Statement of Work

# School of Computing Resource Maximisation Planning System

4 August 2024

**Proposer: Belinda Bergin**

**Team Members:**

- **Alex Boxall**
- **Edward Nivison**
- **Filip Mazur**
- **Hexuan Meng**
- **Matthew Cawley**
- **Rachel Cao**
- **Sineeha Kodwani**

# Background

The scheduling of tutorials at the Australian National University (ANU) is a key activity in the allocation of limited resources for the ANUs School of Computing (SoCo).

The SoCo Education Project Officer, Belinda Bergin, is required to allocate these resources to teaching activities respective to resource limitations whilst meeting course and teaching requirements. The solution in place to address this operational task is a complex time-consuming activity that requires significant effort to produce a schedule that addresses resource constraints and course requirements.

Whilst not limited to the SoCo, the current solution highlights its own drawbacks in its usability when scheduling resources at a large scale. The existing platform adopted by the ANU fails to create a streamlined user experience to achieve this crucial aspect of university operations.

The project vision, in alignment with broad ANU objectives of operational efficiency, student support and to provide each student with an equitable physical learning environment respective to their course, degree and resource requirements.

To achieve the vision and assist the SoCo in scheduling teaching activities, the “School of Computing Resource Maximisation Planning System for Labs” project was formed in early 2024. This project aims to address the usability challenges of the current protracted scheduling process, by creating a user-friendly medium for administrative staff to employ an algorithm to solve this optimisation problem. By considering several constraints such as course requirements, staff availability, room capacity and available technology, the algorithm seeks to provide the best space and resource efficient timetable.

By collaborating with relevant stakeholders, the project will provide a tool which can integrate into the existing ecosystem, allowing administrative staff to efficiently create timetables. The resulting timetables aim to provide students with tutorials that meet course and educational requirements, and teaching staff with appropriate working schedules and conditions.

Addressing these challenges faced by the SoCo Education Project Officer and administrative staff, this team can create the foundations of a tool which promotes a resource conscious culture without sacrificing on flexibility. The creation of this tool for the SoCo establishes the basis for wider adoption across the ANU to improve the allocation of educational resources for greater number of students.

## Overview

The SoCo Resource Maximisation Planning System for Labs aims to implement a solution for the scheduling labs for the SoCo. By assessing predefined course information, requirements, requests and constraints, the solution will generate a timetable which addresses each aspect to the best of its ability. The proposed solution will be significantly more time efficient and resource efficient than the current process of physically creating a timetable with minimal technological assistance.

## Project Objectives

**Efficiency**: To develop a system capable of creating the most efficient lab schedules, maximising the use of available resources.

**Flexibility**: To accommodate a wide range of constraints and course requirements, ensuring the system's applicability across various teaching periods and evolving academic needs.

**Usability**: To provide a user-friendly interface that allows staff to input constraints, view schedules, and make adjustments as needed.

**Scalability**: To ensure the system can handle an increasing number of courses, labs, and constraints without a decrease in performance.

**Maintainability:** To provide a solution which can be modified, extended on and further improved by developers external to the development team.

## Basis

The following assumptions will be made about the problem, and form the scope of what is expected of the solution (the system):

1. The system is only concerned with the scheduling of labs for COMP courses at ANU, in the following rooms: HN1.23, HN1.24, N109, N111, N112, N113, N114
2. Lecture and drop in sessions are predefined, therefore they are not required to be scheduled by the solution.
3. The lab schedule produced, should, as much as possible:
    a. schedule labs to rooms efficiency, such that fewer rooms are used
    b. avoid clustering classes during similar times to provide a variety of times.
    c. avoid clashing with labs which may cause significant scheduling difficulty for students (e.g. avoiding scheduling clashes with courses commonly taken together, avoiding scheduling clashes on labs which run very few activities).
4. *Basis 3.c* is only required to be considered if the clash includes a lab that has 3 labs or fewer.
5. The system is not expected to find a solution that meets all conditions outlined in *Basis 3* for all courses and may prioritise certain courses or conditions.
6. The system will be provided with, for each course:
    a. the expected number of student enrolments
    b. the expected number of tutors
    c. the length of the lab
    d. the times and lengths of their lectures
    e. other relevant course requests (e.g. projectors available in computer labs)
7. The system will be provided with, for each room:
    a. the maximum number of students
    b. whether or not that room has projectors and/or recording systems
    c. other special features of that room that are relevant for lab scheduling
8. The solution will provide a schedule on the assumption of an ‘uninterrupted’ week and will not be required to consider public holidays, intermittent room unavailability.
9. Labs are assumed to run in every week and the generated will not be required to schedule labs that do not run during certain weeks.
10. The system will treat dual-coded courses as a single COMP course containing the combined number of enrolments (e.g. ‘COMP2300’ will cover all of COMP2300, ENGN2219 and COMP6719); most postgraduate courses are merged with the dual-code assumption, as such, are not required to be considered.

