Water Pipe Puzzle Game

CS39620 Minor Project Report

Author: Arran Jones ([arj23@aber.ac.uk](mailto:arj23@aber.ac.uk))

Supervisor: Dr. Myra Wilson (mxw@aber.ac.uk)

25th April 2019

Version 1.4 (Draft)

This report is submitted as partial fulfilment of a BSc degree in  
Computer Science / Mathematics (GG14)

Department of Computer Science

Aberystwyth University

Aberystwyth

Ceredigion

SY23 3DB

Wales, UK

Declaration of originality

I confirm that:

* This submission is my own work, except where clearly indicated.
* I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
* I have read the regulations on Unacceptable Academic Practice from the University’s Academic Registry (AR) and the relevant sections of the current Student Handbook of the Department of Computer Science.
* In submitting this work, I understand and agree to abide by the University’s regulations governing these issues.

Name: Arran Jones

Date: 25/04/2019

Consent to share this work

By including my name below, I hereby agree to this project's report and technical work being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name: Arran Jones

Date: 25/04/2019

Acknowledgements

I am grateful to Myra Wilson for her support and guidance throughout my project and maintaining consistent meetings to ensure I'm progressing as planned.

I’d like to thank my user testers for their efforts and the advice they've provided for my project.

Abstract

As the popularity of puzzle games continues to grow day by day, there is a distinct lack of levels available to the consumer. There may be several reasons behind this, but the major two reasons are:

* Pay walling in order to get more money from the players and in exchange give them a new set of levels
* A lack of future level additions to the game, due to a lack of support or a struggle to design unique puzzles.

The premise behind this project is to create a simple puzzle game that goes against this trend, while also maintaining the replayability of classic puzzle games.

This project will be a simple water pipe puzzle game that will generate its own random (but solvable) levels, increasing the replay value of game. The game will also save some basic statistics for the game in order to give the user a slightly more competitive feeling when playing the game at different difficulties, this should further help maintain a more replayable experience.

Contents

1. Background, Analysis & Process 2

1.1. Background 2

1.1.1. Background Preparation 2

1.1.2. Similar Applications 2

1.1.3. Functionality Research 2

1.1.4. Project Goals 2

1.2. Analysis 2

1.1.3. Functional Requirements 2

1.3. Process 2

2. Design & implementation 2

2.1. Overall Architecture 2

2.2. Sprint 1 - User Interface 2

2.3. Sprint 2 - Grid Generation 2

2.4. Sprint 3 - Implementing Search Algorithm 2

2.5. Sprint 4 - Determining Difficulty 2

2.6. Sprint 5 - Saving Statistics 2

2.7. Sprint 6 - Reading Statistics from Database 2

3. Testing 2

3.1. Overall Approach to Testing 2

3.2. Automated Testing 2

4.2.1. Unit Tests 2

4.2.2. User Interface Testing 2

4.2.3. Stress Testing 2

4.2.4. Other Types of Testing 2

3.3. Integration Testing 2

4.4. User Testing 2

4. Critical Evaluation 2

5. Annotated Bibliography 2

6. Appendices 2

A. Third-Party Code and Libraries 2

B. Ethics Submission 2

C. Code Samples 2

# Background, Analysis & Process

## Background

### Background Preparation

Before beginning the project, some major decisions had to be made, such as the platform the game would be made for, the language it would be written in and the search algorithm required to check for solutions.

**Platform and Language**

The first major decision that had to be made was whether the project would be a mobile or a desktop application.

Desktop has the advantages of more powerful hardware and larger screen sizes, allowing for more flexibility in grid sizes and processing power for the search algorithm used. This could potentially make the game run smoother and allow for some customisability, such as allowing the player to choose the size of the grid.

Mobile has the advantage of a more widely adopted touch screen interface. This would allow for a more intuitive interface and a cleaner feel for rotating the pipes. The game is also very simple and feels more suited to a mobile experience, as it’s the kind of game one would play when they have a spare five minutes with nothing else to do.

If mobile is to be used, there is still the decision of either Android or iOS. There are similarities between the two, such as the touchscreen for input, but there are very fundamental differences.

iOS is commonly seen as the easier mobile platform to develop for as it uses swift, a language based on Objective-C which was created by apple, specifically for iOS development. This would allow faster and more efficient development of the project, while maintaining a similar level of performance. A major disadvantage to iOS development is the requirement of XCode which is a mac exclusive IDE, preventing development on a Windows or Linux machine. There is also no prior knowledge of Objective-C, so there would be a large amount of research needed in order to complete this project.

Android development uses Android Studio, which is completely cross-platform. This is a major advantage as it will allow for production on any desktop or laptop, including mac if necessary. Android also has a predicted mobile operating system market share of around 72.77%, in comparison to apple’s 26.21% as of March 2019 (according to statcounter.com). This provides a much larger pool of users who may wish to download the app, increasing popularity, and in a real-world scenario providing greater monetary returns.

If Android is chosen as the mobile operating system of choice, there would be one final decision to make, which would be the language used to develop the application. Android studio officially supports two languages; Java and Kotlin.

Java is a very reputable object-oriented language which has been the standard language for developing Android applications for many years. It has the advantage of a large amount of documentation and a much larger community of programmers using the language to write support material.

Kotlin is both an object-oriented and functional programming language, this has the advantage of allowing users to use elements from both forms. The number of Android developers adopting Kotlin is increasing day by day, increasing the amount of support material available, but there was no prior knowledge of this language, therefore a relatively large amount of research would be required to learn the language.

