THIS PAGE CAN BE DELETED FROM YOUR FINAL REPORT

(ALL text in blue can be deleted from your final submission)

T­he following guidelines **must** be adhered to:

**Writing Style**Your report should be written from a third person, observational perspective. Do not use the following terms:

|  |  |  |  |
| --- | --- | --- | --- |
| I  Me  My  Mine  We  I believe  In my opinion  According to me | Our  Us  Ours  Myself  We're  I feel  As I see it | I'm  Ourselves  We've  I know  As far as I'm concerned  I think  Personally | I have  I will  We can  Our experience  From my point of view  In my perspective |

Do use or model from the following examples which are written from an observational third person perspective:

|  |  |
| --- | --- |
| "The researcher" or "the observer"  "It was observed that..."  "The data suggest..."  "The experiment revealed..."  "The results show..."  "One can infer that..."  "The study indicates..."  "The findings imply..."  "The evidence supports..."  "An analysis of the data" | "The procedure was executed as follows..."  "The measurements were taken..."  "The subjects/participants behaved in a certain way..."  "The test conditions were controlled..."  "The trends in the data"  "The observations were made by the team."  "The process was documented as..."  "The phenomenon was recorded."  "A statistical analysis was conducted."  "The data were collected and analysed." |

THIS PAGE CAN BE DELETED FROM YOUR FINAL REPORT

(ALL text in blue can be deleted from your final submission)

**Report Layout and Style**

T­he following guidelines **must** be adhered to:

* All **text will be black** in the report unless in very exceptional circumstance.
* Main body text must be at least **11pt font** using either **Arial or Calibri** font.
* Main body text will have **1.5 line spacing**.
* **Margins** will be a minimum of **2 cm on each side**.
* All **pages** will be **numbered consecutively**.
* Start **each section on a new page**.
* **Figures** must have **captions** and be **numbered** (e.g., Figure 1).
* **Tables** must have **captions** and be **numbered** (e.g., Table 1).
* **Figures** may be **black and white**, or **colour**.

**Word Count**

* There is a **10,000-word limit** for the project. This is a maximum, not a target. Shorter is often better.
* All words in the main body, excluding words in figures and tables, will be included in the word count.
* Consider what can be removed from the main body and placed in an appendix to reduce the word count as not every detail may be necessary in the main document.

**Copyright and Intellectual Property Rights (IPR)**

Your report should be written considering that it will be within the public domain. Normally, you retain copyright over your written work and Intellectual Property Rights (IPR) over any technical work. There are rare situations where this might not be as simple, for example when working with a company, external client, or on a larger university project. There are strategies you can utilise:

* You can inform your supervisory team that the project cannot be made available to other students undertaking an MSc Project.
* You can provide a shortened report for sharing that does not contain the protected information.
* You can assign IPR to the external collaborator, although you should take great care when doing so. It is best to speak to your supervisor.
* Non-disclosure agreements can be made between the external collaborator and the supervision team.

THIS PAGE CAN BE DELETED FROM YOUR FINAL REPORT

(ALL text in blue can be deleted from your final submission)

Summary of the project sections

1. **Introduction – setting the scene.**
   1. Problem statement.
   2. Aims and objectives.
   3. Motivation/background.
   4. Legal
   5. Social
   6. Ethical
   7. Professional
   8. Report outline.
2. **Literature - Technology review –** a thorough investigation of the problem background and options for solving it.
3. **Methodology –** Given the findings of the previous sections, which methods and approaches may be suitable for your problem, and which one will you choose?
4. **Implementation –** How was methodology described in the previous section applied in order to solve your specific problem? i.e. A description of what you did.
5. **Results –** What is the final outcome of results the of your implementation and evaluation?   
     
   **Evaluation –** How was your implementation validated? Described the steps that you took to evaluate the extent to which your implementation has met the aims of the project.   
     
