# Creating a knowledge base
# for The Binding of Isaac

by

Tyler J. Bowcock

Supervisor: Dr. D D Freydenberger

Department of Computer Science

Loughborough University

May 2023

**Abstract**

# Contents

# List of Figures

# List of Acronyms

**ACID** Atomic, Consistent, Isolated, Durable

**AWS** Amazon Web Services

**HTML** Hyper-Text Markup Language

**HTTP** Hyper-Text Transfer Protocol

**JSX** JavaScript XML

**SQL** Structured Query Language

**WSGI** Web Server Gateway Interface

**XML** Extensible Markup Language

Tyler Bowcock

# Chapter 1

# Introduction

## 1.1 Problem Definition

Item interactions are an important mechanic of most modern roguelike/roguelite games, including The Binding of Isaac. However, with hundreds of items, each with a handful of good or bad interactions, it is nearly impossible to effectively remember them all. Graph databases are purpose-built to store and navigate relationships.[1] The output of this project will be a web application that leverages this feature of graph databases to allow users to query item interactions in The Binding of Isaac.

## 1.2 Aims and Objectives

The goal of this project is to make querying item interactions in The Binding of Isaac quicker and easier by using graph databases. Users will also be able to update the data in the database to ensure it matches any changes in the game.

The aims of the project are to:

1. Create a graph database containing relevant data about The

Binding of Isaac.

