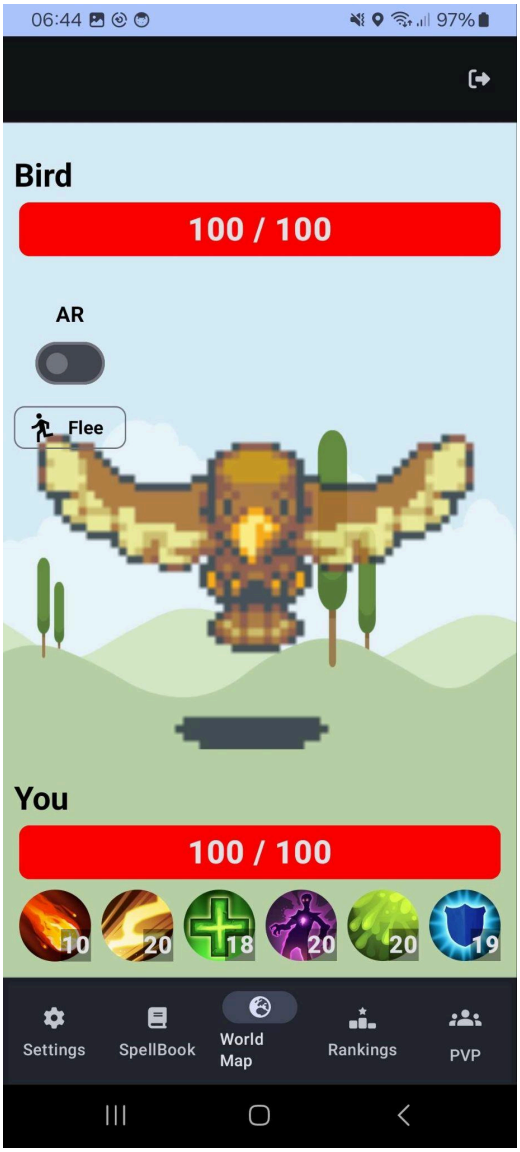
Fig 5 - Draw Results Dialog



The SpellBook displays all of the Spells the player has and which Spells are equipped to their spellbelt.

A player can see the details of the last pressed spell at the top of the page. When a player clicks on a spell not in their spellbelt and then clicks on a slot in their spellbelt, that spell will be equipped to the spellbelt. If there was already a spell in that slot, then that spell will be unequipped for the newly equipped spell.

Fig 6 - SpellBook

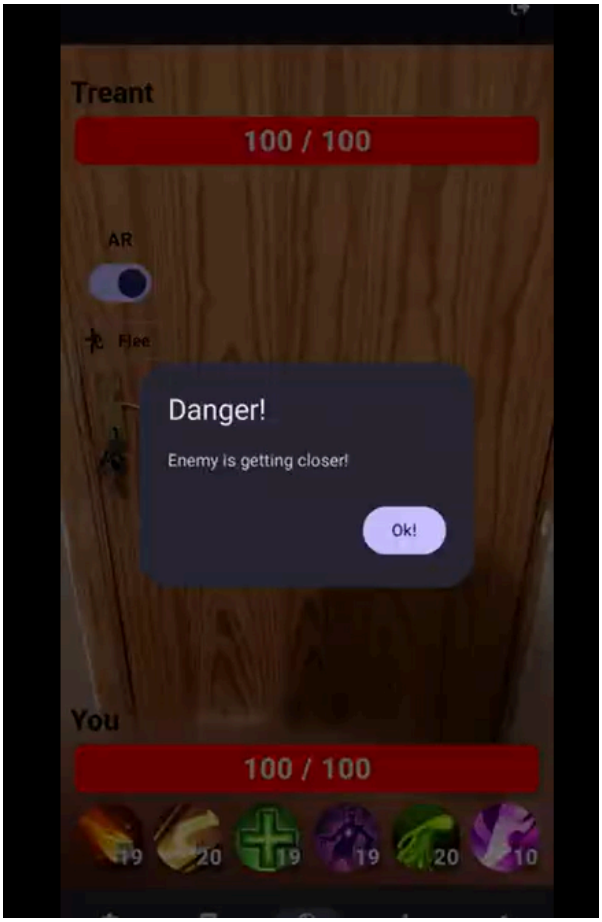


When a player clicks on a creature on the World Map, a pve (player-versus-environment) battle starts against that creature.

