Project Plan

Project Title: River Crossing Game

(with customizable puzzle and solver)

Table of Contents

1. Abstract

This report aims to describe the project management plan for implementing the river crossing puzzle. It will indicate background information, such as the project description, stakeholders, scope, and objective.

In addition, this report will discuss the constraints, software development methodology, and project team organization which aims to minimize the development risk. Last, the information includes the project work breakdown structure with the Gantt Chart for the development team to follow.

2. Project Description

River crossing puzzle is a famous puzzle game. The goal of the game is typically to move all the items/characters from one land to another land. Meanwhile, the items/characters may have special relationship with each other or have some restrictions on moving, bringing challenge and difficulty to the puzzle, such as “sheep cannot stay with tiger if farmer is not around”, “criminal will hurt others if police is not around”, “only farmer can move ship”. With the increased restrictions and character, the puzzle's difficulty will rise. There are many computations required to solve the puzzle optimally.

This project, River Crossing Game (with customizable puzzle and solver), aims to provide a playable river crossing game, and a puzzle solver to find optimal solution of the given puzzle. At the same time, allowing user to customize their own puzzle with simple text editor.

2.1. Stakeholders

Stakeholders	Description
User	User can input step in the program to process the game.

	User can also input rule and character to solver to compute solution.
Project team	Develop the complete river crossing simulator. Design suitable data structure and algothium for the solver to compute optimal solution

3. Project Constraints

3.1. Time constraint

This project must be delivered within the deadline (the first week of December). The time constraint is not negotiable or exceeded. Thus, tight control in the progress and rescheduling must be applied to every development phase.

3.2. Cost constraint

The major cost of the project is manpower. The project team consists of five members, and not all members are familiar with coding. Even Though human resources are limited, the project's aims can be fulfilled with careful control and adjustment to the other constraints.

3.3. Scope constraint

Since the other constraints, like time and cost, are fixed and cannot be rearranged, the project scope is the only one that can be fine-tuned to obtain a balance point. Therefore, the project scope must be designed to fit the other two constraints (e.g. time and cost). As a result, the idea of a command line cross-river puzzle game was raised.

4. Objective

This project aims to provide River Crossing Game for users to play, and automatic puzzle solver to find the optimal solution of the puzzle. At the same, allowing users to customize their own puzzle to play and solve. Below are the functional requirements that satisfied this goal.

4.1. Solver algorithm (Solver mode)

The algorithm should provide the shortest solution for the given river crossing puzzle, and able to detect whether the puzzle has a solution. The solution should be detailed which show the steps clearly.

4.2. River crossing puzzle simulator (Player mode)

There should be a game simulator for users to taste how the river crossing game works. This river crossing puzzle software will simulate the real river crossing game for the user to play. Users can text command to move the items/characters and receive instant feedback, such as whether a move is invalid.

4.3. Puzzle customization

As the river crossing game can be adapted in multiple rules, the program should allow the user to customize the items/characters as well as the rules. The puzzle information (e.g., characters, rules, initial state) will store in an external file. Players can modify the attributes in that file and create their own puzzle, then import the modified file to play with or obtain the puzzle solution.

5. Development Methodology

5.1. Project Development Approach and Practices

To finish the project on time and run the game without bugs. The Schedule-Oriented Practices and Test-Driven Approach are applied.

5.2. Schedule-Oriented Practices

The project was measured and rescheduled from time to time following the cycle on the left side. We keep Tracking Progress, Measuring Quality and Timing, Re-planning & Rescheduling.

Fig 1 Schedule-Oriented Practices Cycle

5.2.1. Test-Driven Approach

When measuring the project, every task was tested for quality assurance. In case of failure, the task will be rejected for correction and not allowed to go further to the next cycle. The first step to do is to write a failing test case. Then make coding to pass the failing test case. In case necessary, the entire code is refactored. Finally, the whole process is repeated, and more tests are written over time.

