

Steven-moore-ass1

by Steven Moore

Submission date: 25-Apr-2023 03:45PM (UTC+0100)

Submission ID: 203381251

File name: 25258_Steven_Moore_Steven-moore-ass1_490085_246363711.docx (2.11M)

Word count: 3013

Character count: 17891

The background of the cover is a photograph of a desk. In the top left, a portion of a laptop keyboard is visible. To the right, there are two potted plants: a larger one with long, green, variegated leaves and a smaller one with thick, green, pointed leaves. In the bottom center, a white notebook with horizontal lines is open, with a white pen resting on it. A large blue rectangle is overlaid on the left side of the image, containing the title and authors' names.

SOFTWARE ENGINEERING DESIGN & TESTING

Software Engineering
Principles and Testing

UWTSD
Steven Moore
Michael Dacey



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Contents

Introduction	3
I. Evaluation of designs produced	3
A. Agile Development	3
1) 12 principles of agile development	3
B. User Stories	4
1) User story map – release 1	4
2) User stories – release 2	4
3) User stories – release 3	5
C. CRC Cards	5
1) Sequence diagram	8
D. Testing	8
1) Black box testing	9
2) White box testing	10
E. Advantages and Disadvantages of Agile [13]	10
II. Bibliography	11

INTRODUCTION:

This report evaluates the software design for a connect 4 game which was undertaken with an agile approach, making use of user stories, CRC (Class-Responsibility-Collaboration) cards, and testing strategies typically seen in agile software development projects. The report also reflects upon the author's experiences in producing the design, assessing the strengths and weaknesses of the agile approach, and highlighting the importance of software testing in agile development.

In an agile development context, user stories and user story maps are used for requirements capture and prioritisation. The benefits of user stories lie in their flexibility and adaptability, allowing for iterative development cycles and ease of prioritisation. Software testing, specifically black and white box testing, is crucial for identifying defects early in the development process. The concept of "testable" code is discussed, and the principles of unit testing in agile development are investigated.

I. EVALUATION OF DESIGNS PRODUCED

This section aims to evaluate the designs we produced as a team and to reflect upon the overall experience in producing them in an agile environment.

A. Agile Development

Agile development is a set of software development principles and practices that emerged from the Agile Manifesto, created by a group of software developers back in 2001 [1]. The Agile Manifesto values individuals and interactions, working software, customer collaboration, and responding to change over following a plan.

*“Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan”
-Manifesto for Agile Software Development*

1) 12 principles of agile development

The Agile Manifesto consists of 12 principles that serve as **guidelines** for Agile development. These principles are [2]:

1. Prioritise individuals and interactions over processes and tools.
2. Deliver working software frequently, with a preference for shorter timescales.
3. Collaborate with customers and stakeholders throughout the project.
4. Respond to change and embrace it as a source of competitive advantage.
5. Build projects around motivated individuals and give them the tools and support they need.
6. Use face-to-face communication as much as possible for maximum effectiveness.
7. Measure progress primarily by working software and customer satisfaction.
8. Maintain a sustainable pace of work for the team and avoid overloading them.
9. Strive for technical excellence and good design to enhance agility.
10. Keep things simple and focus on what's necessary to achieve the project's goals.
11. Allow self-organising teams to make decisions and drive the project forward.
12. Reflect regularly on the team's performance and adjust accordingly for continuous improvement.

These principles emphasise the importance of individuals, collaboration, responsiveness to change, and iterative improvement in Agile development. They prioritise delivering value to customers, adopting a collaborative team mindset, and embracing change as an opportunity for improvement. By adhering to these principles, Agile teams aim to achieve customer satisfaction, maximise flexibility, and continuously improve their processes and products.

[3] Some popular Agile methodologies include Scrum, Kanban, and Lean, among others. These methodologies provide frameworks for implementing Agile principles in different ways, but they all share common characteristics, such as iterative development, continuous feedback, and cross-functional teams.

B. User Stories

[4] A user story is a concise description of a feature, functionality, or requirement from the perspective of an end user or customer. It is typically used in Agile and Scrum methodologies as a means of capturing and communicating user requirements in a simple and understandable way.

