Software Engineering Group Project - Final Report

Author(s): G.D Hughes [gwh18@aber.ac.uk]

Config Ref: Final-Report-GP9

Date: 10 May 2023

Version: 0.3

Status: In review

Department of Computer Science

Aberystwyth University

Aberystwyth

Ceredigion

SY23 3DB

CONTENTS

1. INTRODUCTION

1.1 Purpose of this document

The purpose of this document is to provide a report of the progress and types of teamwork done by Group 9.

1.2 Scope

This document should be read by all members of the team so that they are given the chance to agree or disagree with its contents. This document should be read by the markers.

It is assumed the reader is familiar with the following documents:

- SE.QA.RS-CS22120 - Chess Tutor Requirements Specification [1]
- UI-Spec-Docu-GP9 - UI Specification Document and Presentation [2]
- Design-Spec-GP9 - Design Specification [3]
- Test-Spec-GP9 - Test Specification [4]

1.3 Objectives

This document will provide a management summary, a historical account of the project, a summary of the final state of the project, a summary on the performance of each team member, and a critical evaluation of the team and the project.

2. FINAL REPORT

2.1 Management Summary

The project has satisfied all written functional requirements. All the documentation has been reviewed and judged by QA to be of high enough standard to publish. No document is missing any sections or content that would otherwise be vital for its function.

The group has produced a Chess tutor with a functioning backend which has passed all system tests and JUnit tests, and a functioning frontend that provides a clear user interface that communicates the state of the board to the user in an appropriate manner. The Chess tutor allows for player to save and load matches they've played against an opponent on a shared screen, and allows the users to view through the step through the moves of a completed game or an unfinished game currently being played. It contains all the major rules of chess, the standard piece movement logic, pawn promotion, en passant, however a late last minute bug was found that wasn't caught by system tests of JUnit tests to do with castling. The user is able to castle over hostile tiles which should not be allowed.

The group was productive, each member over the course of the project averaging at least eight hours a week of group project work according to blogs. Progress was tracked by the team leader on an excel sheet and with the exception of one or two weeks, the group reliably updated their blogs and progress could be reviewed every meeting with the project manager to inform the assignment of members to actions over the following week. Informal meetings were held every week Wednesday at 3pm to review documentation, reassign members to different tasks if necessary, and walkthrough code implementation and the architectural design.

During these informal meetings, problems would be addressed ahead of that week's meeting with the project manager. The team generally met deadlines on producing the code and documents of the program. However, it became apparent that processes to keep the Unit Testing team up to date with development of the system were necessary. A process of using branches and GitLab issues to streamline the process of making changes to the main code repository and keep an audit log of features being worked on.

2.2 A Historical Account of the Project

The project began the 23rd of January. The group met and assigned tasks to volunteers. Roles such as project leader and QA manager hadn't been agreed upon yet, so the group agreed to table an informal meeting on Wednesday to discuss roles and progress. Sean and Tyler were assigned to the researching the requirements of the UI specification, Jim and Craymon assigned to researching the Test Specification, whilst Gwion and Shaun were assigned to discussing the design and any potential problems the group might face creating the codebase for the Chess Tutour. Jack and Cieran were not present for the first tutorials due to circumstance. Tyler begun researching the specification for the group roles to explain the duties to the group.

It was decided early that the model for the program would be hierarchical. The frontend would talk to the backend and the backend would response. The processes would not occur in parallel as there may be issues with attempting to implement a complex thread

based model in a group without much experience. This discussion included Jack, the frontend lead Gwion, and the backend lead Shaun, where they agreed the backend and frontend would be developed in isolated packages but with agreed means of communicating with eachother.

The most anticipated problem with the code base was how the separate codebases would communicate with one another during integration. It was discussed that the backend would store Forsyth Edwards Notation strings which could be split into sections to read the state of the board and update the frontend. The goal was to develop a means of the simplest and least intensive communication method between the two packages so they could be developed in isolation. This early design discussion in the first week laid the groundworks for the Design Specification and all codebase prototyping that came after.

During the first informal meeting, the majority of tasks had already been accomplished and spikework on the backend begun by Shaun. Roles were unanimously decided during the meeting. Tyler was given QA manager role and chose Sean as his deputy. Gwion was given the leader role and chose Jack as his deputy.