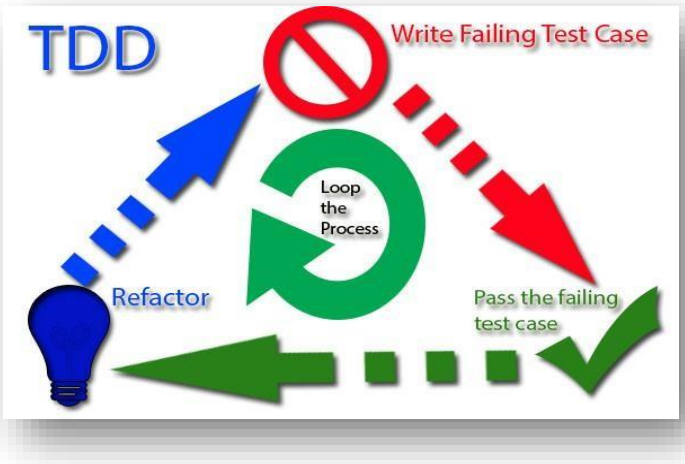

Fig 2 Test-Driven Approach Cycle

5.3. Programming Development Methodologies

5.3.1. Concurrent Engineering Model

The divide and conquers principle is adopted in the programming part. The coding is divided into four submodules: the Game user interface, the Game logic development, the Solver mode development and the Player mode development. Thus, the initial planning involved numerous designs of game flow, classes, parameters, methods, and coding patterns. Each submodule is worked on its functions and follows a spiral approach. Moreover, fixed periodic integration trials are made to evaluate risk in advance. Such a parallel methodology is suitable for a tight schedule like this project. Once all the submodules are done and tested, the final program is ready for integration and implementation.

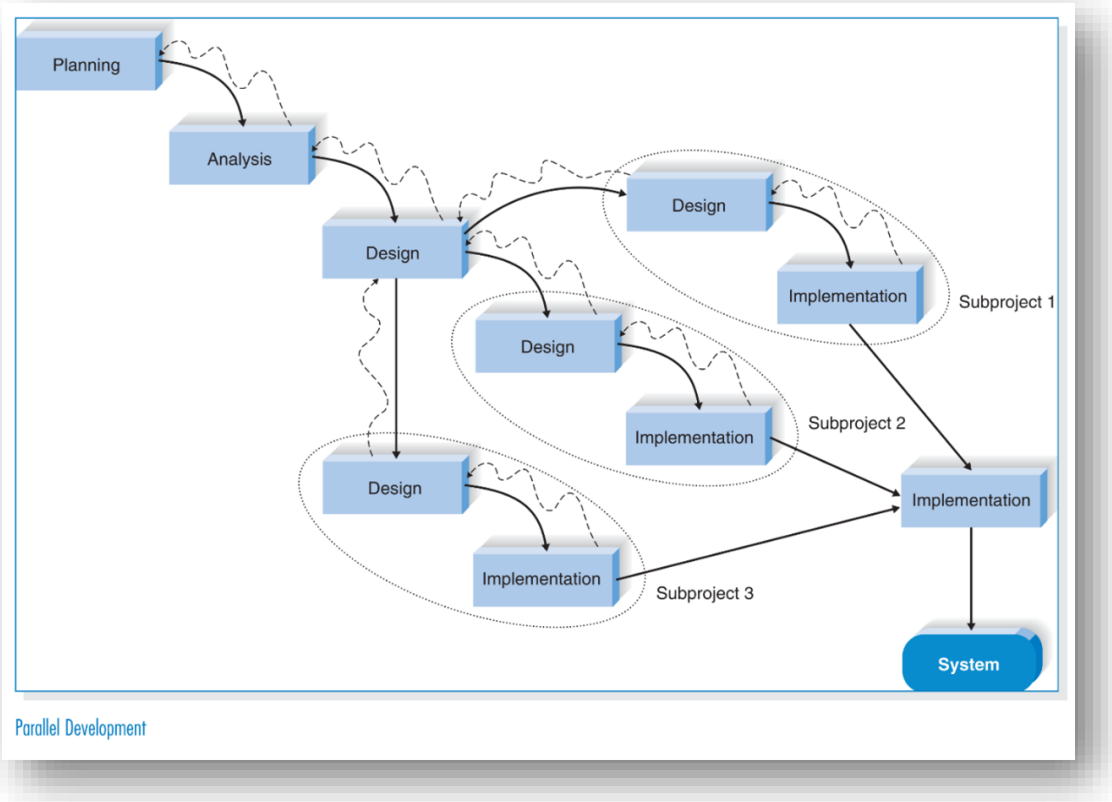

Fig 3 Parallel Development Flow (from 'Systems Analysis Design UML 5th Edition')

5.3.2. Spiral Model

As the coding is divided into several submodules, the Spiral process model is used. Since user interfaces mainly drive the program, Spiral is a quick method to have an initial function to examine the reactions between the puzzle logic and the interfaces.