2. Develop a web application that utilises a graph database to helps users to find item interactions in the game.

3. Explore testing methodologies to aid in producing a stable application with high quality code.

4. Search for possible ways to extend the project with future updates.

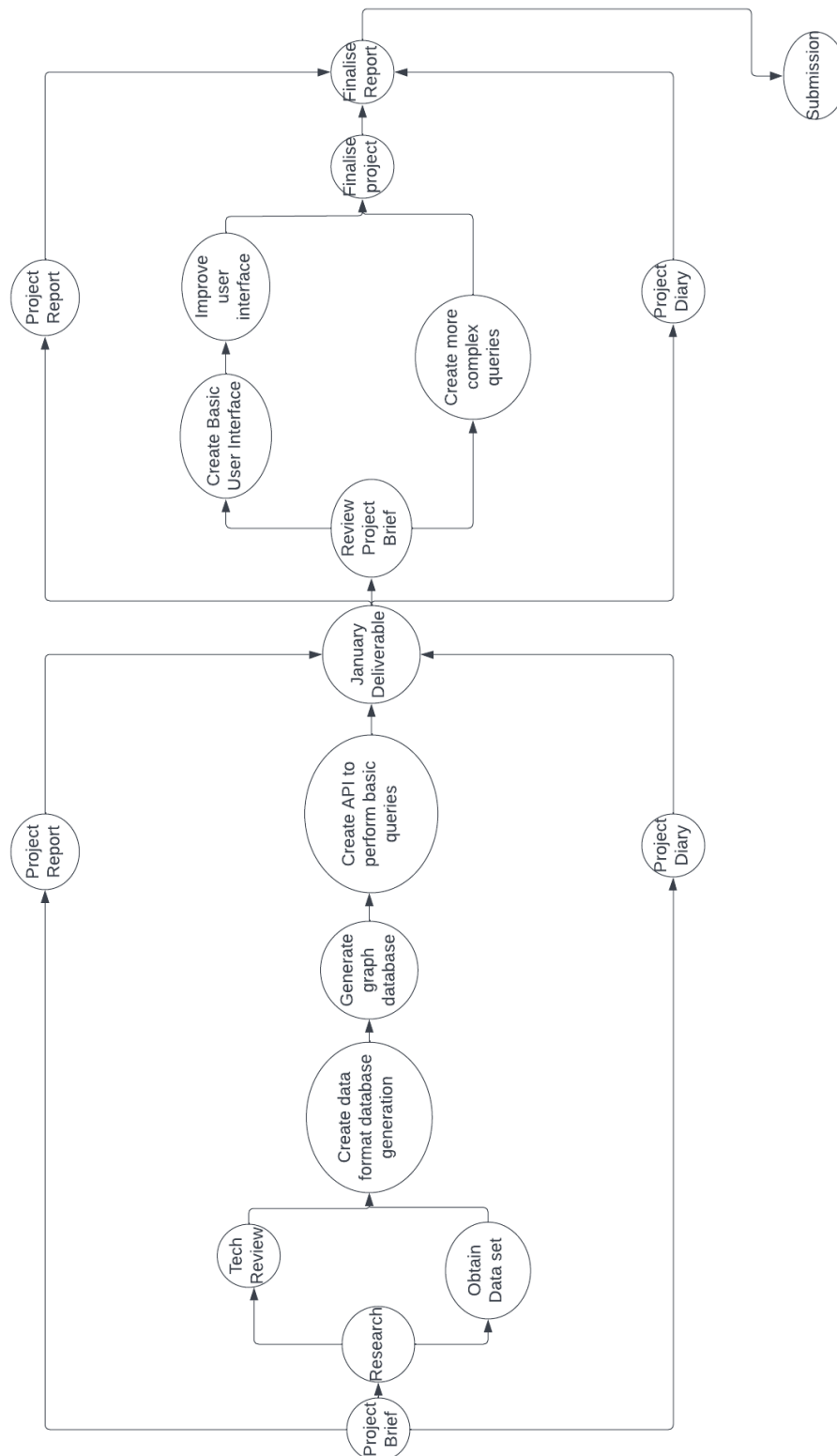## 1.3 Risks and Constraints

**Cost**

This project has no budget and so any services used in the development of the application will need to be free.

**Dataset Availability**

The data needed to create the database may become unavailable or unusable.

## 1.4 Project Plan

Figure 1.1: PERT chart showing project flow

# Chapter 2

# Background

## 2.1 Introduction

Through this chapter research will be conducted into varying areas related to the project. The conclusions made from this research will support any design or implementation decisions made in the course of the project.

## 2.2 Existing Solutions

While there is no existing solution that is a direct comparison to this project, there are applications that have a similar purpose. These will be analysed to determine what features should also be implemented in this project and what could be improved by this project.

### Fandom Wiki

The Binding of Isaac has two wiki sites hosted on the Fandom Wiki platform; one for the original flash game[2], and one for the modern version, commonly referred to as 'Rebirth'[3]. For the purposes of this project we will only be considering the modern

version as it is widely considered the 'goto' version within the game's community.

The website contains comprehensive information on all aspects of the game, and it is continually updated by the community. Users can navigate the site using either predefined categories or a powerful search tool.

Advantages

- Contains information on all aspects of the game

- Actively maintained by the community

- Useful search functionality

Disadvantages

- So much information can make it hard to find what is relevant

- Unable to search for interactions, have to go via each item

**Platinum God**

Platinum God is a self-described 'Isaac Cheat Sheet'[4], and it contains item and key mechanic information for all versions of the game. The site is maintained by one person, and it claims to be more accurate than the community wiki as its updates are 'tested thoroughly in the game using Cheat Engine'[5]. The information is split into pages based on the version of the game; users can navigate this using the item icons which are arrayed on the page, or by using the search functionality. The search tool has some supported keywords, but will still usually require entering an exact match to an entry in the data. For certain versions of the game there is also a synergy finder tool which lets the user

enter two items to see how they interact. However, this is limited to older versions of the game and only a small set of the items are actually included in the tool.

Advantages

- Information is more reliable than the community wiki

- Easier to reference quickly due to there being less information

Disadvantages

- Only one maintainer can mean long update times

- Only contains basic information about each item

- Limited or no synergy information for most items

- Harder to find items without knowing the name or what the item looks like

## 2.3 Technology Review

This section will research the potential technologies for the project and conclude with decisions about each.

