# A Metaheuristic Approach for Multiple Sequence Alignment

**October 2024**

**By**

**Peter Moorhouse**

**Student number 202247615**

**Word count: 1945**

# Contents

1. Project background and purpose ..... 3
1.1. Introduction ..... 3
1.2. Objectives ..... 3
1.2.1 Primary Objectives ..... 3
1.2.2 Secondary Objectives ..... 4
1.3. Scope ..... 4
1.4. Deliverables ..... 5
1.4.1 Software (update dates) (could be a table?) ..... 5
1.4.2 Documents ..... 5
1.5. Constraints ..... 6
1.6. Assumptions ..... 6
2. Project rationale and operation ..... 7
2.1. Project benefits ..... 7
2.2. Project operation ..... 7
2.3. Options ..... 8
2.3.1 Programming Languages ..... 8
2.3.2 Algorithm Design ..... 8
2.4. Risk analysis and mitigation ..... 8
2.4.1 Risk Matrix ..... 8
2.4.2 Risk Analysis ..... 9
2.5. Ethical and legal considerations (needs citations) ..... 11
2.6. Commercial considerations ..... 11
3. Project methodology and outcomes ..... 12
3.1. Initial project plan ..... 12
3.1.1. Tasks and milestones ..... 12
3.1.2. Schedule Gantt chart ..... 14
3.2. Project control ..... 15
3.3. Project evaluation ..... 15
4. References ..... 16
5. Appendix a ..... 17

# 1. Project background and purpose

## 1.1. Introduction

In this project, a software tool will be developed to tackle the multi-objective optimization problem of Multiple Sequence Alignment (MSA) - a common analysis task in Bioinformatics. The tool will aim to produce high-quality alignments of biological sequences in a time-efficient manner.

## 1.2. Objectives

### 1.2.1 Primary Objectives

**1. Perform Multiple Sequence Alignment in a Time-Efficient Manner**

The produced software should be able to align a typical testcase of 6 protein sequences within 10 seconds on a university desktop computer. The resulting alignment of sequences must be a valid solution – conserving the original sequence content and identifiers given as input.

**2. Assess the Viability of a Single-State Approach for Iterative Alignment**

To address their underrepresentation in recent studies, a single-state metaheuristic algorithm such as 'Simulated Annealing' should be implemented and assessed in its ability to guide an effective optimization process for MSA. A single-state form of the software could be contrasted against a population-based approach or assessed relative to an external tool such as Clustal Omega.

**3. Support Established Bioinformatics File Formats**

The alignment tool should be able to read biological sequences from an established file format such as FASTA. Likewise, the tool should support an established file format for outputting sequence alignments, such as FASTA, PHYLIP or NEXUS. The user should be able to specify the input source and output destination as command-line arguments.

**4. Consistently Produce High-Quality Alignments of Sequences**

As assessed in a case study using structural benchmarking, the alignment tool should demonstrate the ability to consistently produce alignments of a comparable quality to established software packages such as Clustal Omega and MUSCLE. (TODO: add that this is informed by experts)

### 1.2.2 Secondary Objectives

**5. Output a Set of Alignments Offering Compromises Between Objectives**

Keeping step with recent research, the software should leverage multiple objective functions to guide the pareto-optimization process. As output, the tool should produce a non-dominated set of at least 5 alignments that represent different compromises between the objectives.

**6. Support Batch Alignment of Multiple Files**

The alignment tool should support the alignment of a series of input files from a specified directory. The user should be able to specify a source and destination directory using command-line arguments. The software should work through each input in sequence and output the resulting alignments to the destination directory.

**7. Indicate Progress in Aligning Sequences**

As the software may need to process a set of sequences for multiple seconds at a time, the software should display a clear indicator of how much progress has been made on the current alignment. For example, the program could present a progress bar using ASCII characters.

## 1.3. Scope

This project will entail the development of software that performs the specialist task of Multiple Sequence Alignment (MSA). The alignment tool will be developed using an agile methodology and leverage metaheuristic algorithms to tackle the MSA problem. A series of experiments will be undertaken with the goal of improving each successive iteration of the software – to be released at intervals alongside details of its performance on benchmark testcases.

The project should conclude with a comparative case study, comparing the performance of developed tool against established alternatives such as MAFFT, Muscle and ClustalOmega.