Thus, the Spiral model is applied to each developing parallel line. Due to the iterative nature of the process model, each loop in the Spiral will increase the submodule complexity and functions. It results in flexible progress in the expanded functions and complexity of the submodule.

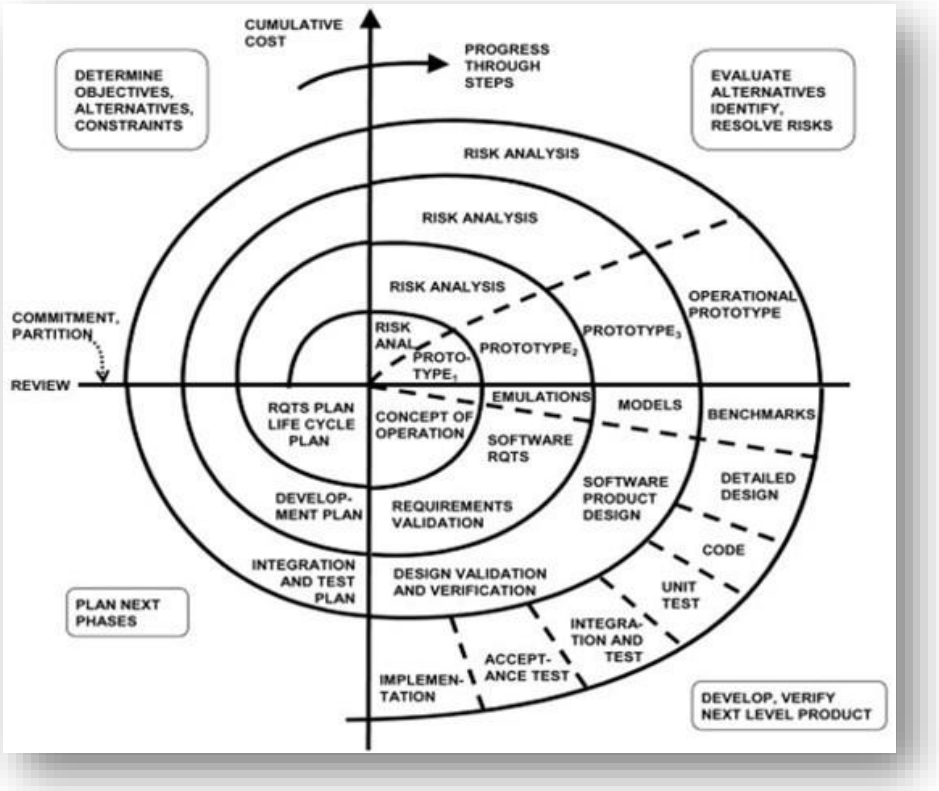

Fig 4 The original spiral model diagram. Credit - Barry Boehm, & Wilfred J. Hansen

