



UNSW
A U S T R A L I A

SOFTWARE ENGINEERING WORKSHOP 2A (SENG2011)

INITIAL PROJECT PLAN



Authors:

Daniel BU	z3424724
Paul JUKIC	z3424666
Robert KWAN	z3467737
Alex JAROSZEWICZ	z3461930
Sanjay NARAYANA	z3424639
Joshua MORTON	z3413752
Ben MO	z5017073
Fufu HU	z3458699

Table of Contents

[Project goals and criteria](#)

[Task Breakdown](#)

[Acceptance Criteria](#)

[Task Relationships](#)

[Work Estimate](#)

[Tasks](#)

[Scheduling](#)

[Roles and Responsibilities](#)

[Project Management Approach](#)

[Project Control](#)

[Communications](#)

[Formal Communication](#)

[Informal Communication](#)

[Information](#)

[Quality](#)

[Issues](#)

[Staff Performance](#)

[Risk](#)

[Risk Response](#)

[Appendix](#)

Project goals and criteria

The aim of this project is to create a flight booking system for the travel agency WeBookU. This system takes input in the form of either a text file containing flight data, or user input representing user queries into the flight booking system. When a user sends a query into our system, we want to display a list of possible flight plans that the user can take for them to reach their destination. The flight booking system must implement both a command-line interface, and a graphical user interface.

The flight plans must be listed in an order dictated by the user. The preference options include ordering via cost (ascending order), via flight time (ascending order) and by frequent flier value with a given airline (descending order). These three ordering preferences may also be combined such that it is ordered by one preference, and then if any of the flight plans have an equivalent preferential value, then it is then further ordered by the second preference, and so on. For example, if the user indicates an order of (cost, flight time), then the results should be ordered by cost in ascending order first, and then any flight plans with an equivalent total cost should then be ordered by flight time in ascending order.

The inputted flight data and user queries will follow an agreed upon syntax. Any input which does not conform to this syntax will be reported back to the user for them to correct before entering it into our system again. The agreed upon syntax for our input will be listed in the appendix of this document.

In summary, in order to complete our project we need to achieve the following higher-level goals:

1. Accept files as data input listing various flights
 - System should also verify that the input conforms to our given syntax
 - Any formatting errors made by the user should produce the necessary errors to the user
2. Accept queries from the user and return a list of suitable flight plans to match the user's needs
 - System should also verify that the user queries conform to our given syntax
 - Any formatting errors made by the user should produce the necessary errors to the user
 - The results should be returned in the order requested by the user

- Users can change their query parameters indefinitely until they have found a suitable travelling plan for themselves
3. Create a command-line interface for our flight booking system
 4. Create a graphical-user interface for our flight booking system

The order for which these goals will be achieved will be explained further below.

Project schedule

Task Breakdown

1. Coding

- Design the Program; the interactions between java files needed using UML Diagrams.
- Accept and store flight data
- Accept and store Queries
- Order Flight Data accordingly by Queries
- Output results from Queries

2. Comments specifications (proofs, assertions, post, pre, invariants)

- Create Precondition and Postcondition specifications for routines
- Find loop invariants and reason

3. Verify back-end Algorithms using Dafny

4. GUI

- Design the interface
- Implement the design
- Make the design functional
-

5. Project Management Report

- Initial Project Plan
- Project Operation Report
- Individual Member Statements

Acceptance Criteria

For the assignment to be coded satisfactorily, the program must always produce correct output from the valid input.

For the individual sections:

- The UML diagrams will need to be valid and correct. They must completely cover the entire project and correctly connect the entire project.
- The data taken as input must be accepted by the program and the program must correctly deal with the data if it is incorrect. This data must then be stored correctly after the completion of the program.
- Similarly to the program data, the data taken as a query must be used appropriately, and the correct output must be produced and stored.
- The data will have to be sorted correctly based on the date of the flights available.
- The output of the data from the query must be formatted correctly, and contain the correct results.