The creature and player will take rounds slinging spells at each other until one loses. When either loses, the battle is over. If the player lost OR won, they will be returned to the world map with full health.

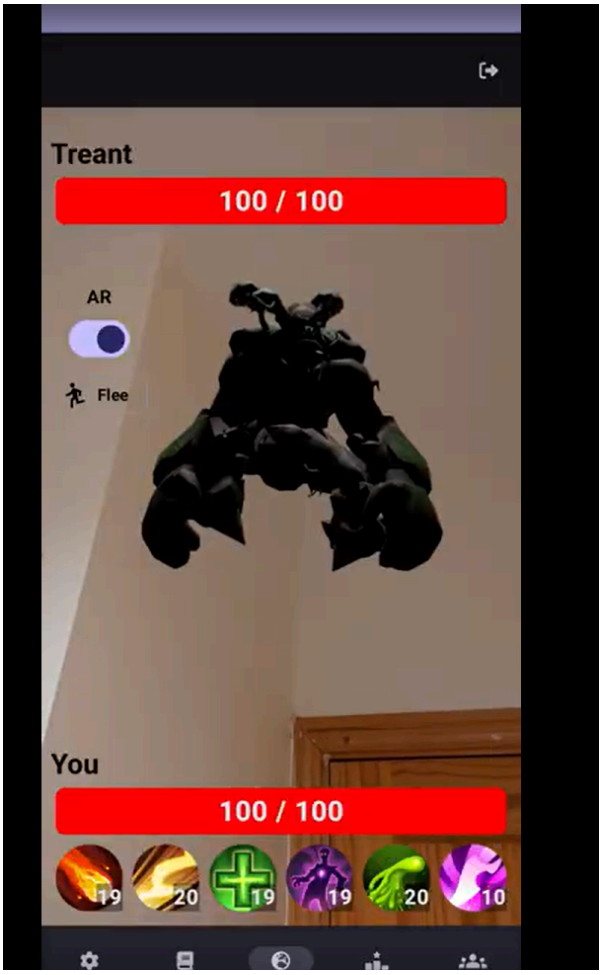
Anytime a spell is cast by the player in a battle, that spell is reduced by a single count in that player's Spellbook

Fig 7 - Battle Creature



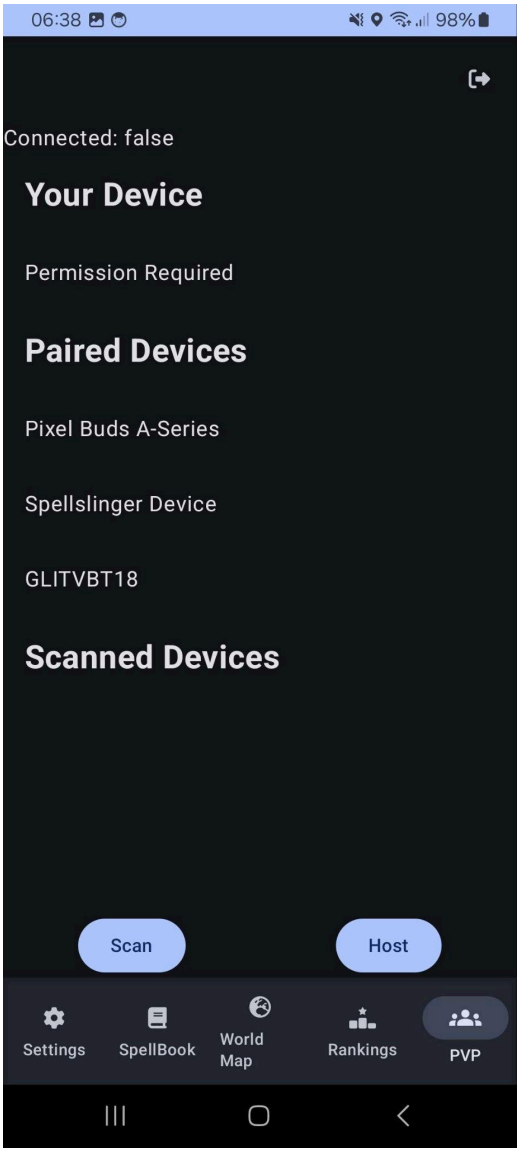
The dialog that pops up when the augmented reality enemy has appeared in the real world