Work began on the UI specification using a template Tyler had provided for the team to follow based on the standards. During these early weeks, it became obvious how important parallel communication was. The UI specification waited on the frontend prototype screens for some time delaying progress. Cieran was moved to the frontend to solve this and Jack assigned to the UI specification to help Tyler. This was to ensure Tyler could focus on how best to perform his role as QA manager.

The UI specification's first version was reviewed and submitted on the 27th of february. No section was incomplete, however it was agreed review sessions should be held earlier.

Every tutorial started with a review of a spreadsheet maintained by the leader of that week's hours committed by each member as a whole and on individual tasks. This informed future decisions on task assignment and helped maintain productivity over the project's lifetime.

After the UI specification was submitted, Gwion and Shaun were assigned to begin work on the design document. Jack remained to work on the UI Specification feedback whilst Craymon and Sean began researching JUnit. The Test Specification was reviewed and submitted on the 6th of March with feedback from the group.

One issue realised was that the Test Specification would need to be very explicit with the boardstate being tested, and how best it was to write that board state in a clear way for system tests during integration and testing week. It was decided as a team, the base way to handle this was to use FEN board notation to solve this issue. The remaining members of the test team moved to work primarily on the JUnit package, although the Test Specification continued to be maintained and added to as the codebase took shape. The test specification would also need to be continually maintained and added to as the codebase took shape.

The Design Specification was a document so large in scope it became the work of four team members. The authors were divided into their areas of responsibility, Jack for the log handler, Shaun for the logic package, and Gwion for the frontend package. One of the greatest challenges in writing an accurate and cohesive document was that if the author did not understand the system enough to explain it, the JUnit team would encounter numerous issues as well. A code walkthrough of the frontend and backend packages was run on the 22nd of April ahead of Design specification submission to help demystify the codebase to the team. The Design specification was reviewed by the team at a Wednesday meeting, then submitted after feedback on the 20th of March.

The architectural design of the program was agreed upon, however the software engineering process of the group was discovered to be problematic. A major issue discovered was how the codebase had been developed in isolation from the JUnit package for two weeks without the team noticing.

As a result, new procedures were agreed upon. The team had already begun using Gitlab issues as a tool to track necessary documentation changes, it could be used to mark files for active development and work on the codebase would be done all in one folder. Tyler provided a Maven pom file that would ensure the codebase being worked on by everyone had the same version of java with the same dependencies. There was a general consensus to use branches rather than copying work to a new folder to work on new features. These processes introduced a rigid filestructure to dev to support branches, and reassured the JUnit package authors they would be working from the most current accepted code.

Ahead of Easter, the team leader created a timeline of goals to achieve by integration and testing week. Whilst the frontend lagged behind the backend codebase, it was agreed that integrating the current codebase was more important than working on new features. Rather than sacrificing quality or time, the team agreed to cut features if necessary.

Integration and testing week begun the 2nd of May, however integration begun the week prior and the team available ran a trial day the 1st of May. A process was agreed of working from 10am to 5pm, and notifying absence ahead. The early integration prototype served as proof of concept how the frontend can communicate with the backend, and much of the logic although rewritten was kept. Features would be worked on in separate branches in isolation to maintain workflow, however one problem encountered was large changes to the codebase caused issues during merges. The team shortened branch lifetimes so new features were merged back into main in easy sizes to handle. Each new feature was tested against the system tests and JUnit tests by Craymon, Sean, and Jim. Features were integrated or worked on in feature branches by Gwion, Jack, Cieran, and Shaun. Tyler reviewed code to submit to source and worked on the maintenance manual.

Where bugs were encountered by the test team, the leader was notified and the fixes were made on separate branches by the coders and merged back into main whilst new features were being worked on.

On the 4th of May, a .jar was produced and tested on the IS computers. After passing all system tests and JUnit tests, the program was declared complete and all code was pushed to source after review. The maintenance manual was largely complete. One final documentation review was held the 10th of May where the performance of all team members was evaluated and written up. All code and all documents will have been submitted complete with no sections missing on the 11th of May and acceptance testing done the 12th of May.

2.3 Final State of the Project

The team has successfully implemented the majority of features. Games can be saved to finish another time, and finished games loaded to view. The wording of FR10 is ambiguous enough to be left to interpretation as it did not specify whether only finished games can be replayed. As such, the team spent some time implementing the ability to view previous moves on unfinished games which may be an erroneous feature.

The state of the project may also contain some unforeseen bugs. System tests were not robust enough to cover all edge cases and after code submission, a bug was found with castling that allows the king to castle over attacked squares which breaks the castling rule.