The comment specifications can be proved to be correct if they follow the correct guidelines, and form a logical proof.

- Each routine will need a *pre* and *post* condition to ensure that the program is correct in each stage of output. These pre and postconditions can be checked with Dafny, or by checking manually and verifying with multiple members of the group.
- Having correct loop invariants will need to be figured out by the team through reasoning. It will be satisfactory when the loop invariant logically holds throughout the loop and is accepted by Dafny.

To correctly accept the dafny code, the program must be completed and produce the correct output, whilst maintaining a pass from dafny. Furthermore we must not use any assume statements.

Logically, the GUI will only be marked satisfactorily when all members of the group deem it is, after taking in consideration from external sources. But a GUI is a personal preference from person to person, thus this may be difficult. We will take the GUI as completed when all members of the group are satisfied, and a majority of external sources agree that it makes logical sense and is appealing.

The project management report will be deemed satisfied only at the end of Week 13, as the plan will be refined multiple times throughout the weeks, as parts are changed due to interruptions.

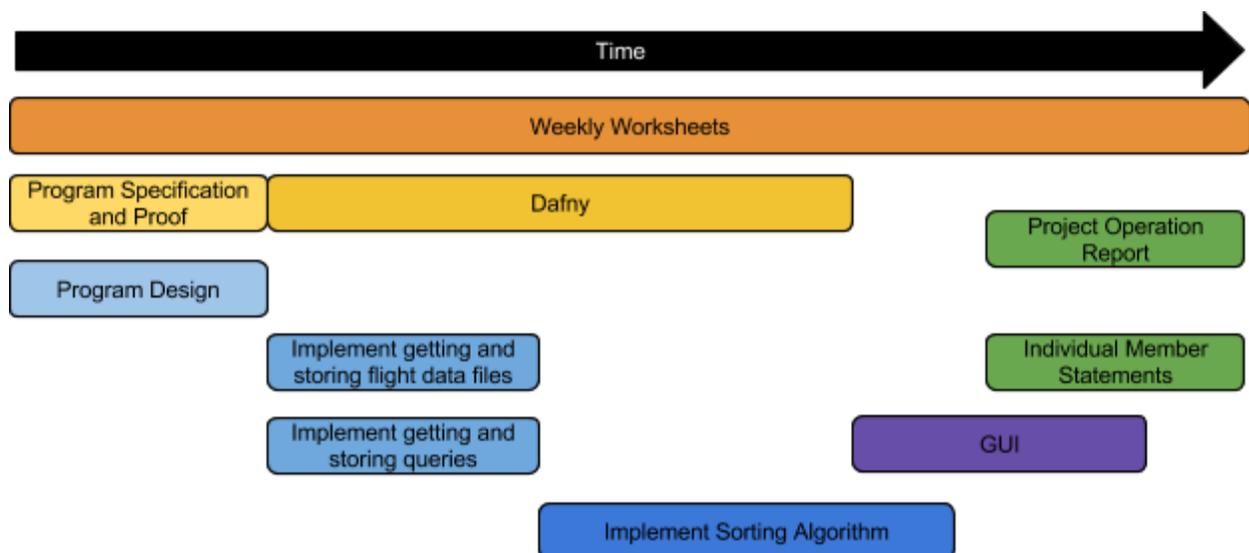
- The initial plan will be completed when it is decided by the group that it successfully completes all the requirements that are required.
- The operation report will be completed by the end of week 13 if it is determined by all group members that it too successfully completes all the requirements asked for in the spec.
- Individual member statements can only be determined to be satisfied by the individual, as this information should not be shared between other members. This satisfaction will be determined if the statement completely satisfies all requirements asked for again from the spec.

Task Relationships

The tasks and the relationship between the tasks are shown below, this includes areas such as precedence constraints, where the program design and program specification will have to be done prior to any code being written.