Fig 8 - Enemy Proximity Warning



The enemy augmented reality model that has appeared in the player's room.

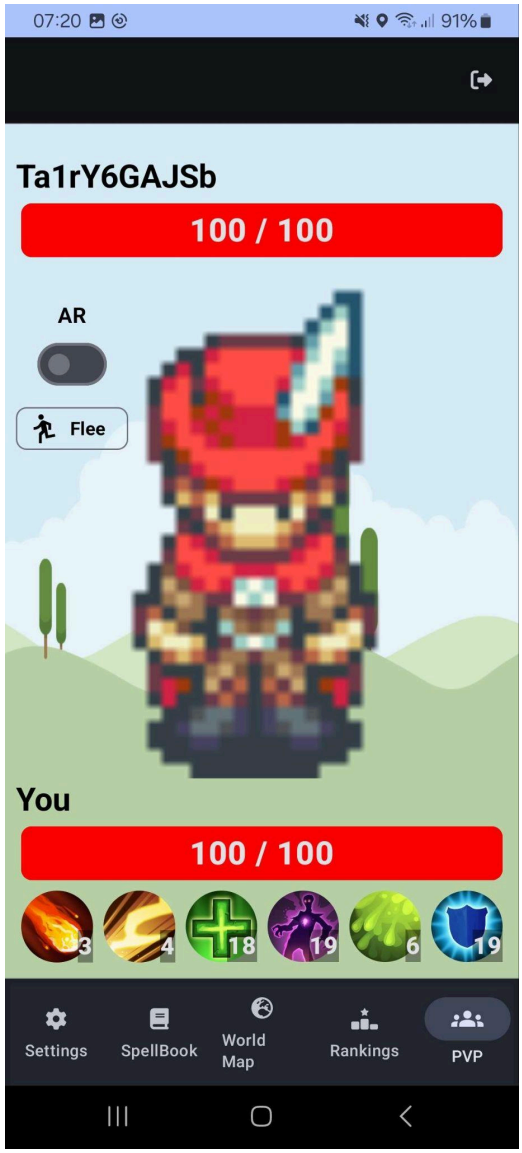
Fig 9 - AR Enemy



The PVP Lobby is a column that allows players to expose the app via bluetooth in order to facilitate location-based multiplayer. A player may scan for hosting apps as a client or vice-versa.

Once paired, the two devices communicate via a bluetooth socket connection to create a room id from the composite of their player ids. This room is then sent to the database to trigger a player-versus-player battle which the device will listen to from the battle screen which it has just navigated from.

Fig 10 - PVP Lobby



A Player versus Player battle largely plays the same as a Player versus creature battle. However the server waits for both spells to be cast before either execute, afterwards it deletes them both.The spellcast function against a player only triggers on write, and since the command still exists until that turn is executed, this prevents updates (attempted multiple spells being cast) from triggering the function again.