11. Lab start times must be on the hour or half hour (e.g. 8am, 8:30am, etc.) and have a length that is a multiple of half hours, and between 1 and 3 hours long.
12. The system is only expected to be able to schedule at most 30 COMP courses, and at most 200 labs in total across those courses

The system also has a number of constraints that will be worked within:

1. User interfaces must consider and be designed to be usable by users who may be less technologically proficient.
2. The algorithm should create a timetable within a reasonable amount of time (less than 1 hour).
3. Labs shall be scheduled with respect to ANU policies, i.e:
    a. scheduled within the hours of 8am-8pm
    b. tutor to student ratios in each lab are at most the following, plus or minus 5:
        i. 1:15 for 1000 level courses
        ii. 1:23 for 2000 level courses
        iii. 1:26 for 3000 and 4000 level courses
        (the tutor to student ratio may be adjustable within the system)
    c. tutors should not be required to teach several labs in a row.
4. Labs shall not clash with their respective lecture(s).

## Success Criteria

With respect to the listed constraints, restrictions and assumptions, the solution shall aim to meet the following criteria:

1. The solution shall provide users with an intuitive medium through which to upload and/or enter data. This can include either directly providing data into an online form or uploading data from a file (e.g. CSV).
2. The algorithm component of the solution shall generate lab schedules with respect to the constraints and requirements outlined in the *Basis.* These schedules should be computed within a reasonable time for data sets representative of real-world data.
3. The user interface shall provide users with the ability to make changes to the generated timetable, such as adjusting lab times and lab locations.
4. The implemented solution shall not be restrictive to the extent that key constrains/restriction cannot be modified in the future. It should provide the flexibility for administrative staff or external developers to make changes.
5. The project is documented to a high standard to allow, without difficulty, for modifications, improvements and adjustments after implementation and beyond the development process.

## Project Status

The project consists of a web-interface served by ReactJS, a Java program powered by Spring Boot and a database hosted on Firebase. The synergy of these three components creates the Lab Allocator solution.

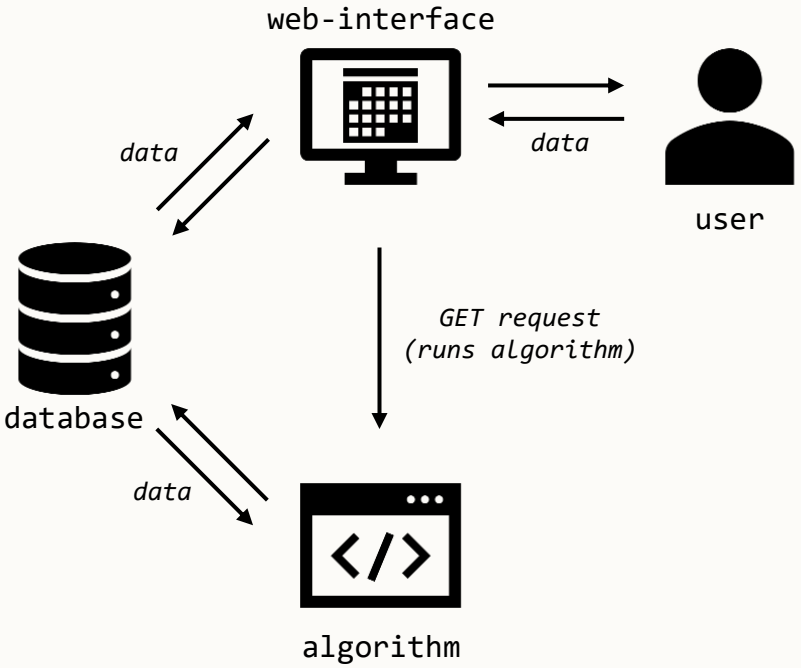



*Figure 1 - Solution components as a visualisation*

**Web-interface and user experience**

Stakeholders require a medium through which to upload and manage data, and the ability to view and manage the generated timetable.
The current state of the frontend is configured to work with firebase, which hosts the database users can interact with.

The current interface allows for data to be uploaded to firebase. Uploaded files are validated with basic logic and entered the database. Once uploading the data has been completed, the algorithm is called to process the uploaded data.

Data can be viewed and managed directly through the interface. This allows data to either be entered or modified should changes be required to the data.

The generated timetable can be viewed in a calendar view, where resulting tutorials can be filtered and have their times adjusted as required.

The web interface facilitates the foundational interactions and processes required for the final product.

**Backend program**

Once requested, the algorithm processes course information to generate a timetable. Currently, the backend retrieves course data from Firebase and uploads the resulting timetable once processing is complete. To activate the timetable generation process, the users web browser sends a HTTP request, the algorithm is executed once the request is received by the program.