6. Project Team Organisation

6.1. Functional Organisation

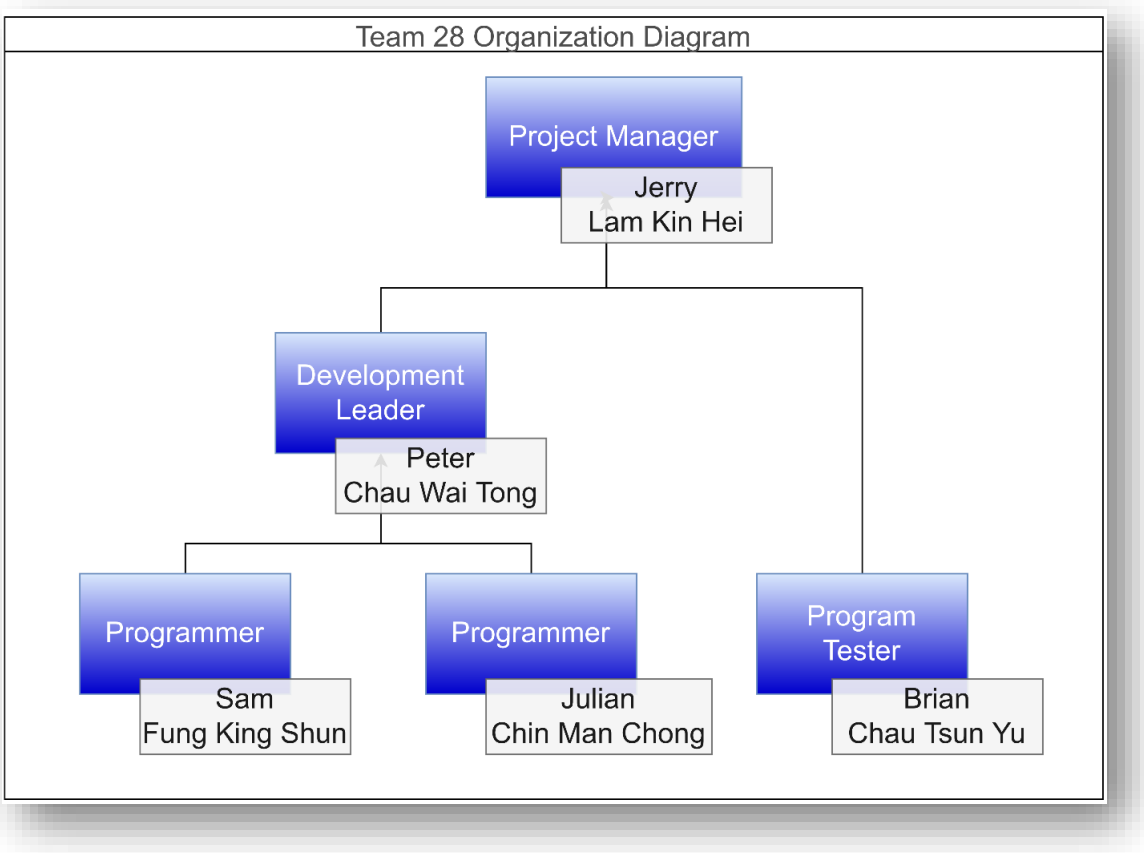

Fig 5 Functional Organization Chart

Even though this is a small team, the roles were designed to fit the functions of the project needed. All the tasks required can be handled by the three function categories listed below. Moreover, every team member was assigned a role according to their characteristics and strengths. An action-oriented character fits into the position of code building.

6.1.1. Scheduling and Documentation

The project manager, Jerry, is responsible for organising the whole project, including planning the project idea and direction, scheduling and documentation, project scope definition, and supporting the team members in case new resources are needed.

6.1.2. Programming and Implementation

As development leader, Peter is responsible for the entire program design, development functions, project scope definition, documentation and supporting the project manager for any arrangement. For programmers, Sam & Julian cooperate with development, leading to writing code and functions.

6.1.3. Testing and Debugging

The program tester, Brian, is responsible for writing unit tests and integration tests to test the program, which can ensure the program runs as expected and functions correctly.