In our team project we started by producing some user stories in order to determine the basic user requirements for a web-based version of Connect 4. The full user stories and user story map is available here:

https://miro.com/welcomeonboard/QTRXMWV4RWszRGNyRnJvRVB5YThUdUhwQkdhWGY4VWxBWXA0RWFBsjY5QVE5Qnhuc3h1YkJvd3oxNDRXMkFFRHwzNDU4NzY0NTQ1MDgyMjQwOTg5fDI=?share_link_id=280831774330

1) User story map – release 1

The first release of our user stories can be seen below. These stories aimed to achieve the most basic but functional version of online Connect 4. These include joining a friend via room code, dropping counters into desired columns, checking the game-board for a line of 4 counters in a row. The player would then receive a notification telling them if they had won or lost the game.

In hindsight it would probably be better to put the multiplayer feature and game logic for detecting winning and losing in the second or even third release. Since the players could play from one computer and would know themselves when they have won or lost.



Figure 1 - Screenshot from Miro board, release 1 - user stories

Another story which could be pushed back to a later release is the “lobby options” – restarting the game story. However, this could be considered an important task for the initial release since users will need to be able to refresh the game board to play multiple games.

These user stories helped our agile team maintain focus on delivering value to the end users, encouraging collaboration and feedback, and promoting flexibility and adaptability to changing requirements throughout the development process. They served as a lightweight and user-centric approach to capturing and managing the requirements in this agile project.

2) User stories – release 2

The user stories below represent the second release for the connect 4 program. They currently include a lobby browser where players can browse a list of public lobbies which allow users to challenge random players to a game. The second release also includes a user story which allows players to create an account to see their game history.

The main objective of the second release was to capture the user requirements for the following rules:

Capturing Opponent's Counters: A player can seize an opponent's counter by cornering it between two of their own counters horizontally, vertically, or diagonally.

Destroying Captured Counters: If a player wins three of their opponent's counters, those counters are removed. Any "holes" left by those counters are filled by counters that can "drop" down a column to fill the hole. Counters can be captured

and destroyed horizontally, vertically, or diagonally. However, counters that drop down do not capture or destroy opponent's counters.

These amendments to the traditional rules of Connect 4 add an additional layer of strategy to the game by allowing players to capture and potentially destroy their opponent's counters, and filling in holes with counters that drop down. The last user story on the right which details choosing whether to capture a counter or create a line of 4, could be considered an epic.

[5] In Agile development, an epic is a large and high-level user story that represents a substantial piece of work or a major feature that requires further breakdown and refinement into smaller, more manageable user stories.



Figure 2 - user stories - release 2

3) User stories – release 3

The third and final release of user stories include features such as board customisation, adding friends, access a tutorial and introducing a rank system. Perhaps AI opponents could have been considered in this release or even a later release down the line.

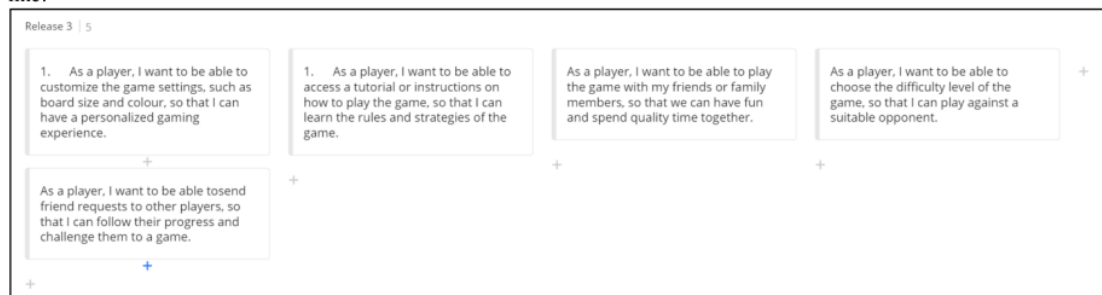


Figure 3 - user stories release 3.