**Search Algorithms**

When first researching the search algorithm that would be required for this project, the search algorithm the internet repeatedly suggested was an A\* search. This is a heuristic search algorithm that always finds the shortest path between two points by always choosing a path which decreases the Euclidean distance as much as possible at each vertex. Upon further thought, this would not make sense for this project as it is required to find every solution, not just the shortest.

After further research, an iterative deepening depth-first search was looking like a very strong option. This is a depth-limited version of a depth first search which is run repeatedly at incrementally larger depths in order to find the shortest path. This can be adapted to allow for the search to continue running until the whole graph has been searched but increases the processing power required.

In order to keep processing power to a minimum it was decided either a simple depth-first or breadth-first search would also be suitable. A depth-first search is a non-heuristic search in which a path is traversed for as long as possible, checking at each stage to see if it has reached its goal. If the target vertex is not found by the end of the path it will take a step back and check for another path to follow, it will repeat this until all paths have been traversed. A breadth-first search is another non-heuristic search in which all neighbours of the root node are checked first, if the target vertex is not found, then the search will continue to the next level of vertices (neighbours of the previously checked vertices.

### Similar Applications

When searching for similar applications on the google play store, only a few were found. Each with the same basic structure of a level progression system. This meant that there weren’t any that generated their own levels and left a gap in the market.

The games also had very similar sized grids, ranging from 5x5 to 9x11. This should provide a decent guideline for the size of the grid to be used in this project.

ADD TO THIS

### Functionality Research

The main functionality research that will be required for this project will be Android development, as there was no prior knowledge of this topic. Android development with java is fundamentally similar but uses slight differences, such as the use of activities for the UI and a close integration with the Android SDK.

In order to research this, the “Codelabs for Android Developer Fundamentals (V2)” training course on the Android Developers website will be used. This is an introduction course to help programmers with prior Java knowledge learn Android development. For some of the other parts necessary for this project some of the other documentation provided on the Android Developers website would be used.

The other functionality research required will be for the search algorithm itself as there was no prior knowledge of these either. This would entail the research already performed in the deciding of the search algorithm, the different ways of implementing a depth-first search and which way would be most suitable for this project and the steps that need to be added in order to deal with the changing connectivity of the graph.

### Project Goals

* **Create a fully functional self-generating game**

For this game to be considered functional, it should be able create a random solvable grid with a number the solutions determined by the difficulty the user selects. This should be done using a search algorithm and should not be pre-defined by the developer. The application should also be able to determine when a solution has been found by the user and notify them of the time taken to complete the puzzle and the number of pipe rotations it took the user to solve the puzzle. These solutions should then be saved into a database, where they can be used to determine some basic statistics.

* **Gain knowledge of Android development**

An aim of this project was to broaden knowledge for development on different platforms. This is essential for professional development and will be very useful for future projects that may be undertaken. A relatively good knowledge of Java was already there prior to this project, so Android development was a good place to start.

* **Gain knowledge of search algorithms**

Prior to this project there was little to no knowledge of search algorithms, which are an essential part of modern software development, therefore for further professional development, this was seen as a good project to undertake.

## Analysis

After the background research, some key decisions needed to be made.

Firstly, it was decided that a mobile operating system would be more appropriate for this project as it is a simple puzzle game. The touch screen interface of a mobile device makes the UI required for this game far more intuitive and the kind of game being created is much more suitable for a mobile game than a desktop game, due to the minimum amount of inputs necessary.

Once the platform was decided, the choice of operating system and language was a relatively simple one as there was still a lot of research that needed to be done before commencing the project. Therefore, Android would be chosen as the operating system, as the prior knowledge in Java and restricted access to a mac made this an easy choice.

After deciding that Java would be used, the next decision to make was what search algorithm to use. Initially, iterative deepening was looking like the best decision, but on further inspection this took far more processing power than a simple depth-first or breadth-first search. As this game works by connecting pipes, creating a single branch tree, in order to get from one point to another; the most logical search algorithm to use would be a variation on a depth first search. A basic depth first search would not be possible as some paths would not be traversable until a pipe had been rotated. Therefore, an extra stage would have to be implemented in order to check all possibilities for each rotation of each vertex.

### Functional Requirements

After analysing the problem, it was decided that there should be 5 main functional requirements:

* Generate a random solvable grid and determine when the grid has been solved
* Ensure the grid is solvable using a search algorithm
* Determine the difficulty selected and choose a suitable grid
* Save statistics for each difficulty into a database
* Read the statistics from the database and show them on a dedicated page

These seemed like the most sensible requirements for the project, as it provides a decent amount of content while maintaining a realistic amount of work for the time allowed.

## Process

It was decided that a Scrum methodology would be suitable for this project, as progressing week by week would help ensure the project would be complete on time while also allowing the planning for each sprint to be done at the start of the sprint. A logical process for each sprint would be to implement one of each of the functional requirements, with one additional sprint at the beginning for creating the user interface.

However, a variation on scrum is required as it is a solo project, therefore there will not be multiple people to fill the roles necessary for a scrum methodology (e.g. scrum master). This was chosen as the methodology as it is a good way of maintaining a solid management of time, with each sprint considering the amount of time left until the end date of the project. The sprints will each be a week long, with the first couple days dedicated to designing the work for that sprint.

Using the emulators provided by Android studio and the created APK files with physical handheld devices will ensure correct functionality across multiple device types and screen sizes.