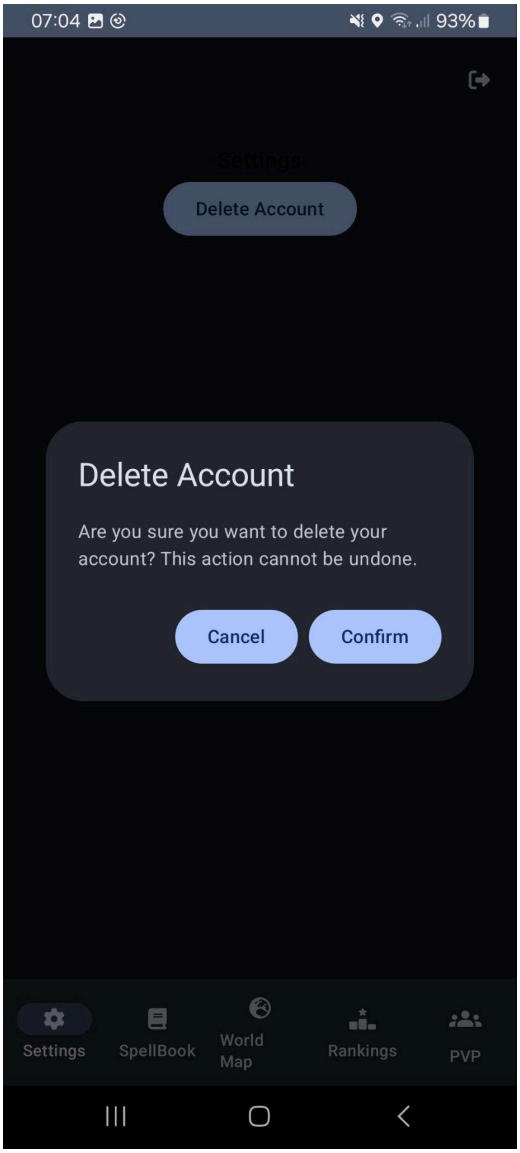
Fig 11 - Battle Player

The leaderboard is a simple screen that retrieves the current rankings from the database and orders them by win count in descending order.

Fig 12 - Leaderboard

The settings screen with the single available option to delete the player's account that invokes a confirmation dialog before the player confirms or decides against the deletion of their account.

Fig 13 - Settings Screen



The confirmation dialog for deleting a player's account. If confirm is pressed, then firebase deletes the player from their authentication servers and the application sends the player to the logout screen. Otherwise, if cancel is clicked or the dialog is dismissed, then nothing happens.

Fig 14 - Delete Account Confirmation Dialog

5. Database Structure

5.1 Creatures

```
{
  "creatures": {
    "bird": {
      "name": "bird",
      "spells": {
        "0": "razorwind",
        "1": "regenerate",
        "2": "toxic"
      }
    }
    // Other creatures
  }
```

Fig 15 - Creature Database Structure

5.2 Locations

```
{
  "locations": {
    // Geohash Key
    "gc765c2y": {
      "area": "wexford",
      "cardinal": "sw",
      "roaming": {
        <creatureId>: {
        "name": "wolf"
        "spawn": 1703131842.0428,
        "despawn": 1713515790.5952811,
        "latitude": 52.390837412514735,
        "longitude": -6.513812402352051
        }
        // Other creatures
      },
      "spellwells": {
        "0": {
          "id": "k9",
          "title": "Loch Gorman",
          "latitude": 52.39099345555365,
          "longitude": -6.513353943104681,
          "spells": {
            "cure": 0.2,
            "fireball": 0.5,
            "toxic": 0.3
          }
          // other spellwells
        }
      }
    }
  }
```

Fig 16 - Locations Database Structure

5.3 Distance

```
{
  "distance": {
    <playerId>: 10,
    // other players
  }
}
```

Fig 17 - Distance database structure

5.4 Leaderboard

```
{
  "leaderboard": {
    <playerId>: {
        "w": 0,
        "l": 0,
        "d": 0,
        "id": <playerId>
    }
    // other players
  }
}
```

Fig 18 - Leaderboard database structure

5.5 read_spellwells

```
{
  "read_spellwells": {
    "k0": {
      "spells": {
        "cure": 0.5,
        "toxic": 0.5
      }
    }
    // other wells
  }
}
```

Fig 19 - read_spellwells database structure

5.6 Skill_points

```
{
  "skill_points": {
    <playerId>: {
    "points": 160,
    "updated": 1713507587
    }
  } // Other players
}
```