Despite being a key feature of some sequence alignment packages such as ClustalX, the development of a rich graphical user interface (GUI) lies outside the scope of this project. Instead, emphasis is placed on producing high-quality alignments in a time-efficient manner.

While a number of studies have explored metaheuristic approaches for MSA, this project aims to address the underrepresentation of single-state methods in recent research. Further, the project offers opportunity to explore novel combinations of objective functions to guide the optimization process.

## 1.4. Deliverables

### 1.4.1 Software (update dates) (could be a table?)

A series of iterations of a metaheuristic alignment tool 'MAli' will be released on GitHub using an iterative development methodology. A new version will be released at the end of each two-week sprint of development. The individual iterations are outlined as follows:

**MAli v0.1** (November 5th, 2024)

- Should be able to read sequences from a suitable file format and perform MSA using a simple strategy to produce a valid (possibly low-quality) alignment as output.

**MAli v0.2** (November 19th, 2024)

- Should demonstrate iterative alignment using metaheuristic algorithms as a 'proof-of-concept', producing higher quality alignments than those of v0.1.

**MAli v1.0** (December 3rd, 2024)

- Should present a full implementation of an iterative alignment tool, improving on v0.2 in either solution quality or time-efficiency.

**MAli v1.1** (January 7th, 2025)

- Should result from experimentation on the design from v1.0, with the goal of improving solution quality or time-efficiency.

**MAli v1.2** (January 21st, 2025)

- Should approximate the 'pareto front' of MSA solutions, producing a set of high-quality alignments as output for a decision maker to choose from.

**MAli v1.3** (February 4th, 2025)

- Should fulfil the documented functional and non-function requirements and offer a performance or quality improvement relative to all previous iterations of the software.

### 1.4.2 Documents

Two key documents will be produced as part of the project:

**Project Definition Document** (October 15th, 2024)

- An unambiguous document (2500 words max.) which clarifies the scope, objectives, and deliverables of the project. The document should propose an overall timeline and evidence consideration of risks, mitigations and ethical concerns.

**Final Report** (April 1st, 2025)

- A comprehensive report (15000 words max.) which details key elements of the project and documents efforts to meet the objectives outlined. The report should include a literature review and conclude with a critical evaluation of the project.

## 1.5. Constraints

In order to be a viable choice of alignment tool for bioinformaticians, the software must be compatible with established bioinformatics file formats.

With existing alignment tools in mind, a suitable format should be chosen for reading sets of sequences as input, and outputting alignments of sequences respectively. Potential formats to consider for outputting alignments include FASTA, PHYLIP and NEXUS.

## 1.6. Assumptions

This work is predicated on the assumption that the performance of alignment software as assessed via structural benchmarking is indicative of the tool's real-world performance at Multiple Sequence Alignment (MSA). An assumption of this nature is necessary as none of the project staff are bioinformaticians, and an external review cannot be commissioned due to financial constraints.

As described by Thompson et al. (2001), structural benchmarks are designed to offer a comprehensive evaluation for sequence alignment software. Today, their use is prevalent in the literature. In a review of 45 recent papers, structural benchmarking was found to be the most popular quality measure for MSA (Ibrahim et al., 2024).

# 2. Project rationale and operation

## 2.1. Project benefits

If successful, the project will provide an evidenced perspective on the viability of a single-state approach for iterative sequence alignment - a gap in recent research. This could draw attention to single-state methods as candidates for further research. Further, the project has scope to contribute to the current understanding of pareto-optimization for MSA, as a new combination of objective functions could be found to be highly effective.

Should the software be shown to produce solutions of sufficiently high quality, the aligner could serve a role alongside other software packages in consensus-based sequence alignment or see direct use as a preference choice by some bioinformaticians. In both these scenarios, the project could serve to improve the accuracy of bioinformatics analysis processes dependent on sequence alignment.

## 2.2. Project operation

An iterative methodology will be used for the development of the software. Drawing from the SCRUM framework (cite scrum here), development will take place in successive two-week sprints. Each sprint will conclude with a new iteration of the software being released, with changes being driven by the sprint goal. Such goals might be to experiment with new objective functions, or to pursue improvements in time efficiency.

The experiments conducted as part of the project stand to benefit from a wealth of literature on metaheuristic algorithms and approaches to the MSA problem. Further, the software can be tested for performance improvements by leveraging publicly available structural benchmark datasets which provide ‘gold standard’ reference solutions manually constructed by experts.