# Design & Implementation

You should concentrate on the more important aspects of the design. It is essential that an overview is presented before going into detail. As well as describing the design adopted it must also explain what other designs were considered and why they were rejected.

The design should describe what you expected to do and might also explain areas that you had to revise after some investigation.

Typically, for an object-oriented design, the discussion will focus on the choice of objects and classes and the allocation of methods to classes. The use made of reusable components should be described and their source referenced. Particularly important decisions concerning data structures usually affect the architecture of a system and so should be described here.

How much material you include on detailed design and implementation will depend very much on the nature of the project. It should not be padded out. Think about the significant aspects of your system. For example, describe the design of the user interface if it is a critical aspect of your system, or provide detail about methods and data structures that are not trivial. Do not spend time on long lists of trivial items and repetitive descriptions. If in doubt about what is appropriate, speak to your supervisor.

You should also identify any support tools that you used. You should discuss your choice of implementation tools - programming language, compilers, database management system, program development environment, etc.

Some example sub-sections may be as follows, but the specific sections are for you to define.

The implementation should discuss any issues you encountered as you tried to implement your design. During the work, you might have found that elements of your design were unnecessary or overly complex; perhaps third-party libraries were available that simplified some of the functions that you intended to implement. If things were easier in some areas, then how did you adapt your project to take account of your findings?

It is more likely that things were more complex than you first thought. In particular, were there any problems or difficulties that you found during implementation that you had to address? Did such problems simply delay you or were they more significant?

You can conclude this section by reviewing the end of the implementation stage against the planned requirements.

## Overall Architecture

As the project is being made in Android Studio using Java, a good place to start was to create a UML Class Diagram for the whole system. This Introduced a lot of early issues, but after restructuring it a couple of times it began to look much more structured:

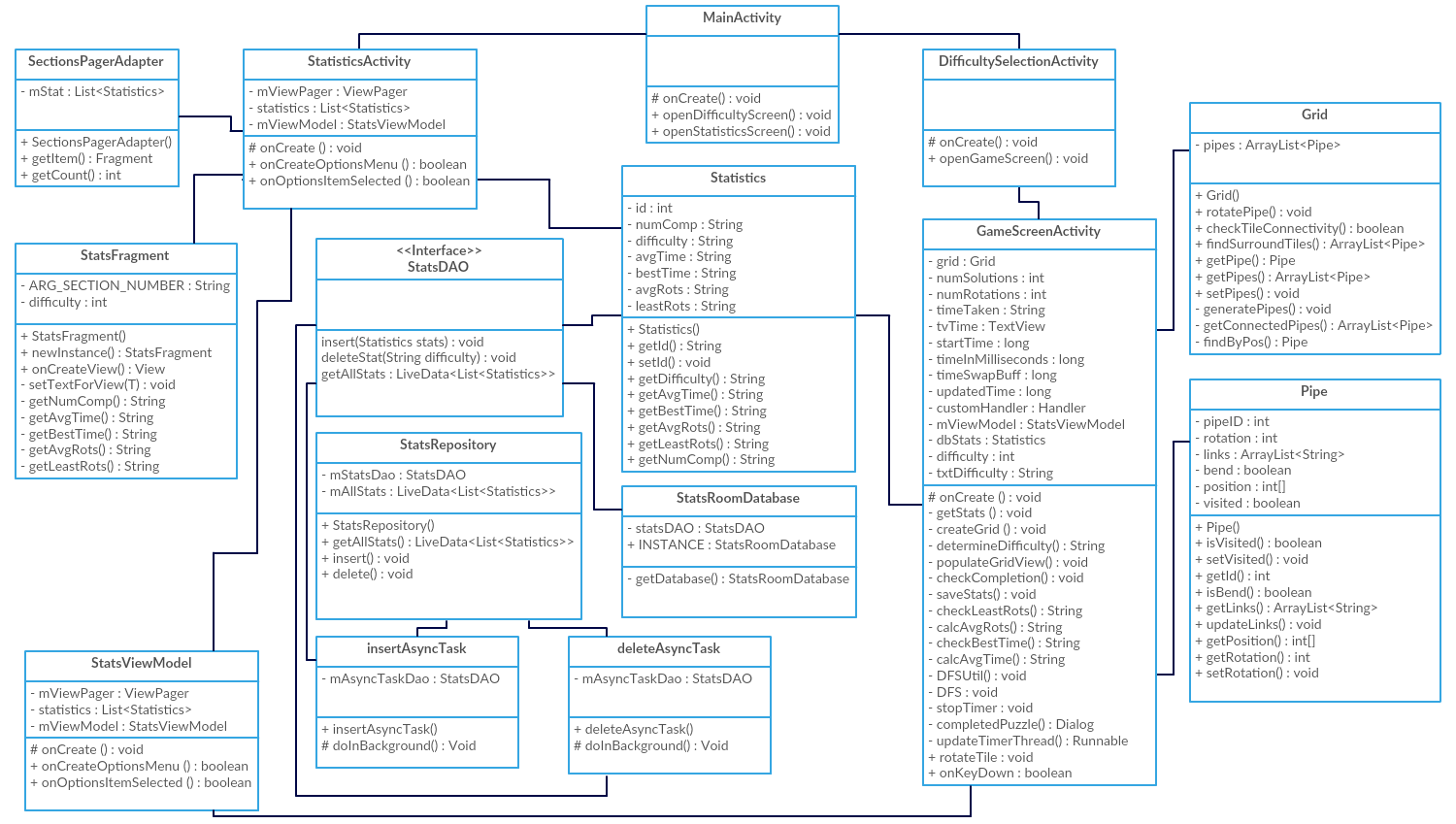


Figure 1. UML Class Diagram of entire project.