The foundations and general structure of the algorithm and related components of the backend stack are complete.
Currently, the algorithm considers the following to determine a ‘good’ schedule:

- The spread of classes throughout the week
- The spread of classes among rooms
- Having certain percentages of labs at certain times (user-defined)
- Number of duplicates (where fewer is better)
- The number of free rooms at any given time
- Having repeat labs (of the same course) in the same room

**Database**

Our current database is implemented in Firebase. The goal of this development period is to develop  a database and integrate into the application. This new database alongside the web-server and algorithm is expected to be hosted within a single Linux environment. This will require the development team to develop a database that replaces Firebase and implement user authentication as recommended by the SoCo facilities team.

Communications with SoCo facilities is ongoing to determine the implementation requirements of the Linux environment.

## Stakeholders

| STAKEHOLDER | REASON |
|---|---|
| Client | They are commissioning the system so that it can serve their needs in allocating labs to classes |
| SoCo Administrative Staff | They may be using the system after it is complete to enter data about course enrolments, lab sizes, etc. |
| CECC Facilities Staff | As the project is intended to be deployed on CECC provided servers, the facilities staff who operate and manage the servers are impacted by the potential maintenance of an additional Linux environment. |
| Course convenors | Provide course information to the client/administrative staff of the School of Computing. The Information provided directly affects the scheduling outcomes of the project |

The following stakeholders do not directly interact with the solution, however, are affected by the outcome of the project:

| STAKEHOLDER | REASON |
|---|---|
| Students | The project directly impacts students by generating schedules which can make up a significant portion of their timetable. Effective scheduling assists students by minimising clashes between courses which are often taken concurrently and provides rooms that meet the tutor student ratios. |
| Tutors | Effective scheduling of classes ensures that tutor to student ratios is not exceeded, making it easier for tutors to teach |
| Other users of the buildings in scope | Effective scheduling means less room times are taken up by labs, allowing others to use the rooms for other purposes |

In addition to the stakeholders above, the students in the SoCo shadow team (Game Parts III) and the tutors of the teams respective tutorial, Dian Lu and David Flores-Condezo, hold a stake in this project. Their input and comments have a direct impact on the functions and output of the SoCo Lab Allocator team.

## Development Process

Utilising GitHub projects, information regarding the development process will be contained in a central ‘Kanban style’ database. The development team shall prioritise action of keeping the project status information including GitHub issues being tracked in this database, accurate and up to date.

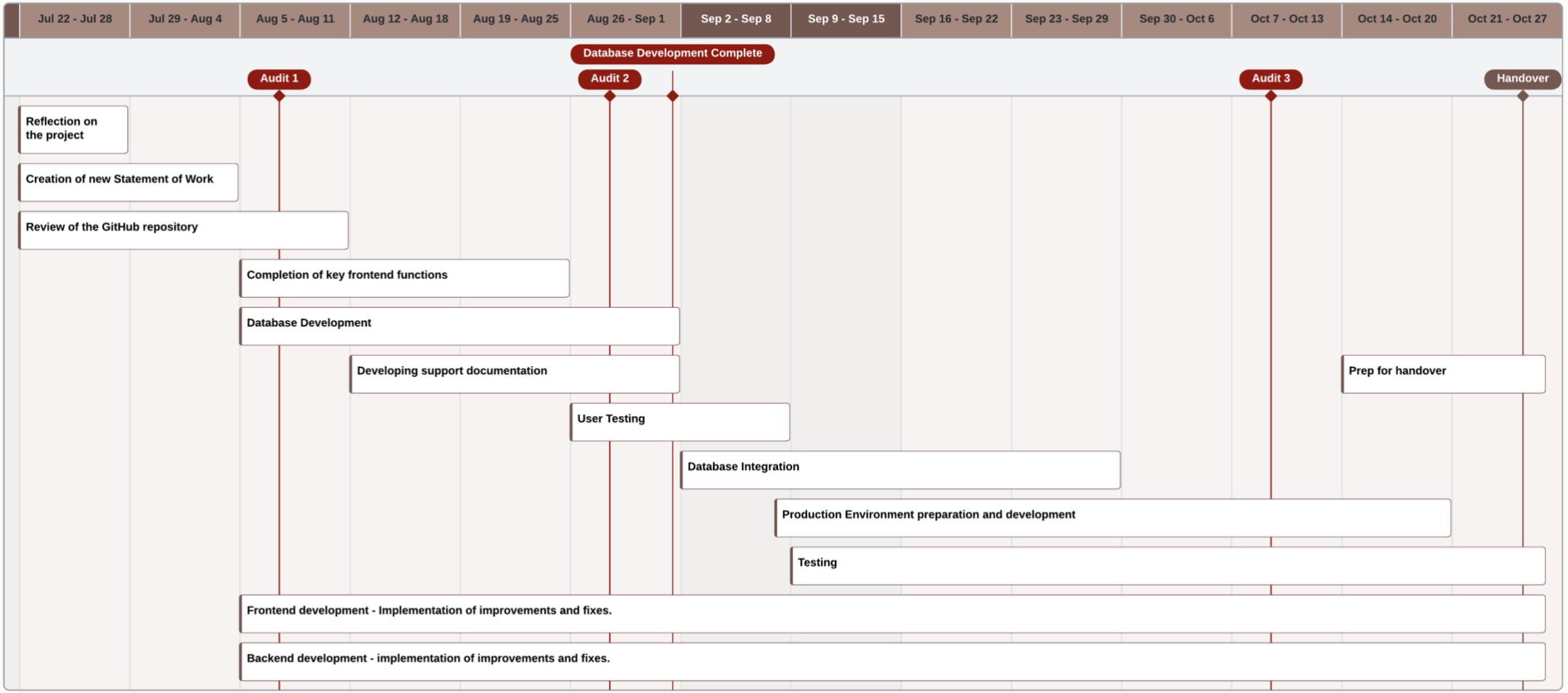
Alongside project status, decisions, reflections and feedback shall be assessed and documented in their respective repository files (or documents not contained in the repository).