   **Work of Others** - How do your results and evaluation compare to the work of others?
6. **Conclusion –** a summary of your main findings, a personal reflection and proposal for future work.
7. **References**
8. **Appendices**

Emergent Dialogue and Quest Generation using LLMs

By

Rezwan Rahman (RAH22529097)

Submitted to

**The University of Roehampton**

In partial fulfilment of the requirements

for the degree of

**BACHELOR OF SCIENCE IN COMPUTING**

**Date:** 19/12/2024

Signed (apply signature below)

**Rezwan Rahman**

**Declaration**

I hereby certify that this report constitutes my own work, that where the language of others is used, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions, or writings of others.

I declare that this report describes the original work that has not been previously presented for the award of any other degree of any other institution.

Acknowledgements

Here it is customary to thank the people who have supported this work and your studies in general. It is up to you who you thank!

Abstract

GUIDANCE: Up to 300 words

A short summary of your project to include the problem, the main literature reviewed, your implementation and your findings.

Write this after you have finished the entire report!

Table of Contents

[1. Introduction 7](#_Toc927214740)

[Research Question or Problem statement 8](#_Toc522749298)

[Aims 8](#_Toc1298644285)

[Objectives 8](#_Toc198637622)

[Legal, Social, Ethical and Professional Considerations 8](#_Toc960797214)

[Background 8](#_Toc273743648)

[Report overview 8](#_Toc210207254)

[2. Literature or Technology Review 8](#_Toc1335231035)

[3. Methodology 9](#_Toc1844834773)

[4. Implementation 10](#_Toc1739317238)

[5. Results 11](#_Toc826321870)

[Evaluation 11](#_Toc1085198293)

[Related Work 12](#_Toc55229855)

[6. Conclusion 12](#_Toc253686570)

[Reflection 13](#_Toc1734103406)

[Future Work 13](#_Toc542248490)

[7. References 13](#_Toc1743583894)

[8. Appendices 14](#_Toc649705763)

# **Introduction**

GUIDANCE: Up to 1000 words  
An initial paragraph introducing the project context and idea.

## Problem Description, Context and Motivation

Your research question or problem statement should include the following considerations:

* What is the problem you are addressing?
* Who is affected by the problem?
* Where and/or when does the problem occur?
* Why is it important to solve the problem?

Problem Description

Context

Motivation

* The problem I’m trying to solve

## Aims

Aims are broad statements of desired outcomes and intentions. We expect about 3.

Enhance and Evolve Player Immersion through Dynamic Content Creation inside Games.

Innovate Game Development with AI Integration

Contribute to the Evolution of Interactive Storytelling

## Objectives

Objectives are the tangible steps that will be taken to achieve the aims i.e., tasks that will be done.

Design and Develop the RPG Framework.

Integrate the AI Language Models into the Framework.

Develop Content Moderation and Context Management.

Evaluate Player Experience and System Performance

Document the Development Process

## Legal

All projects will have legal considerations. You are marked on your ability to identify them and take any actions necessary.

## Social

All projects will have social considerations. You are marked on your ability to identify them and take any actions necessary.

## Ethical

All projects will have ethical considerations. You are marked on your ability to identify them and take any actions necessary.

## Professional

All projects will have professional considerations. You are marked on your ability to identify them and take any actions necessary.

Consider the issues that your project has under each heading. Discuss with your supervisor if you are not sure. If your project required ethical clearance, then this must be clearly stated in this section, including how the ethical considerations were managed.

## Background

Provide the reader with the relevant contextual information necessary to understand why your work is important. You may reference literature here and/or your personal motivation. You should describe the context of your work (for example if it is relevant to a particular area of business, medicine, education etc) and how it addresses known issues in that context. This section is expected to have at least some references to external sources.

## Report overview

Describe the upcoming sections in order -- this provides your reader with a roadmap of the report.