### 2.3.1 Client Side Framework

**Angular**

'Angular is an application-design framework and development platform for creating efficient and sophisticated single-page apps.'[6] It was developed by Google to provide a complete framework for simple to complex single-page apps. Due to the size of this framework and the fact that it utilises Typescript makes for a steep learning curve, however this is counteracted by the large number of resources avaible from it being hugely popular and backed by

a large company.

**React**

'React is a declarative, efficient, and flexible JavaScript library
for building user interfaces. It lets you compose complex UIs
from small and isolated pieces of code called "components".'[7]
React allows the developer to use JSX to combine HTML and
JavaScript into one file, although this is not required to bene-
fit from React. Unlike Angular, this is a library and not a full
framework. This will reduce the amount of learning required to
use it, but it is likely extra libraries will be required to provide
all the functionality required. There is also a lot of resources
available as React is maintained by Meta and is the most popular
of the frameworks considered according the 2022 Stack Overflow
Developer Survey[8].

**Vue**

'An approachable, performant and versatile framework for build-
ing web user interfaces.'[9] While not backed by a large company,
Vue has grown in popularity and has many large commercial spon-
sors. This means there is likely to be less documentation avail-
able, but this library is relatively simple compared to the options
considered above.

**Conclusion**

After reviewing the three frameworks, the decision was made to
use Angular for the project. This is due to prior experience with
the framework, and with no background in JavaScript the increase
in learning curve from Typescript is negligible. However, any of

these choices would have been suitable for the project and the decision is mostly down to personal preference.

### 2.3.2 Server Side Framework

Due to the previously mentioned lack of JavaScript experience, the decision was made to only consider Python based frameworks to reduce the number of languages to learn.

**Django**

'Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design.'[10] As with Angular, this can be used as a full framework and so, while it will contain almost everything required, there will also be a lot of unnecessary features included. Django was designed to support rapid development, as shown by the slogan 'The web framework for perfectionists with deadlines.'[10]. It features database abstraction which allows developers to create 'Models' to represent the data stored in a database, handling communication with the database avoiding the need for coding in SQL.

**Flask**

'Flask is a lightweight WSGI web application framework.'[11] It is a micro-framework that started as a wrapper for Jinja and Werkzeug and has grown into a popular web framework. Flask does not provide much functionality that is already provided by another extension, this keeps Flask small but can introduce complex library requirements.

**Conclusion**

Django was chosen here, again due to prior experience, particularly in using Django and Angular together. While using two large frameworks may result in the project suffering from an amount of bloat, the prior experience means completing the project and having time to refine it is more likely.

### 2.3.3 Database

A graph database is a type of NoSQL database that uses nodes, edges, and properties to represent and store data. It is often described as storing data as it would be drawn on a whiteboard. This approach makes querying relationships in the data much faster as they are embedded in the data, rather than using JOIN operations or cross lookups often seen in SQL implementations. The underlying storage mechanism of a graph database can vary, some depending on an abstraction to store the data in a typical table based manner and some opting for a 'native' approach, maintaining the graph structure throughout the system.

**Neo4j**

'Neo4j is an open-source, NoSQL, native graph database that provides an ACID-compliant transactional backend for your applications'[12] Neo4j provide various tools for developing with graph databases, only the Aura platform will be considered here as it is the direct comparison for the other tools discussed, and a managed service is ideal for this project. A free database instance is provided in AuraDB for each account which allows up to 200000 nodes and 400000 relationships. Aura features a data import tool

that allows developers to create a model of the database structure which can then be populated with data from CSV files. It also has an explore tool, Bloom, this allows developers to view the database graphically and queries can be performed to show expected outputs. Neo4j provide a lot of useful documentation and tutorials for all the services they provide, this includes free e-books and course style content to guide users through the material.

**Amazon Neptune**

'Amazon Neptune is a purpose-built, high-performance graph database engine optimized for storing billions of relationships and querying the graph with milliseconds latency.'[1] Neptune is a managed graph database service provided by AWS. It offers a high throughput and low latency system that automatically scales with demand. As to be expected from any AWS product it is also highly secure and fault-tolerant. However, this comes at a cost; a free trial is offered for 30 days and after that a monthly fee is incurred with additional costs for data transfer, backups, and storage consumption.