In addition to testing the functional and non-functional requirements of the software, a comparative study will be used to assess the final product in context of established alternatives such as MUSCLE and MAFFT (cites needed).

## 2.3. Options

### 2.3.1 Programming Languages

The choice of programming language will likely be informed by the high-level design of the software, after having captured the requirements. A simpler design may be feasible in C++, which may offer performance benefits over other languages. A design with greater complexity, or clear opportunities for unit testing may be a reason to use C# due to its rich ecosystem of supporting tools.

### 2.3.2 Algorithm Design

In designing the algorithm there are a wide range of metaheuristic strategies to choose from, each with different properties, design choices and parameters. In addition to the choice of a strategy, the configuration of its parameters also presents options to be considered. Further, the literature surrounding MSA offers a selection of objective functions for the problem. (evidence this?)

These many different areas of decision-making present a significant challenge in developing a proficient alignment tool. It is for this reason that an iterative development methodology has been proposed. With each successive sprint of development, experiments can be conducted to identify ways to improve on the algorithm design, with the goal of arriving at a strong alignment tool to conclude development.

## 2.4. Risk analysis and mitigation

### 2.4.1 Risk Matrix

| | | Likelihood (L) | | |
|---|---|---|---|---|
| | | 1 - Low | 2 - Medium | 3 - High |
| Severity (S) | 1 - Very Low | 1 | 2 | 3 |
| | 2 - Low | 2 | 4 | 6 |
| | 3 - Medium | 3 | 6 | 9 |
| | 4 - High | 4 | 8 | 12 |
| | 5 - Very High | 5 | 10 | 15 |

***Figure** A risk matrix showing how Risk Impact can be estimated using Likelihood and Severity.*

### 2.4.2 Risk Analysis

| Risk | Raw Risk | Mitigation | L | S | Residual Risk |
|---|---|---|---|---|---|
| Hard disk failure | 1 x 5 = 5 | Proactive: Make use of cloud storage for key files relating to the project wherever possible. Maintaining copies of files across multiple storage platforms (e.g. both OneDrive and GitHub) will further reduce the risk of losing significant amounts of work. | 1 | 1 | 1 x 1 = 1 |
| Poor time management | 3 x 4 = 12 | Proactive: Refer to the project Gantt chart, deliverables and milestones to understand whether the project is 'on-schedule'. Maintain a progress log & aim for transparent communication of progress with the project supervisor. | 2 | 2 | 2 x 2 = 4 |
| Poor project planning | 2 x 3 = 6 | Proactive: Try to break tasks down until they are shorter than two weeks in duration. Discuss these tasks with the project supervisor and agree on clear milestones to indicate progress. | 2 | 2 | 2 x 2 = 4 |
| Final product fails testing due to bugs | 2 x 5 = 10 | Reactive: Since an iterative development methodology is in place, select a previous iteration of the software to be used as the final version. Fix the bug if sufficient time is available. | 2 | 2 | 2 x 2 = 4 |
| Insufficient documentation for use of the software | 2 x 5 = 10 | Proactive: All released iterations of the software must include a clear explanation of the software functionality and directions for use. This information should be in the form of a 'README' file and/or available within the software interface. | 1 | 3 | 1 x 3 = 3 |

(continues on the next page)

| Risk | Raw Risk | Mitigation | L | S | Residual Risk |
|---|---|---|---|---|---|
| None of the software iterations produce valid solutions to the MSA problem | 2 x 4 = 8 | Proactive: Ensure that producing valid solutions to the MSA problem is one of the first requirements to be satisfied by a software release. This should mean that a functional tool is always available to fall back on, while following iterations can aim to improve the performance and solution quality. | 1 | 4 | 1 x 4 = 4 |
| Personal circumstances (e.g. hospitalised) disrupt productivity | 1 x 5 = 5 | Reactive: Communicate these circumstances with the project supervisor as soon as possible. Discuss how the project plan can be adapted if necessary and get in touch with the student services. | 1 | 4 | 1 x 4 = 4 |
| Project supervisor becomes unavailable | 1 x 4 = 4 | Reactive: Discuss this circumstance with the module lead if this situation arises. | 1 | 2 | 1 x 2 = 2 |
| Social restrictions due to an epidemic impact ability to work effectively | 1 x 3 = 3 | Reactive: Work remotely using cloud services. Check whether completion of all primary objectives is still feasible and discuss making revisions to the project plan if necessary. | 1 | 2 | 1 x 2 = 2 |
| Scope creep results in the project being unfinished | 2 x 3 = 6 | Proactive: Work with the project supervisor to create a comprehensive set of primary and secondary objectives for the project. Ensure that any work taken on aligns with these pre-defined objectives. | 1 | 2 | 1 x 2 = 2 |
| External data sets (for testing) become unavailable | 1 x 3 = 3 | Reactive: Generate synthetic test data for testing by writing a script or create a set of simple testcases by hand. Test the software using this data instead and communicate this compromise. | 1 | 2 | 1 x 2 = 2 |