**Refer to the Project Report Builder on Moodle for content that you should include in this section.**

# **Literature - Technology Review**

GUIDANCE: Up to 1500 words

**Note:** All projects will require a technology review. However, not all projects will require a literature review. Please speak to your supervisor to confirm what is required for your project.

## Literature Review

The literature review is a key part of your project and has specific marking criteria attached. The purpose of the literature review is to show that you have researched, understood, and evaluated all aspects of your problem statement. This is your ‘investigation’.

You should describe the problem in detail, mentioning any organisations, institutions or individuals who are important in the context or the solution.

You should correctly reference key articles, books, or papers you have read that are relevant to the problem.

It is not enough to list relevant points. You also need to show the ability to:

* Summarise the main points.
* Discuss (e.g., point out strengths, weaknesses, differences of approach etc).
* Describe how the literature you have found relates to the problem you are addressing.
* Apply your own critical thinking to the material you include.
* Draw your own conclusions from your investigation and show how these will guide your methodology and implementation.

Everything that you write in your literature review must be clearly relevant to your problem statement and aims and objectives, and you must clearly state this.

### Generative Agents: Interactive Simulacra of Human Behaviour

The aim of the study was to develop generative agents which can simulate believable human behaviour. Some of the key research questions asked in these papers includes MemoRAG: Moving towards Next-Gen RAG Via Memory-Inspired Knowledge Discovery

1. How can generative agents simulate realistic individual and social behaviours over time?
2. What architecture is required to ensure agents maintain long-term coherence in memory, behaviour, and interaction?
3. How can emergent social dynamic arise from interactions between generative agents?

#### Methods Used

The researchers created a sandbox environment, which is inspired by the game "The Sims" which is populated with 25 generative agents.

##### Agent Architecture

The researchers integrated a memory stream framework for storing experiences, reflection mechanisms for higher-order generalizations and a dynamic planning system.

##### Implementation

They used ChatGPT to enable agents to perceive their environment, retrieve memories and synthesize plans and engage in various interactions.

##### Evaluation

The researchers have tested agents’ ability to maintain self-knowledge, retrieve memories, generate plans, react and reflect in isolation. They have also observed emergent social dynamics over two simulated days.

##### Interactive Features

The researchers have also designed agents to respond to user commands, adapt to environmental changes as well as demonstrate evolving relationships and behaviours.

### Key Findings and Conclusions

##### Agent Coherence

The architecture integration of observation, reflection and planning proved essential for generating believable behaviour and interactions. Removing the framework would significantly reduce agent performance.

##### Emergent Dynamics

Social phenomena, such as information diffusion and relationship formation arose organically from agent interactions. For example, when a single agent was instructed to host a Valentine's Day event, it led to a town wide party. As information about Valentine’s Day spread.

##### Limitations

Some challenges that occurred was the failure of memory retrieval, fabrication of events and occasional unnatural conversational styles due to the language model.

##### Applications and Implications

Some applications span gaming, training simulations and prototyping social systems. Some ethical issues can be considered such as mitigating parasocial relationships and addressing the risk of misuse.

### MemoRAG: Moving towards Next-Gen RAG Via Memory-Inspired Knowledge Discovery

This study introduces a new concept called MemoRAG, an advanced Retrieval-Augmented Generation (RAG) framework aimed at overcoming limitations of traditional RAG systems.

The Key Research objectives

1. How can a memory-inspired approach improve retrieval and generation for ambiguous or complex tasks?

2. Can long-context tasks involve implicit queries and distributed evidence be handled effectively using global memory mechanisms.

3. What architecture and training paradigms optimize the balance between memory efficiency and task-specific accuracy?

#### Methods Used (Experiments, Modelling and Frameworks)

The implementation of MemoRAG involves a dual architecture

* A light LLM for global memory creation over extended contexts.
* A heavy LLM for retrieval augmented generation tasks.

As well as a memory module which has progressive compression of input tokens into a compact memory token while preserving semantic integrity.