Documentation for each aspect of the project code is expected to be completed in a timely manner to avoid confusion and to allow efficient development. Accurate documentation will allow team members assigned to tasks to install, run and begin development with minimal friction. Additionally, common programming styles, as defined by respective language conventions shall be used.

## Goals and Schedule

The following goals are in place for the development team:

- **Transitioning to an ANU hosted database**
  This will require modification to the frontend and backend functions which manage, fetch and publish data. This will be a major milestone that requires a sound understanding of the database and its integration with the front and backend.
- **Finalising Interfaces and functions**
  The data management page is expected to be completed allowing data to be uploaded and modified with the new database.
- **Modifications and tweaks of the algorithm fitness function**
  This is key in generating timetables which meet the requirements and produce sound results that impact stakeholders.
- **Receiving and consideration of user preference**s
  The backend shall aim to include the ability to accept preferences for timetabling from the user and incorporate these preferences into the algorithms function.
- **Further improvements to scheduling**
  Including more efficient scheduling of resources such as projectors and/or recording equipment in computer labs. The algorithm will also attempt to find suitable solutions should a tutorial room become unavailable for a given time.


Jul 22 - Jul 28
Jul 29 - Aug 4
Aug 5 - Aug 11
Aug 12 - Aug 18
Aug 19 - Aug 25
Aug 26 - Sep 1
Sep 2 - Sep 8
Sep 9 - Sep 15
Sep 16 - Sep 22
Sep 23 - Sep 29
Sep 30 - Oct 6
Oct 7 - Oct 13
Oct 14 - Oct 20
Oct 21 - Oct 27
Database Development Complete
Audit 1
Audit 2
Audit 3
Handover
Reflection on the project
Creation of new Statement of Work
Review of the GitHub repository
Completion of key frontend functions
Database Development
Developing support documentation
Prep for handover
User Testing
Database Integration
Production Environment preparation and development
Testing
Frontend development - Implementation of improvements and fixes.
Backend development - implementation of improvements and fixes.

## Resources and Costs

The solution is expected to be deployed on a Linux environment hosted on an ANU SoCo server dedicated to student projects.

Server resources for this project are not expected to require significant computing resources. The virtual environment provided by the SoCo will host the web server, database and algorithm. The specifications of the Linux environment are not expected to exceed two CPU cores, two gigabytes of server memory and 16 gigabytes of storage.

## Contingency Plan

The development team have scheduled weekly meetings to ensure the project is progressing as expected. These meetings provide the opportunity to assess and redistribute workload to ensure deliverables are on schedule and milestones are being met.

In the case that a deliverable is showing itself unable to reach completion by the given time window, the following procedure is in place:

1. Deliverable components are flagged as not feasible to complete by expected deadline.
2. Deliverable supersedes other scheduled deliverables if (and only if) the priority of the deliverable approaching the deadline does not cause significant disruption and/or the superseded deliverable is not imperative to the development and progression of the project.
3. Client and/or relevant stakeholder(s) advised and engaged to discuss potential solutions, alternatives, expected timeline and impact on project.
4. Development team to implement solution and/or alternative implementation identified in step 3.
5. Once deliverable is complete, process complete.

Should it not be feasible to implement the deliverable within the expected timeframe thorough consultation with client will take place regarding the progression of project.

## IP and License

The IP of the solution will remain the property of the students in the group, so long as the School of Computing is able to continue to use and modify the software freely.

## Acceptance