## 2.5. Ethical and legal considerations (needs citations)

This project has no foreseeable ethical implications and complies with relevant legislation such as The Computer Misuse Act. In compliance with the Data Protection Act, the project will not involve test subjects or sensitive user data. All data to be used with the software will be either entirely synthetic or sourced from named public datasets with clear licenses.

Transparency and reproducibility are highly relevant to this work. While alignment tools are typically non-deterministic in nature, a deliberate effort will be made to communicate the methodology of all experiments undertaken as part of the project to support reproduction of results. Such details will likely include the initial settings and versioning for the tool, with reference to named test cases or datasets where feasible.

## 2.6. Commercial considerations

If undertaken independently from the University of Hull, an estimated cost for this project is £11663.00. A breakdown of this estimate is presented in the table below. The key considerations for expenditure were project staff, software subscriptions and access to journal articles.

| Title | Rate | Quantity | Total Cost |
|---:|---:|---:|---:|
| Undergraduate Researcher | £16.00/h | 400 hours | £6400.00 |
| Project Supervisor - Level 9 | £110.00/h | 20 hours | £2200.00 |
| GitHub Enterprise | £16.04/mo* | 9 months | £144.36 |
| Visual Studio Enterprise | £190.96/mo* | 9 months | £1718.64 |
| Literature Access Budget | £1200.00 | -- | £1200.00 |
| | | **Total Cost** | **£11663.00** |

*Costs converted to GBP from United States Dollar (USD)

***Table 2.6*** *Breakdown of estimated costs for the project – totalling £11663.00.*

While typically not directly monetised, work of this nature may be eligible for charitable funding as it offers potential contribution to bioinformatics and applied soft computing. In other scenarios, a project of this kind might be undertaken by a private entity to develop an in-house tool as a potential advantage over competitors – for example in the pharmaceutical industry.

# 3. Project methodology and outcomes

## 3.1. Initial project plan

### 3.1.1. Tasks and milestones

| Category | ID | Task/Milestone Details | Priority | Est. Duration |
|---|---|---|---|---|
| Planning | 1.1 | Define a set of aims and objectives to inform the project direction. | High | 3 days |
| Planning | 1.2 | Consider any causes for ethical concern. Document potential risks and appropriate mitigations. | Medium | 1 week |
| Planning | 1.3 | Break down and sequence project tasks to produce a Gantt chart of the project schedule. | Medium | 1 week |
| PDD | 2.1 | Write a section covering the project aims, objectives and scope. Define a set of deliverables. | High | 3 days |
| PDD | 2.2 | Write a section conveying the benefits of the project and how it will be conducted. | Medium | 3 days |
| PDD | 2.3 | Ask for feedback on current PDD draft, then make revisions. | Low | 1 day |
| PDD | 2.4 | Detail an account of how the project will be undertaken in terms of tasks and scheduling. | Medium | 3 days |
| PDD | 2.5 | Ensure that the document is well-presented and coherent. | Low | 3 days |
| PDD | M-A | **14th Oct, '24** - The Project Definition Document (PDD) has been completed. | Milestone | 3 weeks |
| Report | 3.1 | Write a section introducing the project including it's aims, objectives and context. | Low | 1 week |
| Report | 3.2 | Conduct a literature search, reading papers and building an overall narrative for the review. | High | 2 weeks |
| Report | 3.3 | Read relevant articles and write about metaheuristics and their applications. | Medium | 3 weeks |
| Report | 3.4 | Read relevant articles and write about the history of MSA software and approaches. | Medium | 3 weeks |
| Report | M-B | **25th Oct, '24** - The literature review has been completed. | Milestone | 5 weeks |
| Report | 4.1 | Capture a discrete set of requirements to direct the design and development of the tool. | High | 3 days |
| Report | 4.2 | Produce UML diagrams as high-level software design incl. Use Case, Package, Class diagrams. | Medium | 3 days |