Creating this before beginning the project should allow for more efficient development once the implementation is started. In this diagram it can be seen that there are four activities (Androids word for screen), two objects for the game itself and one interface & four classes for the database.

## Sprint 1 – User Interface

**Design**

An aesthetically pleasing user interface is not the most important thing about this project as it is a minor project and getting a functional game is much more important in order to meet the goals of the project. In order to design the user interface FluidUI.com will be used. FluidUI is a website which is used to create prototype web and mobile prototypes with a drag and drop style interface. This will allow for a swift design of the user interface, allowing for more time to learn android development and more time to recreate the design.

The UI designed is very simplistic, but keeps a very easy to understand layout:

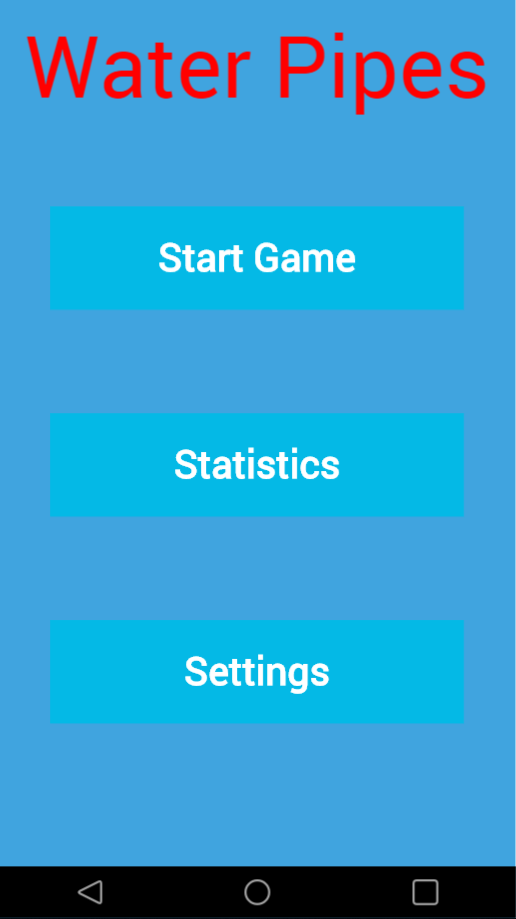
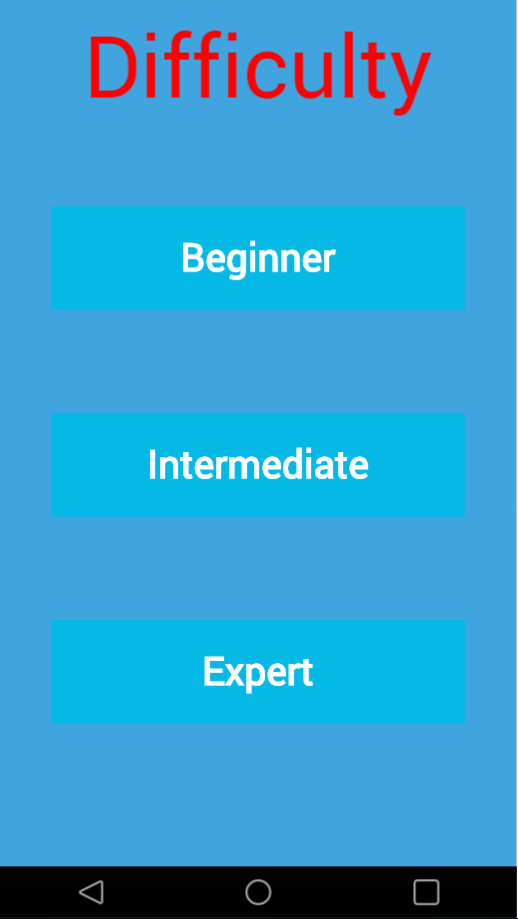
 

Figure 2. Prototype Main Activity Figure 3. Prototype Difficulty Selection Activity