**Conclusion**

The decision was made to use Neo4j. This is primarily because the Aura platform provides a permanent free database instance which has ample resources for this project. There also exists a Python libraries, neomodel and django-neomodel, for easily integrating database access using Django Models.

## 2.4 Conclusion

Throughout this chapter research has been conducted that will continue to provide benefit throughout the design and implementation stages. The review of existing solutions has highlighted the need for this application and has provided insight into how existing solutions have addressed the issue. The technology review has made it clear which technologies should be taken forward in the project as well as providing some background on why these technologies are needed and how they work.

# Chapter 3

# Requirements

# Chapter 4

# Design

## 4.1 Introduction

## 4.2 System Design

## 4.3 User Interface Design

## 4.4 Conclusion

Tyler Bowcock

# Chapter 5

# Implementation

## 5.1 Introduction

## 5.2 Tools

### 5.2.1 IDE

### 5.2.2 Version Control

### 5.2.3 Project Management

### 5.2.4 Database Visualisation

### 5.2.5 Hosting

### 5.2.6 CI/CD

### 5.2.7 Testing

Postman

### 5.2.8 Libraries

### 5.2.9 Client Side

### 5.2.10 Server Side

## 5.3 Data Processing

## 5.4 Database Interacion

## 5.5 Client Side

## 5.6 Conlcusion

# Chapter 6

# Testing

## 6.1 Introduction

### 6.1.1 Functionality Testing

### 6.1.2 Non-Functionality Testing

## 6.2 Conclusion

Tyler Bowcock

# Chapter 7

# Evaluation

## 7.1   Introduction

## 7.2   Project Evaluation

## 7.3   Future Work

## 7.4   Lessons Learned

## 7.5   Conclusion

# Appendix A

# Appendix

# References

[1]  *What Is a Graph Database?* Amazon Web Services, Inc. URL: `https://aws.amazon.com/nosql/graph/` (visited on 01/06/2023).

[2]  *The Binding of Isaac Wiki.* URL: `https://bindingofisaac.fandom.com/wiki/The_Binding_of_Isaac_Wiki` (visited on 01/09/2023).

[3]  *Binding of Isaac: Rebirth Wiki.* URL: `https://bindingofisaacrebirth.fandom.com/wiki/Binding_of_Isaac:_Rebirth_Wiki` (visited on 01/09/2023).

[4]  *Isaac Cheat Sheet - Platinum God.* URL: `https://platinumgod.co.uk/` (visited on 01/09/2023).

[5]  *Frequently Asked Questions - Isaac Cheat Sheet - Platinum God.* URL: `https://platinumgod.co.uk/faq` (visited on 01/09/2023).

[6]  *Angular - Introduction to the Angular Docs.* URL: `https://angular.io/docs` (visited on 01/09/2023).

[7]  *Tutorial: Intro to React – React.* URL: `https://reactjs.org/tutorial/tutorial.html` (visited on 01/09/2023).

[8]  *Stack Overflow Developer Survey 2022.* Stack Overflow. URL: `https://survey.stackoverflow.co/2022/?utm_source=social-share&utm_medium=social&utm_campaign=dev-survey-2022` (visited on 01/10/2023).

[9]  *Vue.Js - The Progressive JavaScript Framework — Vue.Js.* URL: `https://vuejs.org/` (visited on 01/09/2023).

[10]  *Django.* Django Project. URL: `https://www.djangoproject.com/` (visited on 01/09/2023).

[11]  Armin Ronacher. *Flask: A Simple Framework for Building Complex Web Applications.* Version 2.2.2. URL: `https://palletsprojects.com/p/flask` (visited on 01/09/2023).

[12]  *What Is a Graph Database? - Developer Guides.* Neo4j Graph Data Platform. URL: `https://neo4j.com/developer/graph-database/` (visited on 01/09/2023).