Fig 20 - skill_points database structure

5.7 Spellbelts

```
{
  "spellbelts": {
    <playerId>: {
      "first": {
        "name": <spellname>,
        "count": 1
      },
  "second": {
        "name": <spellname>,
        "count": 2
      },
  "third": {
        "name": <spellname>,
        "count": 5
      },
  "fourth": {
        "name": <spellname>,
        "count": 11
      },
  "fifth": {
        "name": <spellname>,
        "count": 3
      },
  "sixth": {
        "name": <spellname>,
        "count": 2
      },
    }
  },
    // other spell belts
}
```

Fig 21 - Spellbelts database structure

5.8 Spellbooks

```
{
  "spellbooks": {
    <playerId>: {
        "absorb": 20,
        "fireball": 10,
        "toxic": 5
    }
    // other players spellbooks
  }
}
```

Fig 22 - Spellbooks database structure

5.9 Spells

```
{
  "spells": {
    "absorb": {
      "name": "absorb",
      "effect": "Gain health equal to the power of the countered spell",
      "type": "counter",
      "power": 0
    }
  }
}
```

Fig 23 - Spells database structure

6. Test Cases

6.1 On Auth/Delete Trigger Test Case

Name	**Result**
Spellbook is created on account creation	Passing
Spellbelt is created on account creation	Passing
Spell in Spellbelt contains is of equal count to SpellBook counterpart	Passing
Spellbook is deleted on account deletion	Passing
Spellbelt is deleted on account deletion	Passing

6.2 Draw Spells Test Cases

Name	**Result**
Player skill points are reduced on draw	Passing
Player spell count is increased on draw	Passing
Draw History is recorded on draw	Passing

6.3 Challenge Player Test Cases

Name	**Result**
A PVP battle can be looked up by either participating player’s id	Passing
A battle is created when a player is challenged	Passing
Both player’s exist in the battle object	Passing
A PVP battle starts with turn 0 created	Passing
Battle commands are deleted after the challenge is resolved	Passing

6.4 PVP Test Cases

Name	**Result**
Spell count is reduced when it is cast	Passing
An attack spell is applied against a status spell	Passing
A defence spell is applied against an attack spell	Passing
A status spell is applied against a defence spell	Passing
Defending player flinches when they use a status spell against an enemy’s attack	Passing
Attacker is countered when their enemy uses a counter against their attack spell	Passing
Player dodges when using a status spell against a defence spell	Passing
The battle is over when a player’s health reaches 0	Passing
An attack against an attack results in a tie	Passing
A defense spell against a defense spell results in a tie	Passing
A status spell against a status spell results in a tie	Passing
Win and Lose are recorded when the battle is over	Passing

6.5 Spell Tests Cases

Name	**Result**
Fireball deals 20 damage	Passing
Earthquake deals 20 damage	Passing
Razorwind deals 20 damage	Passing
Thunderbolt deals 20 damage	Passing
Cure restores 20 health	Passing
Cure cannot restore above max health (100)	Passing
Protect prevents damage	Passing
Counter deals twice as much damage as the attack spell it counters	Passing
Absorb restores health equal to damage that would have been taken	Passing
Absorb cannot restore more than max health (100)	Passing
Slime inflicts toxic status	Passing
Toxic inflicts toxic status	Passing
Regenerate applies regenerate status	Passing

6.6 Status Test Cases

Name	**Result**
Toxic deals 20 damage	Passing
Regenerate heals 20 health	Passing
Regenerate cannot heal past max health (100)	Passing
Status effects vanish when they have no remaining turns left	Passing

7. Challenges

7.1 GDPR

SpellSlinger uses, or intended to use, the player's location for a number of game mechanics such as granting the player skill points as a game currency for travelling distance in real life, making sure that the player doesn't draw from the same spell well more than once within a set duration, or making sure the the two player's battling are within the vicinity of each other. Each of these presented their own challenge.

Locations are collected client-side which means that players may be able to mock their location. Within that in mind, the server added a 'reasonability' check to make sure that players are travelling no more than five kilometres per hour, and if they are, to reduce their distance to that maximum. This may punish the more athletic players for exercising too intensely but for the sake of the average player it demonstrates its value.