Another issue that may be foreseeable is that some systems encounter issues with running the jar file. It was decided that the project be developed in Java .11 however it was discovered that IS Linux systems ran .17 and IS windows systems ran java .8. This was discovered during integration and testing week after the bulk of the codebase had been developed. Some language features utilised in the program had to be replaced with previous versions such as IsBlank() replaced with IsEmpty() when checking strings. There are also frontend issues with scenes not fitting the resolution of the application window on Ubuntu systems which may apply across other untested systems.

However despite those discovered bugs, the GUI is left as simple as possible as no style was specified. On startup, the application window for the GUI should appear and each button for navigating screens and each textfield for entering names is labelled clearly and appropriately, even if the front is a little small. The tile colours were assigned to compliment the piece colours, however the white pieces are borderless and do not contrast as sharply against white tiles which may present some issues for partially sighted users.

FR1 is satisfied. On player startup, to advance to a new board the users must enter their names in the field labelled for black and white players and those names are used to address the users when tracking the active player. The active player is also tracked on the board not only by their name, but an icon that displays the colour of the player which is the active player.

The screen has a chessboard which accurately represents the state of the board in the backend and is updated after every move. There is also a player dashboard to the left of the screen which tracks the active player and contains the buttons to offer a draw or resign for the active player. Therefore FR2 and FR3 are observed to be satisfied.

The player can select a piece of their colour and the valid squares for a selected piece to move will be highlighted. The player does not necessarily need to move a selected piece and can even select a different piece to see that pieces possible moves before making a move. If the player were to press a highlighted tile, the chessboard visually shows the updated state with the piece now in the correct position. The traditional movement rules of chess have all been implemented, including special moves such as en passant and partially correct castling. If a move would put the active players own king in check, it’s not shown as valid. Similarly, whilst the active player’s king is in check, the only valid moves shown are moves that would break check. Therefore FR4 is implemented completely, however FR5 movement rules are not fully implemented due to a castling bug.

The program can also detect check, restricting piece movement in such situations. If a player is placed in checkmate, the player Dashboard will be overlaid with a notification to the users that a certain player has won. This remains the case even as the player goes through the log. The chessboard with be disabled after a victory has been reached and no further moves can be made. This should satisfy FR6 and FR7.

At any point the player can quit by closing the application window, or by pressing quit on the top right corner of the chessboard screen. The player will be presented with the option to quit without saving, or quit with saving. It should be noted, it is assumed the player always wants to keep the current game saved, so the only way to delete a game file is to select the game they wish to play and quit without saving, thus calling a function to delete the file containing the game. This may not be how the user expects the program to function, however it was not explicitly written in FR8 as it does in FR11 that the tutor automatically saves the game. Therefore it is interpreted that a game is saved play by play and deleted when the user requests. The saved games are divided into finished and unfinished folders and any amount can be stored. Either can be loaded to replay the moves, but only in

unfinished games can users continue to make moves. This should satisfy FR8, FR9, FR10, and FR11.

One issue to note is that it is not explicitly written that saves should record the names of players. Currently, it is not implemented that a save stores the names of the black and white player. This may interfere with FR2 in the event a player loads an unfinished game or replays a finished game as it will not be immediately clear which user represents which side unless the user remembers. However the active player is still visually tracked in an obvious way, and names are replaced with “Black Player” and “White Player”.

2.4 Performance of Each Team Member

This section contains a summary of the responsibilities and performance of each team member. This was written by the team leader with the exception of the team leader’s own evaluation which was written by the team. Each team member has read their evaluation and the evaluations of each other and agreed with the content written below.

Gwion Hughes(Group Leader)

Gwions responsibilities as group leader were to lead and direct the team to produce high quality work, and promote teamwork, communication, and cohesion. As leader, Gwion was highly effective and well respected, bringing out the best of each member, taking into consideration everyone's strengths and weaknesses, and managing resources well. Organising meetings, locations, delegating tasks, and tracking the progress of the team providing valuable feedback at every step.

Gwion also contributed the majority of the front end codebase, communicating effectively with the rest of the team and developers to ensure that everything worked smoothly, and to produce high quality and cohesive code that was explained well to every team member. He also presented a code walkthrough for the front end code that helped the rest of the team work efficiently with the code base. The documentation he produced for the design specification was of high quality and vital to ensure that everyone understood the functioning of the front end code.