Our team will then work on the proof of routines and implementing the specification of the program while working on the design using CRC cards and then UML diagrams. At the same time we will write algorithms for getting and storing flight data, queries, and the sorting algorithm which will be verified using Dafny.

Near the end of the completion of the main sorting algorithm, the GUI can be created as long as the algorithm returns output in the correct format, and doesn't have to be correct yet. Finally, after the completion of the sorting algorithm, the whole team will move onto completing the Project Management Report.



Work Estimate

The following table shows different estimates for each task in the project. The estimates are “man-hour” estimates, meaning the total uninterrupted time spent working. The final estimate column is the average of the optimistic and conservative estimates with a 20% safety factor added on, making it lean more towards the conservative side.

Tasks

Task	Optimistic Estimate (hours)	Conservative Estimate (hours)	Final Estimate (hours)
Program specification and proof	3	8	6.6
Program Design (UML Diagram)	2	4	3.6
Dafny	6	14	12
Implementing retrieval/storage of flight data files	2	6	4.8
Implementing retrieval/storage of queries	2	6	5.6
Implementing comparators for sorting results	4	8	7.2
GUI	6	12	10.8
Project Operation Report	4	8	7.2
Individual Member Statements	0.5	1	0.9
Total:	29.5	67	58.7

From the table above, the total estimated hours for the project per person is **7.3** hours each ($58.7 \div 8$). We are aiming to finish the project by the end of week 12, so the average workload per person per week is approximately **2** hours each.

Scheduling

Because we will be undertaking this project whilst accommodating a busy university schedule, delays will be inevitable. Issues that arise will only add to this problem. As such, the use of a Gantt chart or other similar scheduling tools is impractical - it is unlikely that tasks will be started and completed by the dates determined at the start of our project. We will employ a versatile, flexible schedule that is determined every week and reflected in the weekly worksheets. However, this may lead to compression of tasks near the deadline, increasing possibility of error (and thus time required to solve them). To prevent this from happening, we will review project progress through both informal and formal communication methods (meetings).

One thing that is fairly concrete in the project schedule is the overall order of the overarching tasks specified in the Task Relationships section. Although we are not employing an agile development process, our project scheduling is quite similar in that we try to fulfill an optimistic amount of tasks per week, pushing back those that do not get completed whilst constantly reviewing project management to ensure that we can achieve the specifications by the deadline, at standards determined by the quality control process.

Roles and Responsibilities

We strongly believe that tasks would be completed efficiently if individuals are inclined to perform tasks allocated.

We gathered each group member's preferences on project task and then assigned responsibilities according to their preferences. The following chart shows these assignments.

Task	Member(s) in charge
Program specification and proof	Alex Jaroszewicz
Program Design (UML Diagram)	Daniel Bu
Dafny	Fufu Hu, Daniel Bu*
Implementing retrieval/storage of flight data files	Robert Kwan
Implementing retrieval/storage of queries	Paul Jukic
Implementing comparators for sorting results	Ben Mo
GUI	Joshua Morton, Sanjay Narayana
Project Operation Report	Everyone**

* Daniel Bu's task is relatively easy and he is willing to help Fufu Hu out after finishing his own task. ** We have decided to do Project Operation Report together after finishing everything else.

For better cooperation, we have divided our team into 4 different groups.

GUI Group (**Joshua Morton, Sanjay Narayana**): They will be working on getting the GUI setup and linked to our Java main project source code during this project.

Implementation Group (**Robert Kwan, Paul Jukic, Ben Mo**): In this group, everyone's task is fairly similar. They can organise meetings with the UML group to update their current working status.

UML group (**Daniel Bu**): This group is the glue to help the GUI group and implementation group to understand each other's strategies.

Specification and Verification Group (**Fufu Hu, Alex Jaroszewicz**): This group is working on tasks which are strongly related to concepts taught in COMP2111.