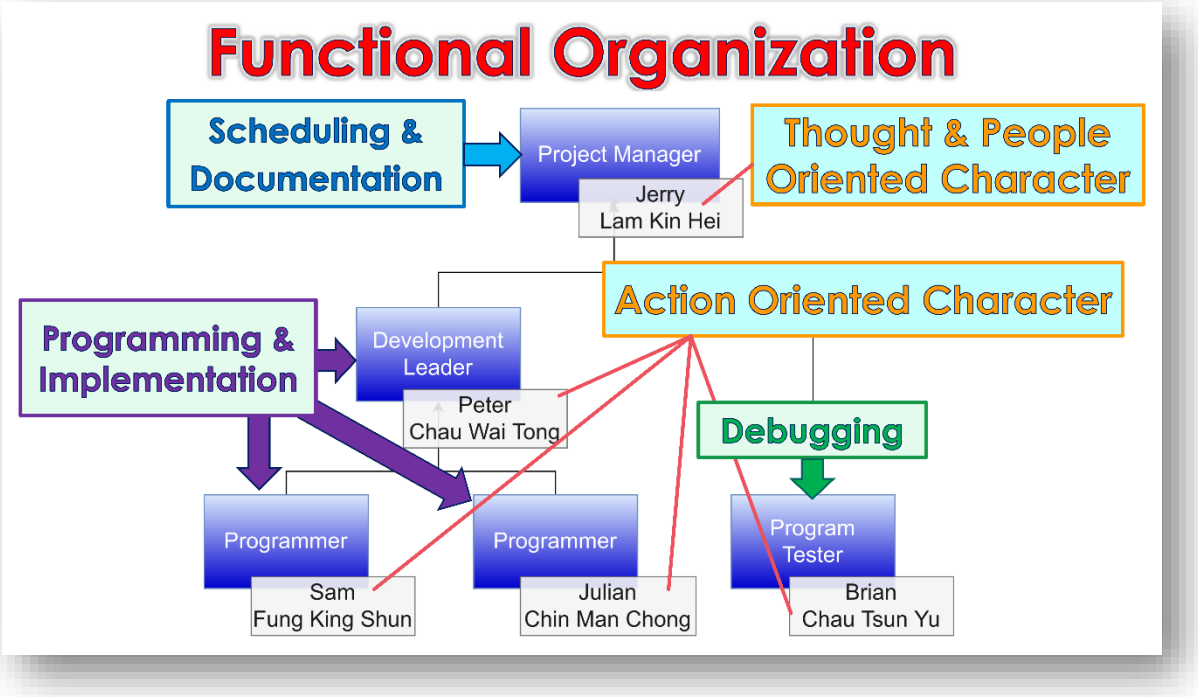

Fig 6 Details Description of the Team Functional Organization

6.2. Work Breakdown (WBS) Structure

6.2.1. Bottom-Up Techniques

Bottom-Up Techniques were used to create the project work breakdown. The team started at the lowest level of the task's details and requirements. Then analogised, the similarity of all those tasks and workforce resources should be allocated. Thus, capable of fine-tuning the project template. The last step was to brainstorm all tasks to be done and group them into a few categories.

6.2.2. Process-Oriented WBS

To closely monitor the project's deliverables and manage the project schedule. The process-typed WBS was selected to let the team focuses quickly on every task's progress. It provided a thorough breakdown of jobs from a functional perspective and gave a more coherent project scope.

The tasks were divided into four categories: Planning and Schedule, Programming & Implementation, Debugging & Test, and Documentation. A detailed list follows below:

1. Planning and Scheduling
    - 1.1. Topic discussion
    - 1.2. Define scope
    - 1.3. Develop project schedule
    - 1.4. Develop project staffing plan
2. Programming & Implementation
    - 2.1. Feasibility study
    - 2.2. Use case diagram design
    - 2.3. Sequence Design
    - 2.4. Design pattern analysis
    - 2.5. Game Logic Development
    - 2.6. River crossing simulator
    - 2.7. Auto solver
    - 2.8. Design pattern analysis
    - 2.9. Code Refactoring
    - 2.10. Packaging
3. Debugging & Testing
    - 3.1. Unit Test
    - 3.2. Integration Test
    - 3.3. Coverage Analysis
    - 3.4. Testing detail
4. Documentation
    - 4.1. Bug report
    - 4.2. Reports of Planning, Analysis, Release & Logs