SpellWells were initially intended to be time-based events where a player travels to a well in order to draw spells. The player who drew the spells would be prevented from drawing the spells for a set amount of time. However this was recorded by associating a player id with a timestamp against an entity that had a location. It is not a direct violation of GDPR, it may not be a violation at all, but it is associating a player's id with a location through a proxy entity, so to err on the side of caution this mechanic was discarded. SpellWells now use skill points as currency to draw spells.

The non-existent arena object was to be used to help identify the location of two players within the same area. This suffers from a similar issue of the Spell Well where I am associating a player id with a location through a proxy on the server. The solution to this was to introduce bluetooth as a matchmaking system because it pairs devices locally and is also significantly harder to fake than a location. The bluetooth devices pair to each other, once paired the host creates a bluetooth server socket for the client to join two, at this point they create a room id out of their two ids.

7.2 Augmented Reality

SpellSlinger uses Augmented Reality for player versus creature battles. Google has ARCore but it is a very bare API to the AR components of an application. There are additional libraries such as SceneForm from Google but it is no longer maintained, there is also SceneForm-Maintained which was the previously maintained version of Scene Form but it too is also no longer maintained. SceneView, a Jetpack Compose version of SceneForm, is what was eventually settled on.

SceneView presented some issues itself. SceneView handles the required camera permission automatically for applications that depend on it. This conflicted with something in my own personal “Samsung Galaxy A23” device which did not allow that because according to the Samsung Device the camera was required by a different application. This issue turned out to be an upstream problem with Samsung specifically, so I conceded and bought a Google Pixel 7a as a dedicated development device. In later versions of SceneView, this problem no longer exists however AR from SceneView still cannot run on Samsung Devices

7.3 Rules Enforcement

The initial plan for rules enforcement of the game mechanics was to use the “Firebase Database Rules” json document. This was a json document that mirrored the path of the Firebase Realtime Database, however it could compare information such as if the author of the database write had the same id as the player object it was writing too OR validation to check if the data that was sent is valid to write there.

Firebase Rules was not fit for this purpose past a certain point. Being a JSON document meant that comments were not allowed, which is unsuitable for what is essentially an expression language. The nested structure was also unpleasant to read. Where necessary these rules were replaced by rules inside the python function triggered by a database write to a command operation. In future, as well, the document would be structured in such a way so that it would be a list of different player owned domain models under a player id rather than a list of player ids under each domain model. This can help reduces the necessity of the “.read” and “.write” rules as it could be done on a player by player basis and delegating to the domain models rather than using it on a model by model basis and delegating it to the player

8. Learning Outcomes

1. Kotlin & Jetpack Compose Design Patterns

Kotlin Flows: Kotlin views data as a stream of changes rather than changes performed to one object. This is included in the design of Kotlin Flows which are relatively simple unbounded producer-consumer buffers that are typically thread safe within the context of Kotlin Code.

Kotlin Coroutines: Kotlin utilises Coroutines in order to ‘collect’ data from a Kotlin Flow on different threads (namely Main, IO & Undefined which may be any) and update other flows with that data. This is used in the project extensively in the World Map. The player location is one flow, when it changes it launches another flow that gets the players geohash, this in turn gets all eight neighbours of that geohash, then we use the combine function to combine all of these flows into a single observable flow that outputs a collection which is the entirety of spell wells and roaming creatures in the players current area.

Higher-Order Composables: Jetpack Composes creates UI by using a nested set of functions, for example a Row that represents a SpellBelt with Spell Composables as its children UI. To pass a callback event from the screen when a Spell is pressed, that event has to be first passed down to the spellbelt and then again into the Spell itself, which passes the event down to its child clickable, usually a button. If a dependency were to be added to the spell, it would need to travel the same several layers down. This leads to fragile code which can be hard to refactor. Jetpack Compose solves this problem through the use of composition, where the Spell composables are built at the same level of the SpellBelt, and those composables with the previously drilled state are now passed down instead of the state.