Project Management Approach

Project Control

Project control ultimately depends on constant monitoring of current progress and quality whilst actively providing appropriate solutions to issues that may arise. A number of project management tools will be used in order to keep track of our progress and performance

Trello

Trello is a project management tool which we will use to keep track of upcoming tasks, due dates and progress. Tasks will be broken down into smaller items so that they can be checked off as they are completed, allowing the team to monitor the status of the project close to real-time. Furthermore, inside Trello we will have weekly task cards that will make it clear what we have to do each week. It is incredibly easy to make changes to tasks

Weekly Worksheets

These worksheets complement strongly with Trello to make project control easy. It is manually created every week but will give an overview of tasks to be completed each week (transferred to Trello), tasks currently in progress and finished tasks. Furthermore, it allows us to see who is undertaking which task ahead of time.

GitHub

GitHub is used as a repository for the team to store code. This also allows us to identify who made certain changes or additions and thus simplifies tracking down any performance issues. It also enables us to track issues/bugs which might affect the flow of the project.

Should anyone discover the need to make adjustments to progress or performance, we will first discuss with the team (with communication techniques listed below) to reach a solution that everyone finds satisfactory. Then we will document the change and update our project control tools with our solution so that it remains consistent and useful. However, we would still prefer to minimise our changes because it impacts planning.

Communications

The software development team will make use of both verbal face-to-face communication as well as electronic communication methods. In general, in-person meetings will be reserved for formal meetings, whereas electronic communication will be used for throwaway or day-to-day activities.

Formal Communication

Formal communication methods for this project will take the form of bi-weekly 'stand up' meetings at the beginning of the work day. This will involve each project member giving a brief overview of what they have accomplished since the previous meeting, as well as their intended goals for the day. This will give everyone a clear idea of what tasks everyone is working on for the day, and team members will be able to coordinate their tasks with other members easily. These 'stand up' meetings will also create a sense of accountability within the project team. Any major accomplishments or goals reached should be noted by the project leader.

Informal Communication

At certain phases of development, two or more departments may need to meet. These cross-department meetings will be organised by the project leader at a time which is convenient for all involved parties. The minutes from these meetings will be maintained by the project leader, who will then upload these minutes to the project online directory in order to keep an accessible record of the meeting available.

For less formal day-to-day communications, team members can use instant messaging clients to communicate if the involved team members feel a record of their conversation will be necessary in the future. Formal face-to-face communication may be replaced by online communication at any time the team members cannot find a suitable time to meet together. The important talking points from these meetings will be recorded into the 'Weekly Worksheets'.

Information

Project information such as the Project Management Report, weekly worksheets, project designs (such as UML Diagrams), informal proofs and proofs using Dafny, will be stored across various files and folders in our Google Drive shared folder.

In terms of project due dates, progress, and some issues, these will be done over the group trello board, which has a section for each task and new cards containing new sub tasks to do will be added by each sub team's group members as they arise, and crossed off as they are done.

For our source code files, it will all be hosted on GitHub, with everything in the master branch simply for sake of simplicity. File/Class name conventions will be decided upon during Project Design, and any new class that is needed will be added to the folder as needed, new unexpected class names should be based upon common sense. Anyone who is assigned to work on certain files can add their own helper classes as they need, usually changes made upon a central file are discussed among the affected group members before any changes are made (the affected group members are those that are currently working on the central file and those who depend on it).

We will also be making use of git in eclipse in order to keep the features of eclipse while able to still push, pull and merge files easily.

Repo Manager: Daniel

Quality

Quality will be controlled through regular code reviews, test specs, unit testing and functional testing. In particular, it is important that code reviews are done by 1-2 different developers from the developer whose code is being reviewed. Test specs should be written by at least 1 developer who isn't implementing the function being tested. Each developer should write their own unit tests for functions they write using the test spec. Functional testing will ensure that different components of the system work together correctly, and everyone should contribute to writing these.

The benefits of having code reviewed by different people is that we will end up having others familiar with the other developers' code, so in the case of an emergency, there shouldn't be too many problems if someone has to pick up someone else's work.