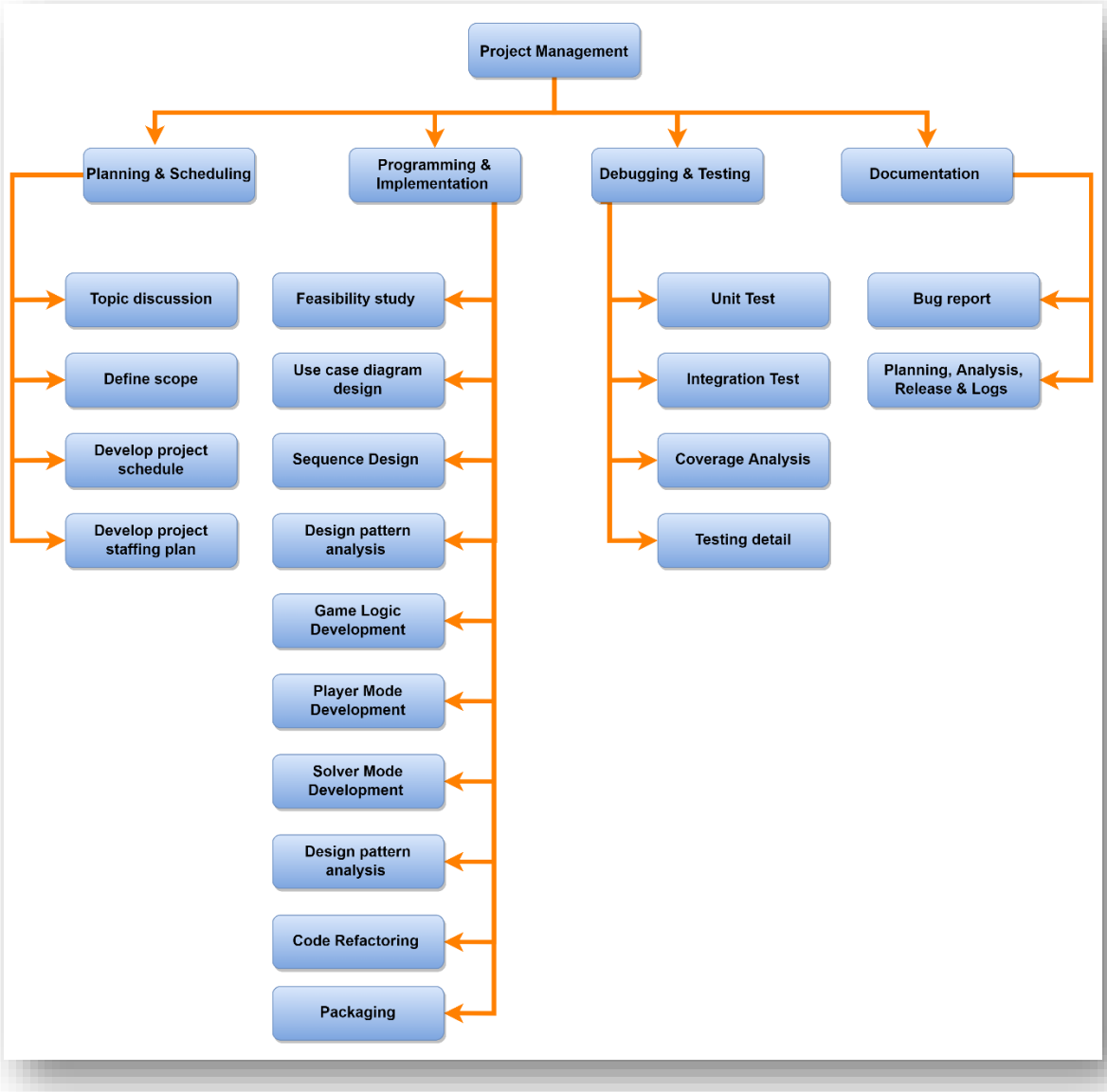

Fig 7 Process type work breakdown structure

7. Project Gantt Chart

From the Gantt Chart here, there are 6 phases arranged by finished time. They are the Initial Project phase, the Analysis and Design phase, the Software Development phase, the Testing phase, the Debugging & Refactoring phase, and the Project Closing. The tasks list below shows all the tasks under the corresponding stage.

River Crossing Puzzle Game Tasks List

Tasks

Name	Begin date	End date
Project management	05/09/2022	25/11/2022
Initial Project	05/09/2022	20/09/2022
Topic discussion	05/09/2022	07/09/2022
Feasibility study	08/09/2022	12/09/2022
Define scope	08/09/2022	12/09/2022
Develop project schedule	13/09/2022	19/09/2022
Analysis and Design	13/09/2022	08/10/2022
Use case diagram design	13/09/2022	19/09/2022
Sequence Design	13/09/2022	19/09/2022
Class Diagram Creation	20/09/2022	23/09/2022
Design pattern analysis	20/09/2022	07/10/2022
Software Development	26/09/2022	15/10/2022
Game Logic Development	26/09/2022	14/10/2022
Player Mode Development	26/09/2022	14/10/2022
Solver Mode Development	26/09/2022	14/10/2022
Game User Interface	26/09/2022	14/10/2022
Testing	03/10/2022	28/10/2022
Unit Test	03/10/2022	21/10/2022
Integration Test	10/10/2022	27/10/2022
Coverage Analysis	17/10/2022	27/10/2022
Testing detail	12/10/2022	21/10/2022
Impementation Test	20/10/2022	27/10/2022
Debugging & Refactoring	30/09/2022	16/11/2022
Bug report	24/10/2022	15/11/2022
Code Refactoring	30/09/2022	31/10/2022
Project Closing	12/09/2022	25/11/2022
Packageing	28/10/2022	24/11/2022
Documentation	12/09/2022	24/11/2022
Planning Report	12/09/2022	07/10/2022
Analysis Report	28/09/2022	28/10/2022
Release & Logs	12/09/2022	24/11/2022