Pre-trained on large datasets such as RedPajama to form memory structures.

They evaluated the models using benchmarks such as UltraDomain, this includes complex tasks from diverse domains such as law, finance and education.

#### Key Findings and Conclusions

##### Enhanced Retrieval and Generation

MemoRAG significantly outperformed RAG systems, especially in tasks which require implicit reasoning or long-context integration.

##### Applicability Across Domains

Showcased robust performance across various domains.

##### Challenges and Future Directions

While MemoRAG excels in many areas. Some challenges include scalability with extremely high context lengths.

## Technology Review

The technology review focuses on technology that will be and could be used for the project. Firstly, you should review and summarise different technology options you could apply to your project. Then, you should write down the rationale for your own choice.

### Game Engine

#### Godot (GDScript)

##### Benefits

* Lightweight and Open Source
* Ideal for 2D Games but has options for 3D games.
* Extensive community support and tutorials for indie developers
* Built-in scripting language (GDScript) facilitates quick prototyping.

##### Limitations

* Performance can be suboptimal for large-scale 3D games.
* Smaller community compared to Unreal Engine.

##### Overview

Godot offers a lightweight, open-source game engine curated for indie developers. The engine is primarily suited for 2D game development but provides solid ground for 3D Games. Because of Godot's open-source and simple-to-learn identity, it has garnered a community committed to producing content and guides for the game engine.

The Godot engine also comes with its own scripting language, GDScript, which is a Python-like programming language specifically built for the Godot engine. Because of GDScript's simplicity, it allows for quick game prototyping while also meeting game performance demands.

##### Games Developed in Godot

1. Endoparasitic

2. Brotato

3. Road to Vostok

#### Unreal Engine (Blueprint)

##### Benefits

* High-fidelity rendering capabilities for immersive visuals
* Visual scripting Blueprint simplifies logic creation for non-programmers.
* Large library of pre-built assets and plugins

##### Limitations

* Steep learning curve
* Resource-intensive; requires high-performance hardware.

##### Overview

Unreal Engine is a highly regarded and widely used game engine in the gaming industry. It excels in high visual fidelity, with its out-of-the-world immersion. Unreal Engine is more suited for AAA developers and large teams. Unreal Engine primarily attracts game developers who want to make games which have cutting-edge graphics and are generally large-scale projects.

Unreal Engine, like Godot, includes its programming language called Blueprint.

Blueprint is a visual programming language which is designed for ease of use and accessibility for developers who may not have a strong programming background. Unreal Engine also supports C++ directly out of the box when developers require more control and functionality. Unreal allows developers to use these two programming languages interchangeably.

##### Games Developed in Unreal Engine

1. Fortnite
2. Hellblade: Senua’s Sacrifice
3. Black Myth: Wukong
4. Bioshock Infinite
5. A Plague Tale: Innocence

#### Rust

##### Benefits

* Memory safety without a garbage collector
* High performance for backend systems game logic.
* Strong community focus on correctness and reliability.

##### Limitations

* Steeper learning curve for developers unfamiliar with low-level programming
* Smaller ecosystem compared to C++

##### Overview

Rust is generally a new programming language often seen as a modern version of C/C++. Rust excels in memory-safe and high-performance code.

Rust is a general-purpose system-level programming language that excels in areas that require low-level control. Rust excels in memory safety and high-performance projects. This is due to the unique nature of the programming language, as it breaks many traditional features of various programming languages.

Rust emphasises memory safety with a strict ownership system and borrows a checker. This allows the prevention of null pointers dangling references, and data races at compile time. Through this system, rust can guarantee memory safety.

Rust has also adopted a modern syntax style and has many benefits such as fearless concurrency as its ownership and borrow checker provide a built-in safety for this style of programming.

##### Language Features

* No Raw Pointers in Safe code
* Relies on Ownership, Borrowing and Lifetimes
* Pattern Matching, An Expressive `match` statement
* Crates and Packages Ecosystem behind the Cargo Package Manager which is modern and user-friendly.