Issues

Solving issues is the key part of completing any project. Once all issues have been solved, a project is complete. The only difficult thing is that new issues are discovered over time, as it is quite difficult to think and handle for every edge case as you are creating the program.

Issues can be discovered by all team members, if they are testing the program manually, if they are using a test suite to rigorously test the code, or by attempting to break the code by testing with edge cases. All these various ways will allow most issues to be discovered, as well as through our weekly code reviews, most flaws should be picked up as it passes under the scrutiny of multiple people, as well as automatic testing.

For our team, issues will be tracked via github issues. This allows us a convenient location that we all have access too, to see and solve the problems that come up. As we will consistently be viewing github to see what has recently been done by other members, the issues tab provides notifications to the team if something has arisen. Hence by using github, we will be able to quickly communicate to the whole team if something has occurred and needs fixing.

For handling the problem, we will use a tagging system. By tagging the person whose code is responsible for handling the issue, it is clear to everyone who needs to do what work. This tagging system will also allow others to give information in the bug file about what they believe is the problem, and possible solutions that could be incorporated to fix the bug itself.

Bugs will also be discussed in our weekly meetings, as well as in our communication on social media. Many logical bugs can be solved this way as we discuss between teammates, possible solutions. Discussion between members can also help if someone is stuck on trying to discover a solution for the problem they are having.

Finally, issues if they still have not been resolved can be put in as a current task needed to be accomplished on our Trello board. This will allow all team members to see that it is a current focus, and will need to be completed in the current week's work cycle.

Staff Performance

The staff's performance will be maximised through a variety of strategies, which include first building team relationships, through having team meetings and getting to know each other. At the same time, after understanding each others' goals, strengths, weaknesses, and finding common ground, we will be able to allocate tasks accordingly. This will allow our team members to improve their own performance as they will be placed in the roles in which they have a clear understanding of the objectives and what is expected from that task by the whole team.

Another method of improving our team's performance will be based on higher collaboration between different roles as this will help teams aim towards the same goal, and allow each sub team to see how their area of work affects other areas of the project.

Furthermore, each objective and its detailed tasks will be written in a clear manner, in order to allow the team to correctly follow it and thus achieve a correct result effectively.

Also a few incentives will be used, both positive and negative, in order to incentivize positive performance and reduce negative performance. The most effective method to induce positive performance will be based around peer pressure, and having a chart of all the specific tasks, so that sub teams can cross off sections as they are completed. These allow the team to move forward, both in progress and performance levels, as having your peers encourage you and having others rely on you doesn't give one much room to procrastinate and/or leave the task completely undone. The chart of all tasks allows the state of progress to be visualised, which is a helpful motivating factor for the team. The chart will be placed with all the project files on the google drive shared folder.

Finally, the performance levels of team members must also be continually monitored in order to decide on the strategy in order to raise that performance or to maintain it. If there has been an underperformance, we need to ensure the team member understands the importance of the time/quality constraints placed on the team through our own scheduling, and make sure there aren't any problems in understanding or personal problems with the task or outside of the project. This will also be done through team and/or individual meetings, either in person or through skype.

Risk

Risks are *possible* events, that, without responsive action, could adversely impact our project outcomes. Below is a collection of possible events that could occur, with its implications shown on the right.

Risk	Implications
Missing group member(s)	Low team morale, Low productivity
Emerging internal conflict within group members	Low productivity, low trust between members, productivity is impeded.
Inflammation of requirements (As project increases in size, requirements may unexpectedly increase)	Scheduled tasks fall in risk of being inaccurate,
Incorrect time scheduling (Growing nature of software means it is difficult to estimate time to complete tasks)	Low productivity
Mis-communication and misinterpretation	Incorrect tasks implemented, time wasted, resources wasted
Loss of project files	Low team morale, Low productivity, resources wasted
Wrong specification. (Specification supplied is not accurate enough)	Time wasted, resources wasted, poor overall productivity.