Fig 8 Tasks list

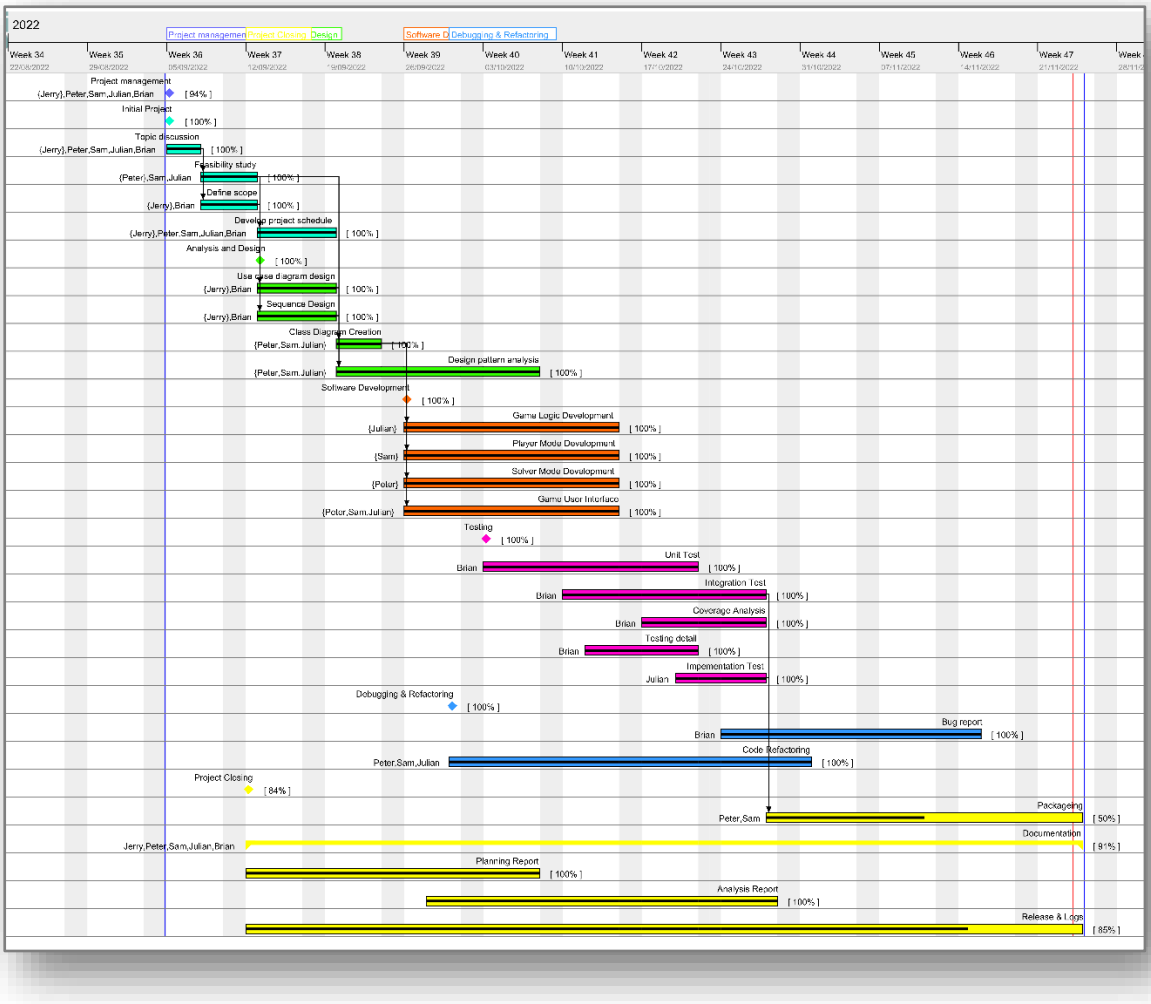

Fig 9 The project Gantt chart

Finally, we worked out the critical path from the Gantt Chart, and let all members take special care of them.