***Figure 3.1.1a*** *– A list of tasks & milestones covering project planning, report writing and software design. (Table 1 of 2)*

| Category | ID | Task/Milestone Details | Priority | Est. Duration |
|---|---|---|---|---|
| Report | 4.3 | Ask for feedback on literature review and design sections, then make revisions. | Low | 2 weeks |
| Report | 4.4 | Write a technical background section on the MSA problem to provide context for readers. | Low | 2 weeks |
| Report | 4.5 | Evaluate the project as a whole, including comments on development and software testing. | High | 2 weeks |
| Report | 4.6 | Ask for feedback on technical background and evaluation sections, then make revisions. | Low | 2 weeks |
| Report | M-J | **1st Apr, '25** - The final report has been completed, giving a detailed account of the project. | Milestone | 6 months |
| Soft. Dev. | M-C | **12th Nov, '24** - MAli v0.1 is released - a basic tool that produces low quality, but valid solutions for MSA. | Milestone | 2 weeks |
| Soft. Dev. | M-D | **26th Nov, '24** - MAli v0.2 is released - introducing a metaheuristic algorithm to guide the alignment. | Milestone | 2 weeks |
| Soft. Dev. | M-E | **10th Dec, '24** - MAli v1.0 is released - an improvement on v0.2 and indicative of a full implementation. | Milestone | 2 weeks |
| Soft. Dev. | M-F | **14th Jan, '25** - MAli v1.1 is released - resulting from experimentation on the v1.0 design. | Milestone | 2 weeks |
| Soft. Dev. | M-G | **28th Jan, '25** - MAli v1.2 is released - producing a selection of high-quality solutions as output. | Milestone | 2 weeks |
| Soft. Dev. | M-H | **11th Feb, '25** - MAli v1.3 is released – fulfilling the defined requirements as a capstone of development. | Milestone | 2 weeks |
| Evaluation | 5.1 | Test the defined functional & non-functional requirements using a university desktop computer. | High | 1 week |
| Evaluation | 5.2 | Test the software alongside alternatives using structural benchmarking to allow for comparison. | Medium | 1 week |
| Evaluation | 5.3 | Analyse the performance and quality of the tools tested in 5.2. Discuss the results. | Medium | 1 week |
| Evaluation | M-I | **4th Mar, '25** - Software requirements have been tested and a case study has been completed. | Milestone | 3 weeks |

***Figure 3.1.1b*** *– A list of tasks & milestones covering report completion, development milestones and software evaluation. (Task List Table 2 of 2)*

### *A Note on Development Work Items*

Blah blah agile blah blah research blah blah a current backlog is given in 3.1.3 but this is subject to change because cite the agile manifesto.