Risk Response

Responding to risk will be done in a calm, orderly fashion in the meeting with team members adhering to this risk response table. Members will also discuss their thoughts on the topic with adherence to the **Communications** section outlined above.

Risk	Risk Management Strategy
Missing group member(s)	Accept: Keep group member up to date, in accordance with informal communication protocols established earlier.
Emerging internal conflict within group members	Avoid and Mitigate: Avoid inflating situation and instigating fights. Mitigate by ensuring friendly, constructive communication protocols between each team members such that all issues are resolved democratically.
Inflammation of requirements	Accept: Manage time, resources effectively and discuss requirements between team members in case of potential action that must be taken.
Incorrect time scheduling	Mitigate: Incorporate accurate time-estimations into scheduling tasks.
Mis-communication and misinterpretation	Avoid: Ensure all team members are up-to-date with information and are all on the same page through meetings, asking questions and writing up minutes for meetings that team members have missed.
Loss of project files	Avoid: Ensure all code is backed up correctly, ensure all have equal, unimpeded access to all required project files at any given instance
Wrong specification.	Mitigate: discuss important payoffs that must be made in order to assure that productivity can continue

Appendix

The system is to read the database containing flight data from a file, that is formatted according to the following context-free grammar, with start symbol S :

$$S \rightarrow \epsilon$$
$$S \rightarrow \textit{Flight } S$$
$$\textit{Flight} \rightarrow [\textit{Date}, \textit{Time}, \textit{Name}, \textit{Name}, \textit{Duration}, \textit{Name}, \textit{Number}]$$
$$\textit{Date} \rightarrow \textit{Number} / \textit{Number} / \textit{Number} \quad (\text{representing Day/Month/Year})$$
$$\textit{Time} \rightarrow \textit{Number} : \textit{Number} \quad (\text{representing Hours:Minutes, 24 hour clock})$$
$$\textit{Duration} \rightarrow \textit{Number} \quad (\text{representing number of minutes})$$
$$\textit{Number} \rightarrow \textit{Digit} | \textit{Digit } \textit{Number}$$
$$\textit{Digit} \rightarrow 0 | .. | 9$$
$$\textit{Name} \rightarrow \textit{CapitalLetter } \textit{Lowercase}$$
$$\textit{CapitalLetter} \rightarrow A | B | .. | Z$$
$$\textit{Lowercase} \rightarrow \epsilon | \textit{LowercaseLetter } \textit{LowerCase}$$
$$\textit{LowercaseLetter} \rightarrow a | b | .. | z$$

Here `query-filename` contains a list of queries, formatted according to the following grammar, with start symbol $\textit{QueryList}$

$$\textit{QueryList} \rightarrow \epsilon$$
$$\textit{QueryList} \rightarrow \textit{Query } \textit{QueryList}$$
$$\textit{Query} \rightarrow [\textit{Date} , \textit{Time}, \textit{Name} , \textit{Name} , \textit{PreferenceOrder} , \textit{Number}]$$
$$\textit{PreferenceOrder} \rightarrow (\textit{PreferenceType} , \textit{PreferenceType} , \textit{PreferenceType})$$
$$\textit{PreferenceType} \rightarrow \textit{Cost} | \textit{Time} | \textit{Name}$$

The output produced in response to a list of queries should be formatted according to the following grammar, with start symbol *QueryAnswerList*

$$QueryAnswerList \rightarrow \epsilon$$
$$QueryAnswerList \rightarrow (Query , [Answer]) QueryAnswerList$$
$$Answer \rightarrow \epsilon$$
$$Answer \rightarrow ((FlightPlan), Number, Number, Number) Answer$$
$$FlightPlan \rightarrow \epsilon$$
$$FlightPlan \rightarrow Flight FlightPlan$$

(reusing nonterminals from the grammars above.) That is, the output is a list of queries and the associated answers, where each answer is a list of entries of the form

$$((FlightPlan), Number, Number, Number)$$