Having multiple release versions of user stories in Agile development allows for incremental delivery, enabling faster value delivery and early feedback, while also providing flexibility and adaptability to changing requirements, priorities, and mitigating risks any risks that may pop up during development of the discussed features.

C. CRC Cards

[6] CRC cards (Class, Responsibilities, and Collaborators) are a visual and collaborative technique used in software development (typically object-oriented programming) to capture and document the responsibilities and collaborations of classes or objects in an object-oriented system.

They are particularly useful when trying to get (in particular) programmers on the same page when it comes to understanding how we (as a team) expect to build the software. Demonstrating the required relationships between each module of the game and even demonstrating what types of data are needed by each object.

In this project we managed to break down the user stories into 5 CRC cards, these are shown below (full version available at: https://miro.com/welcomeonboard/RlpjekNDSUtlY1J1Z2p1OG83MU9tT2h3NzBHbjFGVV BmNUZiWFZQcXUyRWloSTFlbFFyRFVPeVVEQmgwWUF4dXwzNDU4NzY0NTQ1MDgyMjQwOTg5fDI=?share_link_id=148656060128).

CRC Cards			
Class name: Player <table> <tr> <td> Responsibilities - Choose a column to drop a counter in - Keep track of the number of counters and their position on the board - Check for winning conditions by checking for a line of 4 counters </td><td> Collaborators - Game board: to update the board with the player's move </td></tr> </table>	Responsibilities - Choose a column to drop a counter in - Keep track of the number of counters and their position on the board - Check for winning conditions by checking for a line of 4 counters	Collaborators Game board: to update the board with the player's move	The player class needs to be able to choose a column to drop a counter in, this class will only directly collaborate with the front end of the program – the game board.
Responsibilities - Choose a column to drop a counter in - Keep track of the number of counters and their position on the board - Check for winning conditions by checking for a line of 4 counters	Collaborators Game board: to update the board with the player's move		
Class name: Game Board <table> <tr> <td> Responsibilities - Maintain the state of the game board - Check for available slots in a column - Update the board with a player's move - Check for winning conditions by checking for a line of 4 counters </td><td> Collaborators - Game board: to update the board with the player's move </td></tr> </table>	Responsibilities - Maintain the state of the game board - Check for available slots in a column - Update the board with a player's move - Check for winning conditions by checking for a line of 4 counters	Collaborators Game board: to update the board with the player's move	The game board has various responsibilities including maintaining the state of the game, checking available slots in columns and updating itself with the player's moves.
Responsibilities - Maintain the state of the game board - Check for available slots in a column - Update the board with a player's move - Check for winning conditions by checking for a line of 4 counters	Collaborators Game board: to update the board with the player's move		
Class name: Game <table> <tr> <td> Responsibilities - Initialize the game board and players - Control the flow of the game - Determine the winner of the game - Keep track of the game's progress </td><td> Collaborators - Player: to receive the player's move and update the game's progress - Game board: to update the state of the game board </td></tr> </table>	Responsibilities - Initialize the game board and players - Control the flow of the game - Determine the winner of the game - Keep track of the game's progress	Collaborators - Player: to receive the player's move and update the game's progress - Game board: to update the state of the game board	The game class refers to the backend of the program which will be responsible for initialising and updating the game. It will also be responsible for checking for any winning moves. It collaborates with the game board to receive the player's move and to update the state of the game.
Responsibilities - Initialize the game board and players - Control the flow of the game - Determine the winner of the game - Keep track of the game's progress	Collaborators - Player: to receive the player's move and update the game's progress - Game board: to update the state of the game board		