#### C++

##### Benefits

* Widely used in game development, particularly for Unreal Engine.
* Established libraries and frameworks for gaming and AI.
* Flexibility and extensive tooling.

##### Limitation

* Prone to memory management errors.
* Longer development times compared to higher-level languages.

##### Overview

C/C++ is another high-level general-purpose programming language. C++ is the superset of C and follows the object-orientated paradigm. C++ allows manual memory management using keywords such as new and delete or using smart pointers. But this system increases the vulnerability of memory leaks, dangling pointers and data races unless the programmer takes explicit precautions to avoid these problems.

C/C++ is an easier language to get started but is much harder to master due to undefined behaviours and complexity in managing memory and low-level operations.

##### Language Feature

* Extensive support for low-level programming using raw pointers and direct hardware interaction.
* Has a rich and complex feature set, including templates, multiple inheritance and polymorphism.
* Package management using tools such as Conan and vcpkg but isn't integrated or standardized as Rust's Cargo.

### LLM Technologies

#### Ollama

##### Benefits

* Easy integration with existing Projects.
* Focuses on secure and privacy-conscious AI deployment.

##### Limitation

* Limited scalability for highly complex models.

Ollama is an open-source project which serves as a powerful and user-friendly platform for running LLMs on local machines. Ollama allows users to run open-source LLMs such as Llama, Qwen, Mistral and many freer and locally on a user's machine.

How Does Ollama Work?

Ollama bundles all the model information into a single package which is defined by a Modelfile.

Ollama will be used in this project to have local access to LLM APIs while avoiding API costs and allowing the configuration of the LLM later if need be.

The benefits of Using Ollama are

* Enhanced Privacy and Data Security
* No Reliance on Cloud Services
* Customization and Flexibility
* Offline Access
* Cost Savings

#### LangChain

##### Benefits

* Designed specifically for building AI-powered applications.
* Robust support for prompt engineering and memory management.

##### Limitations

* Can have a steep learning curve for developers who are new to AI.

##### Overview

LangChain is a software framework which allows the integration of large language models into applications. LangChain use cases largely overlap with document analysis and summarization, chatbots and code analysis.

LangChain provides a centralized development environment which allows developers to manipulate various data structures, model chaining, and integrate external models and applications with external data sources, while also allowing them to break down issues into smaller sub-tasks.

#### HuggingFace

##### Benefits

* Access to pre-trained models for NLP tasks.
* Strong community support and extensive documentation.
* Flexibility to fine-tune Models

##### Limitations

* Require significant computational resources for large-scale deployments.

##### Overview

Like Ollama, HuggingFace provides a platform to create and browse AI models.

HuggingFace is an open-source data science and machine learning platform, and acts like a hub for people working in the field of AI and Machine Learning.

### Version Control and Collaboration Tools

#### Git

Git is another open source software, primarily as a version control system which helps developers keep track of changes in their code and project. Git is designed to be fast and performant. It is widely adopted by developers and is simple to use.

#### GitHub

As shown above, we use Git to keep track of changes and edits in the project, but we want to store those changes somewhere, this is where GitHub comes into play. GitHub provides a platform to store, share and collaborate code. It also allows the user to track and manage changes, get feedback and use some exclusive features inside of GitHub such as,

* GitHub Project for Project management
* GitHub Wiki for Knowledge Bases for projects
* Bug Tracking

### Journalling and Knowledge Base

#### GitHub Wiki

GitHub Wiki is a place to host documentation for a given project, allowing developers to write notes and host alongside the documentation.

The purpose of GitHub Wiki is to share long-form content about the project, such as how to use, install and how it was designed and its core principles and philosophies.

#### Obsidian

For my final year project, I will keep random notes and information in my Obsidian Vault. Obsidian is a note-taking app which helps users organize thoughts and knowledge.