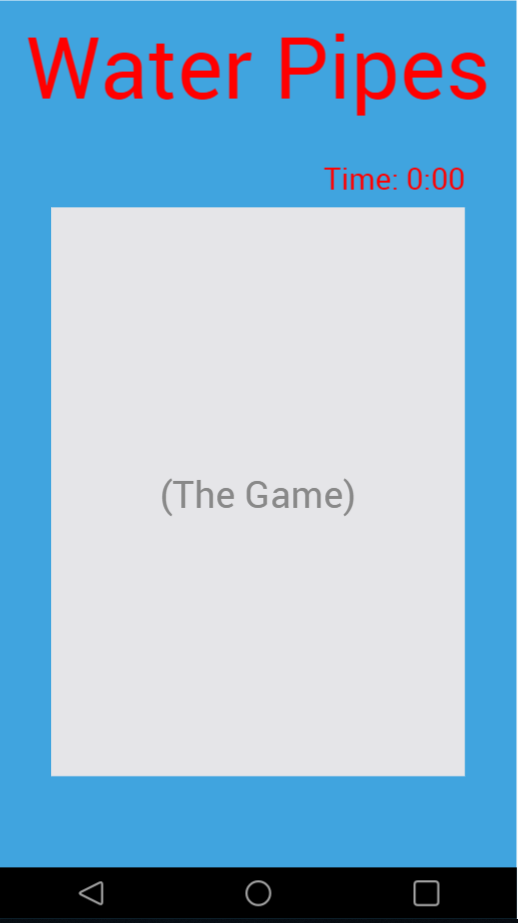
 

Figure 4. Prototype Game Activity Figure 5. Prototype Statistics Activity

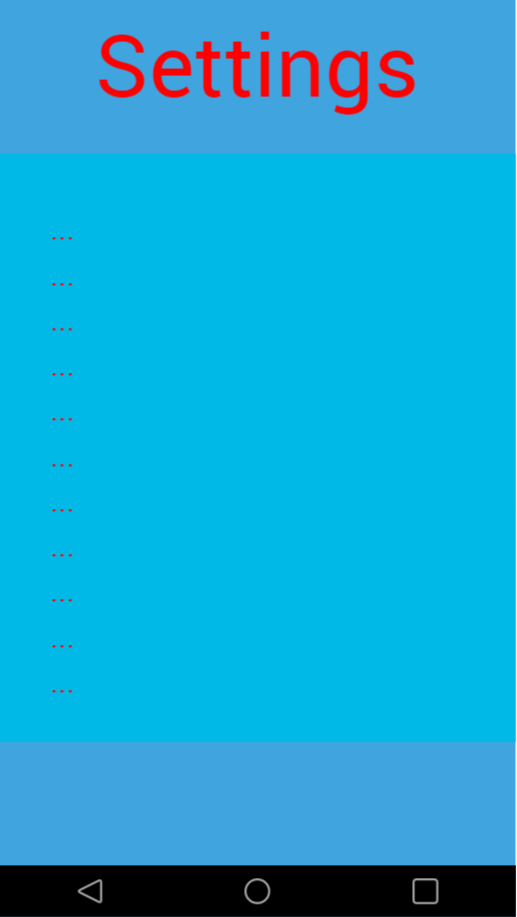


Figure 6. Prototype Settings Activity

The Prototype settings page lacks content as it was undecided as to what settings would be available to the user at the time, and the Statistics page only had a few items of text as the more time that was available at the end would determine how many statistics would be on the final product.

**Implementation**

The implementation of the user interface was relatively seamless, as Android Studio also has a drag and drop style UI builder, but there were a couple of changes made compared to the original prototypes, such as the removal of the Settings button and activity. The Settings button was removed as it is seen as bad mobile development practice to have a button to navigate the user to a settings page. The settings activity was then removed entirely as it was decided that adding settings may add more functionality to the application than is required, making it a larger install than necessary. The time was also moved on the game screen for aesthetic reasons, as there was a lot of blank space nearer the bottom of the screen.

The user interface ended up looking like:

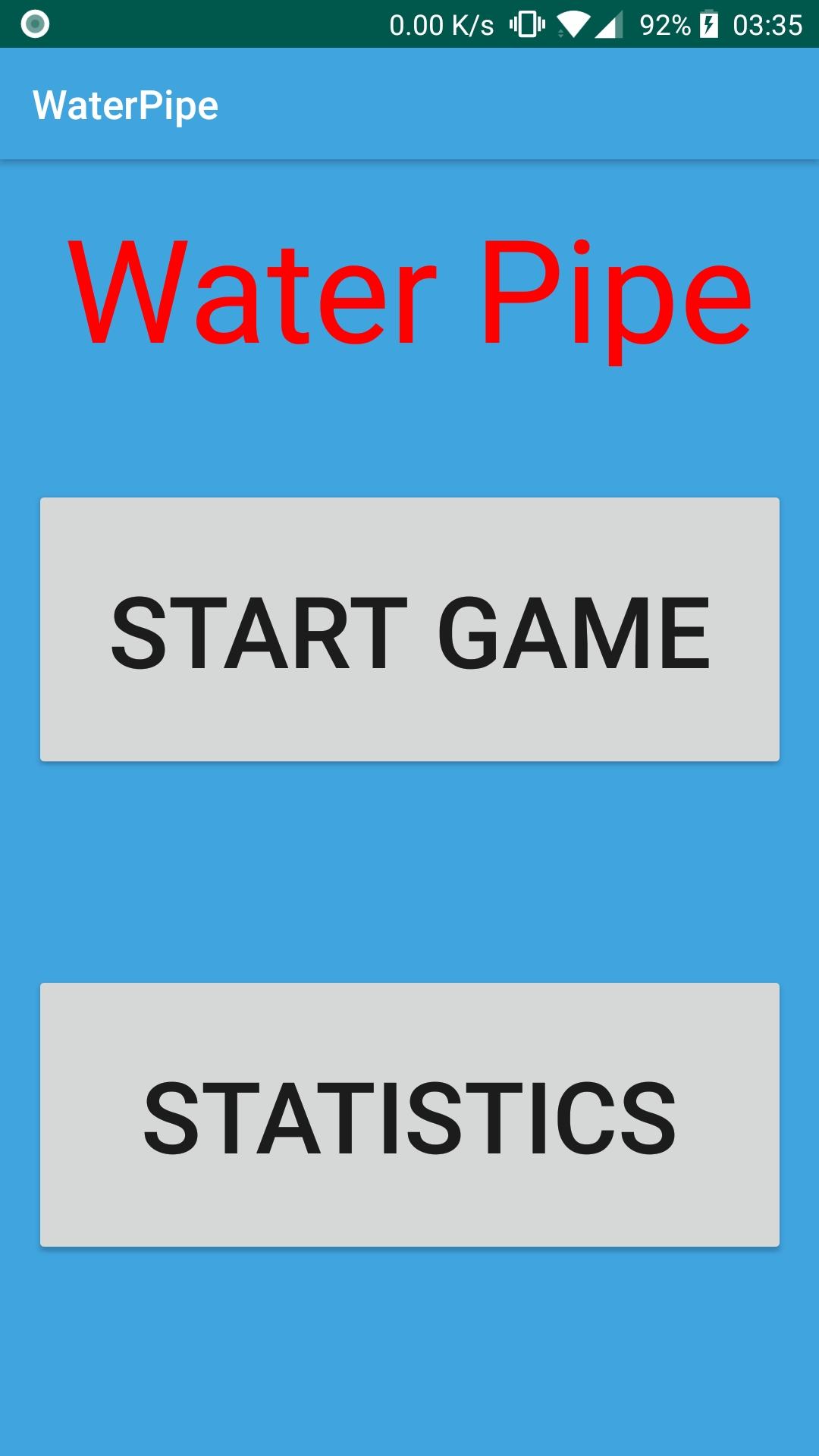
 

Figure 7. Main Activity Figure 8. Difficulty Selection Activity

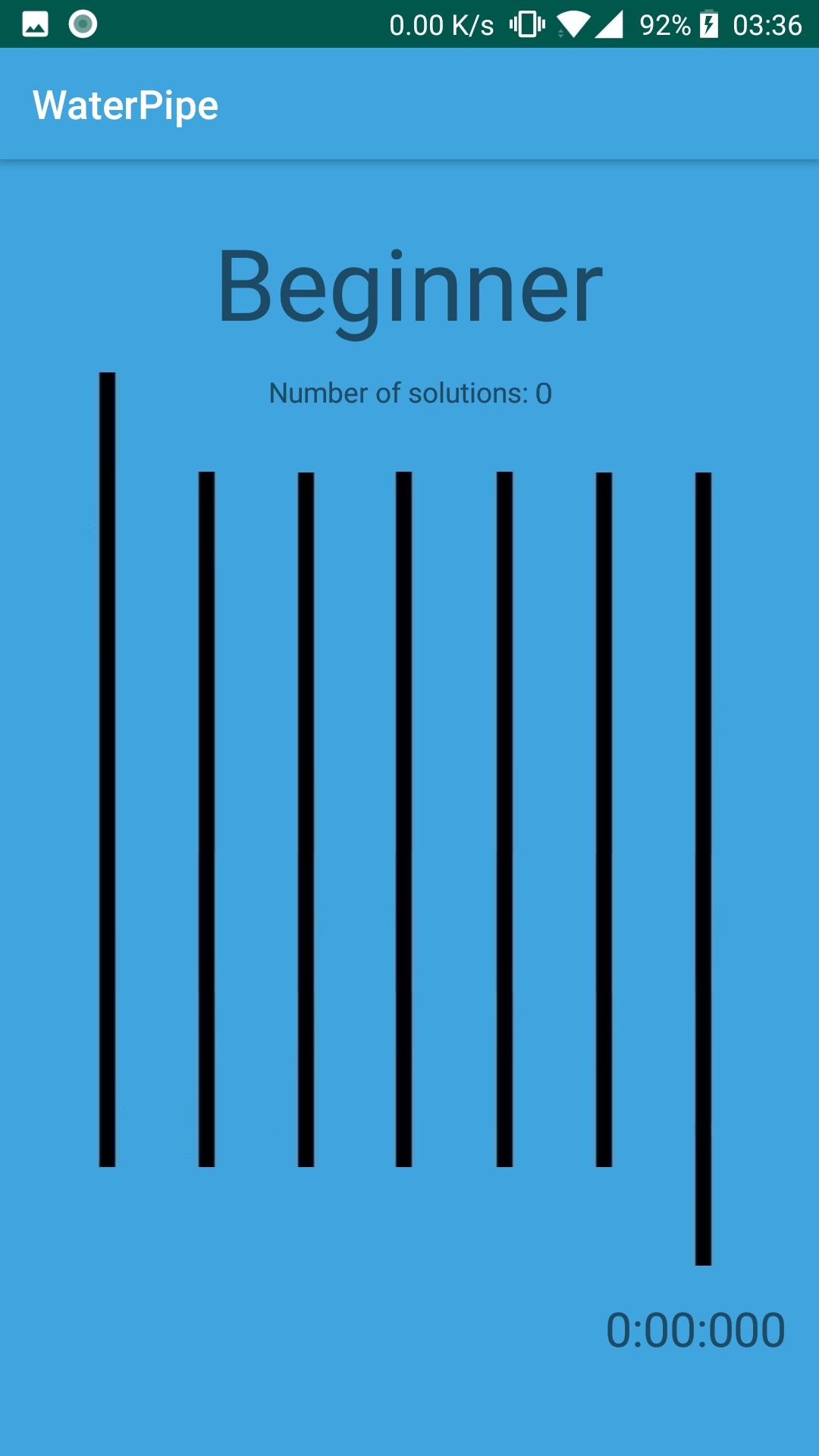
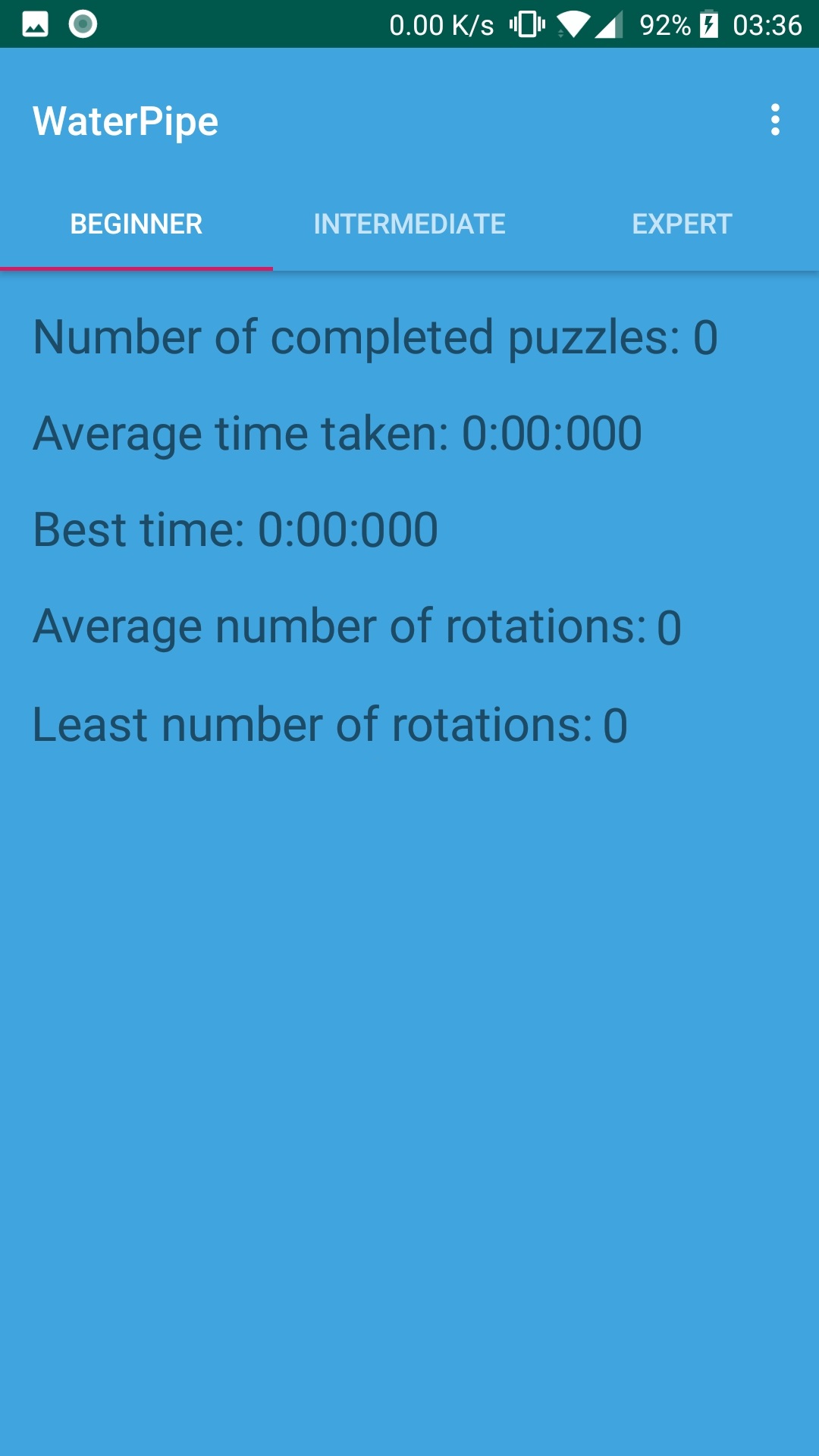
 