During integration week, he led the integration between the front and back end code which was vital in ensuring a final working product, bringing two vital components of the project together. He also delegated tasks extremely well during the week, and communicated effectively with team members yet again ensuring a smooth and problem-free week of integration.

Jack Thompson(Deputy Leader)

Jack Thompson responsibilities as Deputy leader involved covering for the group Leader during the leader’s absence and assisting with coordinating and managing the team. During integration and testing week, he was helpful at delegating tasks between him and Ciaran to fix bugs and implement new features.

Jack was also the technical main working on FR11 Storing and saving Games. His responsibilities from a technical point of view was to provide a save and load feature class. He communicated well with the backend lead, Shaun Royle, and understood the architecture and design of the program. Consequently, his code interfaced seamlessly to the backend. He was the backend's secondary developer, having spent six hours according to his blogs implementing Logs.java and further hours maintaining the class' during integration and testing.

During integration week, Jack was present through Monday to Thursday, communicated his absences clearly, and understood his daily task. He provided insight to the team about branch merging. His efforts to educate the team on how to use branches provided the team with a greater workflow over integration and testing week. He was an easy person to work with and was frequently paired with Ciaran Smith to work on features to do with replaying games and saving and loading. He maintained good code, commenting before pushing and was usually available to answer questions.

Tyler Lewis(QA Manager)

Tyler Lewis' responsibilities as Quality Assurance Manager was to maintain the documents and git repository's quality as well as provide processes inside the group to improve workflow. Tyler ran the minutes of each meeting and provided support for documentation writers. Tyler was a good source of advice on how best to keep documentation to the standards outlined by the project documentation. Tyler always maintained his blogs and submitted them on time, even sending the text file over to his team leader to review when submitting the text folder to gitlab was not possible.

He also provided the maven pom file that streamlined production of the project and ensured all members were working on the same code. Tyler ensured that all code submitted to source was commented and adhered to the Java Code standard. He was a guiding voice during documentation reviews, providing the templates for the documents to adhere to document standards. His advice was reliably professional and in keeping with his role. The processes authored by Tyler provided for the team to follow on development were respected and well thought out. He understood his responsibilities and focused on them, putting his all into performing his role well over spreading himself thinly to work on coding or design. Tyler reconfigured the git repository ahead of integration and testing week to address issues of team members working on code in different folders, ensuring that the code in active development was always the most up to date code.

Sean Hobson(Deputy QA)

Sean Hobson's responsibilities as Deputy Quality Assurance manager involved supporting Tyler in his task. Sean spent nine hours on the UI specification over the first three weeks where after he was moved to work on JUnit tests. He ran the minutes to a tutorial once and communicated well with the QA manager how best to take over that responsibility during the QA manager's absence. Sean contributed over twenty hours of work to producing the JUnit package up to the date of the integration and testing week.

During Integration and Testing week, he worked with Jim and Craymon to maintain the JUnit package. After each release, Sean would run the package and provide valuable

insight into any bugs found on the most current code iteration. This would always be communicated respectfully. He was aware of his responsibilities and always voiced any questions he needed answered to fulfil his role. Sean was seen as a dependable member of the team.

Craymon Chan

Craymon's earliest responsibility was working on the Test Specification. He has contributed over twenty five hours on the test Specification over the first eight weeks to the Test Specification. Using his knowledge of the system tests, he also contributed to the JUnit test package another nineteen hours of work from week three to week nine.

Craymon was not especially vocal with questions or opinions, but receptive to criticism and further improvements to the Test Specification document during review and continued working on System Tests through Integration and Testing. His work testing the program against the system tests alongside Jim provided a good grounds to track the progress of the team over the integration week. He was a dependable member of the team who worked through his tasks and didn't complain when he was assigned more work.

Ciaran Smith

Ciaran was responsible for working on the frontend, first prototyping using FXML and later prototyping integrating the frontend and backend before integration and testing week. His earliest implementation proved how the frontend could be connected to the backend. He was a dependable member who could work through complex tasks alongside others. He was vocal in letting his team know when and where he needed help.

Despite an early rocky start in which due to circumstances he was outside of the university for the first two weeks, he adapted well and picked up tasks without complaint when he joined the team. He worked with Jack and contributed ten hours of work to exploring potential methods of integrating the backend and front end. His methods and codes were commented well in consideration of the coders who'd come next to build upon his logic.