I will primarily use Obsidian to write and draw designs and journal my activities for my final year project. Drafts and Documentation can also be found in the Obsidian Vault.

### Diagramming and Visualization

##### Excalidraw

Excalidraw is a free and open-source diagramming software, used for basic diagrams, if more advanced shapes and diagrams are required, users can install packages. Excalidraw is easily interactable on Obsidian, which is why I'll be using this to draw my diagrams.

##### Draw.io

Draw.io is another free and open-source option, this has been stable for me when for this degree. Using it to draw things such as Flow Charts Entity Relation Diagrams and many other diagrams, this software has all the tools required to make well-made diagrams but is somewhat dated.

## Summary of Outcomes of Literature and Technology Review

Include two tables:

* A table that summarises the benefits and limitations of the literature that has been reviewed.
* A table that summarises the benefits and limitations of the technologies that have been reviewed.

Critically analyse the content in each table and identify how the outcomes of your critical analysis will influence your methodology and your project overall.

**Refer to the Project Report Builder on Moodle for content that you should include in this section.**

|  |  |  |
| --- | --- | --- |
| **Paper** | Benefits | Limitations |
| Generative Agents: Interactive Simulacra of Human Behaviour |  |  |
| MemoRAG: Moving towards Next-Gen RAG Via Memory-Inspired Knowledge Discovery |  |  |

|  |  |  |
| --- | --- | --- |
| **Technology** | Benefits | Limitations |
| Godot |  |  |
| Unreal Engine |  |  |
| Rust |  |  |
| C++ |  |  |
| Ollama |  |  |
| LangChain |  |  |
| HuggingFace |  |  |
| Git |  |  |
| GitHub |  |  |
| GitHub Wiki |  |  |
| Obsidian |  |  |

# **Methodology**

GUIDANCE: Up to 1000 words

This section should answer the question -- **how are you going to undertake the project?**

Describe HOW you are going to create your artefact, including any tools, design methods, data gathering methods, algorithms etc that you are going to use. Tell us WHY you have chosen these methods in favour of others (with reference to the findings of your literature and technology review).

This section should include the following subheadings:

* Design
* Testing and Evaluation
* Project Management
* Technologies and Processes

**Refer to the Project Report Builder on Moodle for content that you should include in this section.**

This section outlines all the methods and techniques used to design, implement and evaluate the solution for emergent dialogues and quest generations. This methodology ensures reproducibility and clarity in the development process.

### Design and Methodology

This research is experimental as well as applied. There has been proof that this system can be applied and implemented.

#### Iterative Design

* **LLM Module** - Responsible for natural language understanding and generation.
* **RAG Systems** - Stores knowledge retrieval for dynamic and context-aware responses.
* **NPC Memory**
* Short Term Memory
  + - Tracks recent interactions to maintain conversational coherence.
* Long Term Memory
  + - Stores historical data for consistent NPC behaviour and personalization.

This is the current architecture of the LLM System. Currently working on the implementation of the LLM + RAG System to allow setup a long-term memory for NPCs and short-term memory. This is a WIP diagram and will be heavily updated in the future.

The project will implement a dual iterative design approach

1. Rapid Prototype Development Phase (Milestone 2):
   * Rapidly Develop and Implement Features
   * Functionality over optimizations.
   * Conduct minimal feature testing to establish feasibility
2. Refinement Phase

* Refactor the codebase to adhere to industry standards, including best practices and design patterns in rust and Godot.
* Optimize system performance based on feedback and testing insights.

#### Model Fine Tuning

After picking a model, the next-phase is to fine tune the model such that it will generate the expected response. The model will be trained to generate correct formatted responses, if it fails to abide by these strict rules it will be asked to generate until it is correct.

### Testing and Evaluation

For Milestone 2, the focus will be on rapid development and prototyping, this phase prioritizes on achieving a functional, integrated solution with minimal testing on the features. The main goal of Milestone 2 is to ensure that the experiment is feasible. Once the experiment proves successful and the solution is verified to work as intended, the next phase will involve refactoring the codebase to align with industry-standards such as implementing design patterns and best practices.