Class name: Counter <table> <tr> <th>Responsibilities</th><th>Collaborators</th></tr> <tr> <td> - Represent a single counter on the game board </td><td> - Game board: to place the counter on the board </td></tr> <tr><td></td><td></td></tr> <tr><td></td><td></td></tr> <tr><td></td><td></td></tr> <tr><td></td><td></td></tr> </table>	Responsibilities	Collaborators	Represent a single counter on the game board	Game board: to place the counter on the board									The counter class represents the pieces on the game board, they don't have many responsibilities or collaborators since all they need to do is represent a single counter on the game board.
Responsibilities	Collaborators												
Represent a single counter on the game board	Game board: to place the counter on the board												
Class name: Winning Conditions <table> <tr> <th>Responsibilities</th><th>Collaborators</th></tr> <tr> <td> - Check for winning conditions on the game board - Determine if a line of 4 counters has been achieved </td><td> - Game board: to access the current state of the board </td></tr> <tr><td></td><td></td></tr> <tr><td></td><td></td></tr> <tr><td></td><td></td></tr> <tr><td></td><td></td></tr> </table>	Responsibilities	Collaborators	- Check for winning conditions on the game board - Determine if a line of 4 counters has been achieved	Game board: to access the current state of the board									The winning condition class is responsible for checking if any player has made a winning move, it does this by collaborating with game board to check whether a line of 4 counters of the same colour have been placed.
Responsibilities	Collaborators												
- Check for winning conditions on the game board - Determine if a line of 4 counters has been achieved	Game board: to access the current state of the board												

1) Sequence diagram

This section takes the CRC cards we developed and places them into a sequence diagram. In the context of CRC cards, a sequence diagram is a type of [7] UML (Unified Modelling Language) diagram that illustrates the interactions and exchanges of messages between objects or classes in a system or software component. It visually depicts the sequence of messages or method calls between objects or classes over time, showing the order of interactions and their flow.

The sequence for the Connect 4 program can be seen below (full version:

https://miro.com/welcomeonboard/RlpjekNDSUdY1J1Z2plOG83MU9tT2h3NzBHbjFGVVBMNUZiWFZOcXUyRWloSTFlbFFyRFVPeVVEQmgwWUF4dXwzNDU4NzY0NTQ1MDgyMjQwOTg5fDI=?share_link_id=888897054315).