Figure 9. Game Activity Figure 10. Statistics Activity

Overall, the user interface stayed relatively close to the original designs, but upon uploading the game to a mobile, it was decided that the buttons should not be blue, as indirect sunlight it was tough to distinguish between the background and the buttons. It was also decided to add a text view to the game screen that shows the number of possible solutions for the puzzle.

## Sprint 2 – Grid Generation

## Sprint 3 – Implementing Search Algorithm

## Sprint 4 – Determining Difficulty

## Sprint 5 – Saving Statistics

## Sprint 6 – Reading Statistics from Database

# 

# Testing

Detailed descriptions of every test case are definitely not what is required in this section; the place for detailed lists of tests cases is in an appendix. In this section, it is more important to show that you adopted a sensible strategy that was, in principle, capable of testing the system adequately even if you did not have the time to test the system fully.

Provide information in the body of your report and the appendix to explain the testing that has been performed. How does this testing address the requirements and design for the project?

How comprehensive is the testing within the constraints of the project? Are you testing the normal working behaviour? Are you testing the exceptional behaviour, e.g. error conditions? Are you testing security issues if they are relevant for your project?

Have you tested your system on “real users”? For example, if your system is supposed to solve a problem for a business, then it would be appropriate to present your approach to involve the users in the testing process and to record the results that you obtained. Depending on the level of detail, it is likely that you would put any detailed results in an appendix.

Whilst testing with “real users” can be useful, don't see it as a way to shortcut detailed testing of your own. Think about issues discussed in the lectures about until testing, integration testing, etc. User testing without sensible testing of your own is not a useful activity.

The following sections indicate some areas you might include. Other sections may be more appropriate to your project.