Stateful & Stateless Composables: With Composition in mind, Jetpack Compose emphasises the separation of stateful and stateless composables. The top-level composable should be a ‘Screen’ composable that is responsible for managing communication with the outside world through it’s related view model, and it passes that state down to a UI Composable which is the parent container of all the other UI Composables. In this case, the state should never pass more than two layers (one from the screen to the UI, and another from the UI to a child widget higher-order composable)

Sealed Events: A Sealed Event is a sealed interface that contains a list of classes or objects that implement its interface. It is reminiscent of enumerated classes but with more flexibility. Developers are encouraged to dispatch these events in a single onEvent callback that consumes a sealed event of that screen’s type and then applies an action based on it inside the UIs related View Model. This is in contrast to exposing each event as a callback directly to the composable which would explode the amount of parameters that exist in the argument list.

2. Firebase Write-Trigger-Observe Design Pattern

   I've been getting a better understanding of the firebase realtime database and I think the traditional methods of create-read-update-delete are not exactly what we want with the realtime database. Instead, everything that is interested in state listens through a 'callBackFlow' (a consumer-producer that consumers change from the network and produces events which are passed to a callback) and updates as it changes. Any create/delete operations required will be handled server-side. The app writes a command to the database, which triggers a function, if all firebase rule checks have passed, then that function updates a non-command value that the app is currently observing.

3. Google Maps

   The ability to interact and place various interactable items in real world locations on the google map.

4. SceneView

   The ability to place an AR Model relative to the player’s location in the world

5. Bluetooth

   To utilise BroadCastReceivers in order to scan for, pair with and connect to other bluetooth devices. The ability to run host and connect to a bluetooth socket server in which devices can communicate with each other via text strings.

9. Project Review

The aim of this project was to create an augmented reality geolocation mobile game. That aim was achieved. The majority of use cases that were set out were also successful, the use cases which did not meet development was due to a change in design philosophy or were altered in some way to fit the technical limitations of the application.

The use case to use skill points as currency to improve the attributes of the player was dropped. This mechanic complicates battle functions, leading to imbalance (what happens when a player who has invested several times more than another, battles that player?). It was dropped in favour of using skill points as currency at spell wells.

The original ‘draw’ use case was complicated by possible GDPR violations through the use of associating a player id with a location. This was discarded in favour of using skill points instead. This also leads to a tighter gameplay loop of travel -> draw -> battle -> travel.

The original multiplayer battle use case had to be modified to use bluetooth to determine player’s locality to each other, in order to avoid possible GDPR complications, and also strengthen.

The weight of how far firebase rules could sustain the complex logic of a battle was overzealous. The use of cloud functions allow for the rules to be extended into more complex expressions if required.

All in all, I would consider this project overall a success.

If I were to begin again, I would take the following changes:

- Start coding immediately
- Use an architecture pattern, such as “clean architecture” from the start, it may be unnecessary at first but it will repay
- Organise the database to have player related items to be listed under a player id, rather than have players listed under each player related item. This should also improve the rules structure by taking advantage of cascading rules (ie a player can read everything underneath their id)
- Implement Dependency Injection immediately.

10. Conclusion

In conclusion, the Spell Slinger game presents a fresh take on geolocation mobile gaming by directly incorporating the draw spell use case with the consumption of battle in spells. The addition of bluetooth in order to encourage matchmaking is an innovative technique in sidestepping concerns about GDPR while taking advantage of local devices.

11. Acknowledgments

Thanks to my supervisor Dr. Joseph Kehoe for allowing this project to exist and all the supervision that he has given me throughout the year.

Thanks to the youtuber, Phillip Lacker, who helped me through some of the more difficult parts of this project, especially the bluetooth connections.

12. Declaration of Plagiarism

Declaration of Plagiarism

Ollscoil Teicneolaíochta an Oirdheiscirt
South East Technological University

I declare, this document in this submission in its entirety is my own work except for where duty acknowledged. I have cited the sources of all the quotations, paragraphs, summaries of information, tables, diagrams, or other material. This includes software and other electronic media that is integral property rights may reside. I have provided the complete biography at the end of my document detailing all the works and resources used in the presentation of this submission. I am aware that failure to comply with the Institute's regulations governing plagiarism constitutes a serious offense.

Student	David Darigan - C00263218
Tutor	Joseph Kehoe
Institution	South East Technological University
Title	Spell Slinger
Submission Date	19/04/2024