## 3.1.2. Schedule Gantt chart

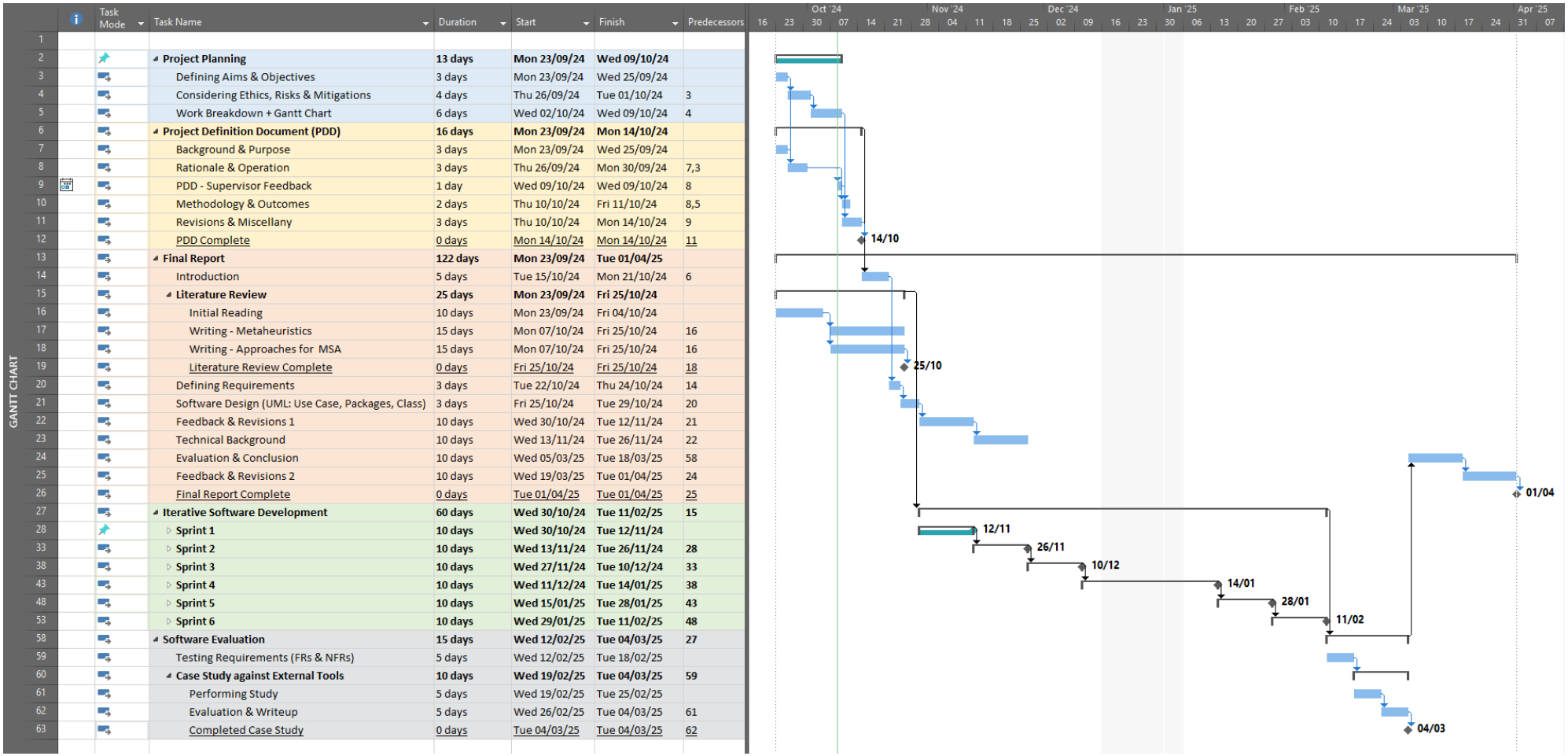


***Figure 3.1.2*** *– Gantt chart illustrating the planned schedule for the project. Individual development sprints have been collapsed for presentability.*

## 3.2. Project control

To manage the project effectively, upcoming tasks from the project schedule will be added to a separate task board (Trello) such that their progress can be monitored. Progress on tasks, especially deliverables and milestones will be discussed transparently with the project supervisor, and with reference to the project schedule. This will give opportunity to adapt the plan if necessary.

The solution quality and time-efficiency of the software will be assessed as part of each software release. The project will conclude with a comparative case study and an assessment on functional and non-functional requirements met. Following this, the overall success of the project can be evaluated with reference to the original aims and objectives. A successful project would see the development of a proficient alignment tool, compatible with existing bioinformatics file formats and accompanied by an evidenced account of its performance relative to available alternatives.

## 3.3. Project evaluation

How will you evaluate the project's artefacts and overall outcomes? What user evaluation will you do? Do not underestimate the importance of this, and include clear details of how you will do the evaluation. Remember that if you intend to test your outputs on people, you must declare this in your ethics review.

Delete the red paragraphs and replace this one with your content (use the "Normal" paragraph style).

# 4. References

List any sources you have used for your background and introduction here. Make sure you use the proper referencing format.

Delete the red paragraphs and replace this one with your content (use the “Normal” paragraph style).

# 5. Appendix a

You may use one or more appendices (label them “Appendix a” “Appendix b” and so on), to add useful reference information which may be relevant to other sections of the report. Do not use appendices simply as a way of writing more than will fit into the main document word count. If you don't need any appendices, then delete this whole section

Delete the red paragraphs and replace this one with your content (use the “Normal” paragraph style).