Jim Brown

Jim's primary responsibilities consisted of producing the System Test's document and working on the JUnit package. Jim was a communicative member of the team, often speaking on behalf of those he was working with on his tasks to provide progress updates and asking questions for feedback. He was also communicative when it came to requesting more responsibilities when he felt he was being underutilised.

During Wednesday informal meetings, Jim was a good source of feedback for documents being reviewed. During Integration and Testing week, he frequently reported any bugs or issues found by the testing team. He was also reliably critical of his own work during reviews, and could be relied on to communicate with his own small group to coordinate on shared tasks in his spare time.

Agreement reached.

Shaun Royle

Shaun Royle was the technical lead for the backend and piece logic of the chess tutor. Shaun Royle was the biggest influence on the architecture of the program and provided the idea of the top-down hierarchy model of how the frontend package would speak to the backend. He contributed forty eight hours to the backend code of the project over the first six weeks, by which stage much of the code was reportedly written to such a high standard the JUnit team had a hard time finding any tests that failed. He produced undoubtedly some of the highest quality code of the project, all commented and formatted to a good standard.

He was present in nearly every tutorial and informal meeting, and communicated with the group his absences making him an easy member of the team to coordinate with. He also provided valuable contributions in the design specification as well as an excellent code walkthrough to educate the team on how the logic of the game worked. Due to how well he communicated with the frontend coders, the task of how to integrate the front end and back end was made easier as they clearly communicated a design expectation before even beginning coding.

2.5 Critical Evaluation of the Team and the Project

In evaluating the teamwork, we must consider that this team contained individuals who were committed enough to work on average at least eight hours a week. The team progressed at a steady rate, never missing a deadline to review and submit documentation, and utilised tools such as GitLab issues to track documentation feedback as well as branches to improve workflow.

A good representation of the teamwork on display is the use of branches. Although not a suggested approach to the software engineering project, any group which utilises branches will not receive support on how to do this unless they educate themselves. The team was committed to integrate the personal experiences and skills of its individual members to make the development processes employed more efficient.

Individuals were committed to their assigned responsibilities and were trusted to handle matters in their competency. It wasn’t necessary for every member to learn JUnit, or for members besides the QA to know the document standards like the back of their hands. Productivity was maintained by trusting that other members of the team dedicated to their own tasks could complete their tasks and communicate effectively when problems arose. This was the case. Regular meetings and progress updates meant that team members did not go unsupported by the group either, and everyone was provided a fair opportunity to discuss any issues. All issues, such as development processes, were handled professionally on the spot and the team could come to a consensus on solutions to issues they discovered.

However the project itself could still be improved. The Functional Requirements were left a little ambiguous, and even though a project manager was always available to discuss with, the project manager in demystifying a FR has to set for each team a fair standard. Understandably, a project manager is sometimes reluctant to give a clear answer. Therefore, questions frequently asked about the project could be noted and the FRs for the project further demystified for future groups.

The Group Nine submission can also be further refined with time. The style of the screen and buttons is primarily the default JavaFX styles to the objects. This simple design doesn't complicate the UI, but does leave it looking a bit unappealing and spartan. The project could be improved by setting more clear requirements for the UI, such as requesting accommodations for partially sighted users.

During the course of this software engineering project, it has become evident how successfully a group can work when responsibilities are divided in a thoughtful manner and the team agrees on processes to develop the project. These processes were respected by individual members who themselves felt respected and valued in the team by their contributions.

During meetings, giving feedback was seen by each member as a way to support eachother, not an opportunity to criticise eachother. Members felt that feedback was coming from a fair place and was welcomed by all members. From keeping code commented to maintaining documents and conducting code walkthroughs, the team worked with consideration that there were other members who may need to access the material they produced. That consideration whilst working on individual tasks must be an essential skill to working in and maintaining a positive team environment.

In consideration of the product achieved and the teamwork demonstrated, I would award this team a minimum of a 60% for their efforts.

3. REFERENCES

[1] SE.QA.RS-CS22120 - Chess Tutor Requirements Specification

[2] UI-Spec-Docu-GP9 - UI Specification Document and Presentation

[3] Design-Spec-GP9 - Design Specification

[4] Test-Spec-GP9 - Test Specification

4. DOCUMENT HISTORY

Version	Issue No.	Date	Changes made to document	Changed by
0.1		10/05/23	Document started. Written feedback on all members of the team.	GWH18
0.2		10/05/23	Agreement made over description of each member's performance. Description of group Leader responsibilities and performance added.	TYW1
0.3		10/05/23	Completed body of document for reviewing.	GWH18