#### Rust Test Cases

Test cases found in rust will primarily be unit test to ensure that each individual component works as intended. Testing various things such as returns of functions, making sure that correct inputs are being parsed.

#### GDUnit Test

GDUnit test will be used to test individual Godot scripts, to validate their correctness. Some examples is to test and verify player movement scripts.

GDUnit test also provides the ability to do integration testing this will validate the interaction between multiple scripts or scenes.

Integration test will be hosted on GDUnit as we can not test the integration test on rust as its a library.

Some Examples of Tests

* Scene Validation
* Behaviour Testing
* Edge Case Testing
* Regression Testing
* Performance Testing
* Mocking and Simulating Events

#### GitHub WorkFlow

GitHub is utilized to automate the testing process, including,

CI/CD Pipelines, which automatically run tests on code commits. Issue tracking to monitor and resolve bugs or feature requests, as well as milestone progression.

### Project Management

#### GitHub Projects

Using GitHub Projects to manage various requirements, difficulty of implementation.

I will also be using GitHub Milestones to manage different milestones. Will also be using the GitHub issues to create requirements which need to be finished as well as bugs that need to be patched.

### Technologies and Processes

#### Godot

Godot serves as the foundation and core of the project, functioning as the primary game engine. It will have game specific code and objects such as levels or scenes, characters and various game components, providing a unified environment for designing and testing the interactive game framework. Godot was selected because of its lightweight, open-source and ease of use nature. As well as its versatility and extensive support for 2D game development.

#### GDScript

GDScript will be the primary language for game specific logic, used to implement game logic within the Godot Engine. This scripting language was designed specifically for Godot, and has simple syntax similar to python, making the game development time quick.

#### Ollama

Ollama is responsible for managing the LLM which will be integrated to the game. We will use Ollama's inbuilt API. It will be responsible for handling and responding to various requests. This will be the primary engine for the generating emergent and immersive dialogue and quests for the player to experience.

#### Git

Git will be used for its version control, enabling efficient management of code changes. GitFlow branching model will be used to separate features, bug fixes and releases for the project further enchaining its maintainability.

#### GitHub

GitHub will host the repository of the project, using LFS GitHub will store images in the LFS server. GitHub will also help with project management, issue tracking and ensuring that the software is safely stored inside another medium.

# **Implementation**

GUIDANCE: Up to 3000 words

Finally, you can tell us WHAT you did, i.e. How did you apply the methodologies you have described in the section above to your actual problem.

This part can be very descriptive but please avoid excessive detail.

Some strategies that can help you write this part:

* Choose a writing style (e.g., first, second, or third-person perspective).
* Start this section with any design work you might have done e.g., System design/architecture, UX design artefacts etc.
* If you divided your work into sprints, that can be a good structure for this section.
* Only include code snippets for particularly challenging parts of your implementation.
* Pick out a few difficult problems you had to solve and tell us in detail how you solved them. This brings your experience to life.

**Refer to the Project Report Builder on Moodle for content that you should include in this section.**

# **Evaluation and Results**

GUIDANCE: Up to 2000 words

This is an important section where you weigh up the strengths and weaknesses of your artefact.

Guidance: If your project has a user-facing element, we expect to see some kind of evaluation of this with representative intended users, for example a ‘think aloud’ usability test.

You can also apply standard metrics for the domain you are working in and see how you have done against them. **Your project does not have to be perfect -- indeed the outcomes might have been bad.** The point is you must evaluate the outcome and discuss its strengths and weaknesses.

This section should include the following subheadings:

* Related Work

**Refer to the Project Report Builder on Moodle for content that you should include in this section.**

# **Conclusion**

GUIDANCE: Up to 1500 words

The conclusion summarises the project. Start by summarising the overall outcome of your project and to what extent the aims and objectives have been met. You need to highlight your key outputs and/or discoveries.