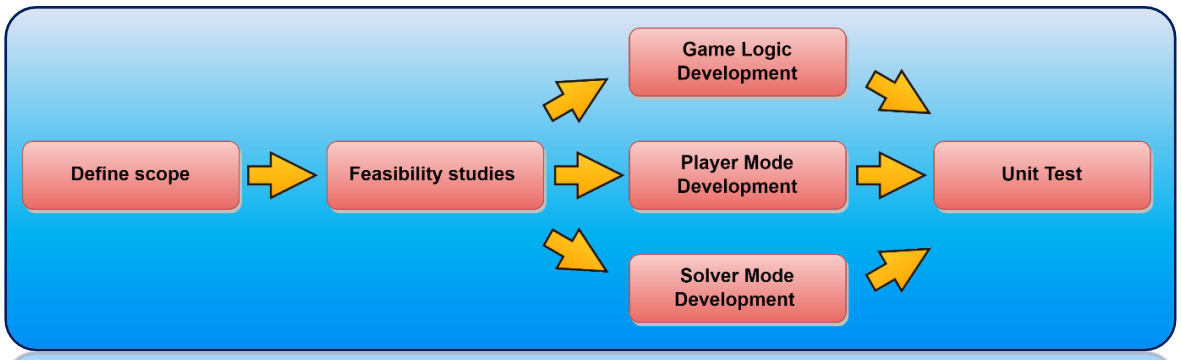

Fig 10 The Critical Paths

8. Project Time Schedule Tracking and Control

For tracking the project's progress and controlling the time schedule, there was a routine meeting every week. The quality and timing of every task will be reviewed, and the reorganization of extra resources will be applied if necessary.

In the case of Solver Mode Development, the team spent two extra working days finishing it due to unexpected situations. Since the Gantt Chart design did not include Saturday and Sunday as working days, it can be caught up with the schedule by working overtime on weekends.

River Crossing Game Project Gantt Tasks Management Table

Begin date	End date	Work Days	Tasks Name	Completed Date	Tacking & Rescheduling		
					Days of Gap	Reschedule	Total Tasks Completed
			Phase – Initial Project				
05/09/2022	07/09/2022	2	Topic discussion	07/09/2022	0	no need	Completed
08/09/2022	12/09/2022	4	Feasibility study	11/09/2022	-1	no need	Completed
08/09/2022	12/09/2022	4	Define scope	17/09/2022	5	done	Completed
13/09/2022	19/09/2022	6	Develop project schedule	19/09/2022	0	no need	Completed
			Phase – Analysis and Design				
13/09/2022	19/09/2022	6	Use case diagram design	19/09/2022	0	no need	Completed
13/09/2022	19/09/2022	6	Sequence Design	19/09/2022	0	no need	Completed
20/09/2022	23/09/2022	3	Class Diagram Creation	26/09/2022	3	done	Completed
20/09/2022	07/10/2022	17	Design pattern analysis	07/10/2022	0	no need	Completed
			Phase – Software Development				
26/09/2022	14/10/2022	18	Game Logic Development	13/10/2022	-1	no need	Completed
26/09/2022	14/10/2022	18	River crossing simulator	14/10/2022	0	no need	Completed
26/09/2022	14/10/2022	18	Auto solver	16/10/2022	2	done	Completed
26/09/2022	14/10/2022	18	Game interface design	10/10/2022	-4	no need	Completed
			Phase – Testing				
03/10/2022	21/10/2022	18	Unit Test	21/10/2022	0	no need	Completed
10/10/2022	27/10/2022	17	Integration Test	27/10/2022	0	no need	Completed
17/10/2022	27/10/2022	10	Coverage Analysis	25/10/2022	-2	no need	Completed
12/10/2022	21/10/2022	9	Testing detail	21/10/2022	0	no need	Completed
20/10/2022	27/10/2022	7	Implementation Test	30/10/2022	3	done	Completed
			Phase – Debugging & Refactoring				
24/10/2022	15/11/2022	22	Bug report	15/11/2022	0	no need	Completed
30/09/2022	31/10/2022	31	Code Refactoring	20/10/2022	-11	no need	Completed
			Phase – Project Closing				
28/10/2022	24/11/2022	27	Packaging			On Schedule	In Progress
26/09/2022	25/11/2022	60	Documentation			On Schedule	In Progress

Fig 11 Tasks Management Table

-end-