## Overall Approach to Testing

## Automated Testing

### Unit Tests

### User Interface Testing

### Stress Testing

### Other Types of Testing

## Integration Testing

## User Testing

# Critical Evaluation

Examiners expect to find a section addressing questions such as:

* Were the requirements correctly identified?
* Were the design decisions correct?
* Could a more suitable set of tools have been chosen?
* How well did the software meet the needs of those who were expecting to use it?
* How well were any other project aims achieved?
* If you were starting again, what would you do differently?

Other questions can be addressed as appropriate for a project.

The questions are an indication of issues you should consider. They are not intended as a specification of a list of sections.

The evaluation is regarded as an important part of the project report; it should demonstrate that you are capable not only of carrying out a piece of work but also of thinking critically about how you did it and how you might have done it better. This is seen as an important part of an honours degree.

There will be good things in the work and aspects of the work that could be improved. As you write this section, identify and discuss the parts of the work that went well and also consider ways in which the work could be improved.

In the latter stages of the module, we will discuss the evaluation. That will probably be around week 9, although that differs each year.

# Annotated Bibliography

This final section should list all relevant resources that you have consulted in researching your project. Each reference should also include a brief annotation.

1. Sylvia Duckworth. A picture of a kitten at Hellifield Peel. <http://www.geograph.org.uk/photo/640959>, 2007. Copyright Sylvia Duckworth and licensed for reuse under a Creative Commons Attribution-Share Alike 2.0 Generic Licence. Accessed August 2011.

This is my annotation. I should add in a description here.

1. Mark Neal, Jan Feyereisl, Rosario Rascunà, and Xiaolei Wang. Don’t touch me, I’m fine: Robot autonomy using an artificial innate immune system. In *Proceedings of the 5th International Conference on Artificial Immune Systems*, pages 349–361. Springer, 2006.

This paper…

1. W.H. Press et al. *Numerical recipes in C*. Cambridge University Press Cambridge, 1992.

This is my annotation. I can add in comments that are in **bold** and *italics*and then further content.

1. Various. Fail blog. <http://www.failblog.org/>, August 2011. Accessed August 2011.  
     
   This is my annotation. I should add in a description here.
2. Apache Software Foundation (2014) “*Apache POI - the Java API for Microsoft Documents*” (Online) Available at: <http://poi.apache.org> Accessed: 14th March 2014.

This is my annotation. I should add in a description here.

1. Apache Software Foundation (2004) “Apache License, Version 2.0” (Online) Available at: <http://www.apache.org/licenses/LICENSE-2.0> Accessed: 14th March 2014.

This is my annotation. I should add in a description here.

1. Neil Taylor, “MMP\_S08 Project Report and Technical Work”, 2019 (Online) Available at: <http://blackboard.aber.ac.uk/> Accessed 19th February 2019.

A document that outlines information about the marking guide for the Project Report and Technical Work. This is published in the Resources folder on Blackboard.

# Appendices

The appendices are for additional content that is useful to support the discussion in the report. It is material that is not necessarily needed in the body of the report, but its inclusion in the appendices makes it easy to access.

For example, if you have developed a Design Specification document as part of a plan-driven approach for the project, then it would be appropriate to include that document as an appendix. In the body of your report you would highlight the most interesting aspects of the design, referring your reader to the full specification for further detail.

If you have taken an agile approach to developing the project, then you may be less likely to have developed a full requirements specification. Perhaps you use stories to keep track of the functionality and the ’future conversations’. It might not be relevant to include all of those in the body of your report. Instead, you might include those in an appendix.

There is a balance to be struck between what is relevant to include in the body of your report and whether additional supporting evidence is appropriate in the appendices. Speak to your supervisor or the module coordinator if you have questions about this.

* 1. Third-Party Code and Libraries

If you have made use of any third-party code or software libraries, i.e. any code that you have not designed and written yourself, then you must include this appendix.

As has been said in lectures, it is acceptable and likely that you will make use of third-party code and software libraries. If third-party code or libraries are used, your work will build on that to produce notable new work. The key requirement is that we understand what your original work is and what work is based on that of other people.

Therefore, you need to clearly state what you have used and where the original material can be found. Also, if you have made any changes to the original versions, you must explain what you have changed.

The following is an example of what you might say.

**Apache POI library** – The project has been used to read and write Microsoft Excel files (XLS) as part of the interaction with the client’s existing system for processing data. Version 3.10-FINAL was used. The library is open source and it is available from the Apache Software Foundation [5]. The library is released using the Apache License [6]. This library was used without modification.

Include as many declarations as appropriate for your work. The specific wording is less important than the fact that you are declaring the relevant work.

* 1. Ethics Submission

This appendix includes a copy of the ethics submission for the project. After you have completed your Ethics submission, you will receive a PDF with a summary of the comments. That document should be embedded in this report, either as images, an embedded PDF or as copied text. The content should also include the Ethics Application Number that you receive.

* 1. Code Samples

This is an example appendix. Include as many appendices as you need. The appendices do not count towards the overall word count for the report.

For some projects, it might be relevant to include some code extracts in an appendix. You are not expected to put all of your code here - the correct place for all of your code is in the technical submission that is made in addition to the Project Report. However, if there are some notable aspects of the code that you discuss, including that in an appendix might be useful to make it easier for your readers to access.

As a general guide, if you are discussing short extracts of code then you are advised to include such code in the body of the report. If there is a longer extract that is relevant, then you might include it as shown in the following section.

Only include code in the appendix if that code is discussed and referred to in the body of the report.

Random Number Generator

The Bayes Durham Shuffle ensures that the pseudo random numbers used in the simulation are further shuffled, ensuring minimal correlation between subsequent random outputs.

// Some example code here…