The following subsections that must appear in your conclusion.

## Future Work

Answer the question -- **What next?**

You've completed a significant piece of work -- perhaps the largest piece of work you have ever done. But no project is ever 100% complete, and you will have found new ideas along the way. If someone were to pick up your project, what avenues should be explored next?

This is an important section, and it helps us understand what you have learned by doing the project and allows you to show you understand what a more ideal solution might look like, outside the constraints of the MSc Project timeframe.

## Reflection

You must critically reflect on the entire project process and how well you have worked on the project. What particular things have you learned during the project? Why were you able and unable to meet project goals? What would have you done differently in hindsight?

**Refer to the Project Report Builder on Moodle for content that you should include in this section.**

# **References**

In this section, you **must** reference any sources used in your work. Typically, these sources will have come up during the investigation and related work sections. Your referencing must use the IEEE referencing style .

It is **highly** recommended that you use reference management software such as RefWorks that is provided by the university. Your project should have as many references as is required. However, having few references indicates that no thorough investigation has occurred.

It is your responsibility to ensure that you have actually read all the material you reference, and that the references provided in your report are legitimate and **NOT AI generated**.

[1] J. Park *et al.*, “Generative Agents: Interactive Simulacra of Human Behavior,” vol. 23, 2023, doi: https://doi.org/10.1145/3586183.3606763.

[2] DevDuck, “How I Organize My 10k+ Line Godot Project!,” *YouTube*, Aug. 02, 2024. https://www.youtube.com/watch?v=4az0VX9ApcA (accessed Jan. 01, 2025).

[3] jeremychone-channel, “GitHub - jeremychone-channel/rust-base: Rust minimalistic base starter template.,” *GitHub*, 2022. https://github.com/jeremychone-channel/rust-base (accessed Jan. 01, 2025).

[4] Asana, “Process Documentation: A How-To Guide With Examples • Asana,” *Asana*, Nov. 21, 2022. https://asana.com/resources/process-documentation

[5] J. Martins, “What Are KPIs? • Asana,” *Asana*, Mar. 01, 2024. https://asana.com/resources/key-performance-indicator-kpi

[6] J. Martins, “Write Better SMART Goals - Tips with Examples,” *Asana*, Feb. 03, 2024. https://asana.com/resources/smart-goals

[7] A. Web Services, “What Is LangChain? - LangChain Explained - AWS,” *Amazon Web Services, Inc.* https://aws.amazon.com/what-is/langchain/

[8] R. Scarborough, “Writing Quality Code: Practicing ‘Make It Work, Make It Right, Make It Fast,’” *Keyhole Software*, Mar. 23, 2023. https://keyholesoftware.com/writing-quality-code-practicing-make-it-work-make-it-right-make-it-fast/

[9] H. Qian, P. Zhang, Z. Liu, K. Mao, and Z. Dou, “MemoRAG: Moving towards Next-Gen RAG Via Memory-Inspired Knowledge Discovery,” *arXiv.org*, 2024. https://arxiv.org/abs/2409.05591

[10] Wikipedia Contributors, “Test-driven development,” *Wikipedia*, Oct. 24, 2018. https://en.wikipedia.org/wiki/Test-driven\_development

# **Appendices**

Appendices appear after references. Your appendices depend on the nature of your project. **Do not assume people will read your appendices.** Even if you direct them to do so in your main text, appendices are considered additional information and should not be relied upon to understand your main body of work. Refer readers to an appendix using a phrase such as *see Appendix A for further details*.

The following documents **must** be included as references:

* Your Project Proposal.
* Evidence of your use of a project management tool.
* A description of how to access any technical output. **It is strongly recommended you use GitHub or something similar to do this.**

Any important communications between you and external stakeholders -- **please ensure private data is removed and communications anonymised.**