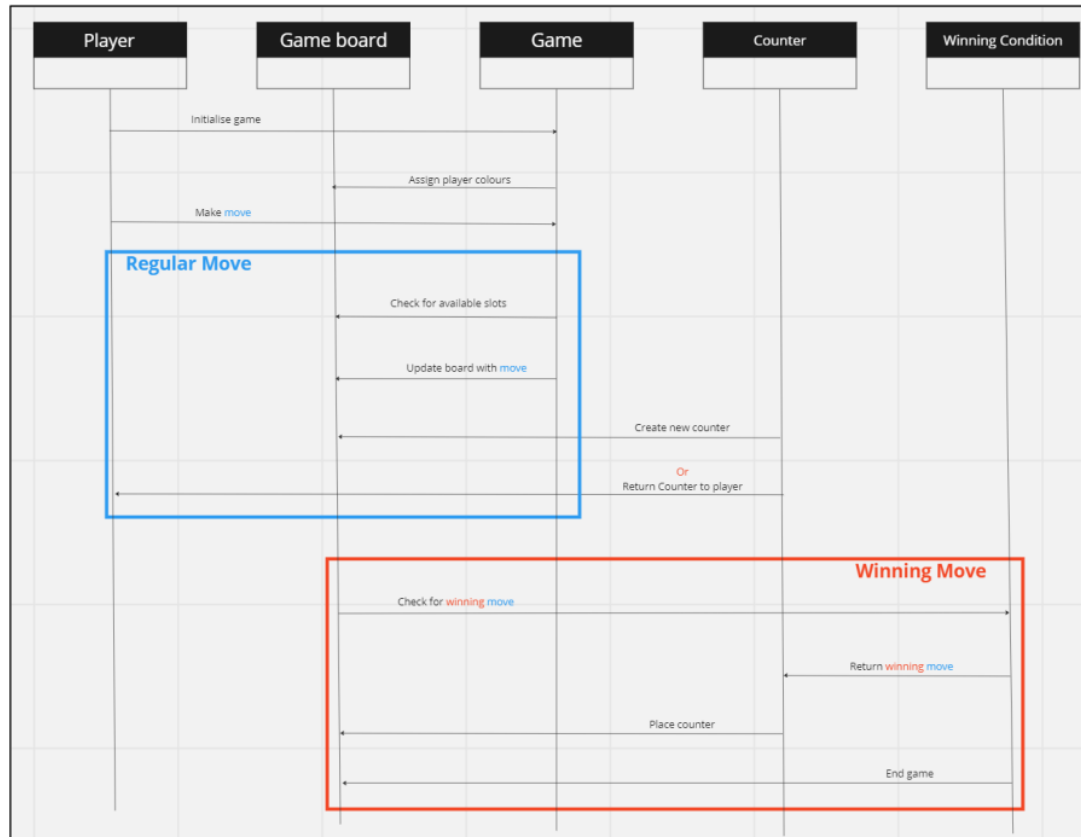


Figure 4 - screenshot of sequence diagram

As shown above a sequence diagram can be used in conjunction with the CRC cards to further refine the design of the system. CRC cards provide a high-level overview of classes or objects and their responsibilities and collaborations, while this sequence diagram provides a more detailed and visual representation of the interactions and flow of messages between these classes. Sequence diagrams can help in identifying potential issues or improvements in the design, as well as in documenting the dynamic behaviour of the system [8].

D. Testing

Software testing is a critical aspect of software engineering that plays a crucial role in ensuring the quality and reliability of software applications. It involves the process of systematically identifying and fixing defects, errors, or bugs in software before it is deployed to end-users. Testing is essential in detecting potential issues, improving performance, and enhancing the overall user experience.

[9] There are various approaches to software testing, including black box testing and white box testing. Black box testing focuses on evaluating the functionality of a software application without considering its internal structure or code. It involves testing the software from the perspective of an end-user, without any knowledge of the internal workings of the software. White

box testing, on the other hand, involves evaluating the internal structure and code of the software application to identify defects and ensure proper coding practices.

1) Black box testing

There are two test strategies which the author has used when coming up with black box tests:

Test to pass:	Test to fail:
- First tests to try. - Not trying to break the code (Happy path should always pass).	- be sneaky, conniving, and devious to force bugs out (also known as <i>error-forcing</i>). - Perform after "test to pass". - Probe common weaknesses in software. - Anticipated errors can be considered either test to pass or fail, the distinction is not important e.g., not entering correct password multiple times.
Test for valid moves: place a counter in a valid column.	Invalid move: Test the game's behaviour when an invalid move is made, such as placing a game piece outside the boundaries of the game board, placing a game piece on top of an existing game piece, or placing a game piece in a filled column.
Test for win/loss conditions: place 4 counters in a row.	Incorrect win/loss detection: Test the game's ability to correctly detect win/loss conditions. For example, deliberately create a scenario where four game pieces of the same colour are not detected as a win, or where the game declares the wrong player as the winner or loser.
Test for game state updates: check that the game updates the board after making a move.	Game state inconsistency: Test the game's behaviour when the game state becomes inconsistent or corrupted. For example, manipulate the game state to have conflicting or invalid values, such as having more game pieces of one colour than allowed, or having an incomplete game state after a move.
Test for game restart/reset: try to restart the lobby or close the game.	Invalid inputs: Test the game's handling of invalid inputs, such as providing incorrect or unexpected inputs through user interactions, such as mouse clicks or touch events. For example, providing non-numeric characters as input, entering extremely large or small values, or entering input in an unexpected format.
Test for user interactions: test that user's can enter room codes, make moves etc..	Compatibility issues: Test the game's compatibility with different web browsers, devices, and screen sizes by deliberately testing with unsupported or incompatible configurations. For example, testing the game on an outdated web browser that is not supported, on a device with low screen resolution or limited resources, or on a non-compatible operating system.
Test for error handling: try to place a counter in a invalid spot.	Boundary cases: Test the game's behaviour in extreme or boundary cases, such as placing the maximum number of game pieces in a single column, filling the entire game board with game pieces, or simulating scenarios where the game board is almost full.
Test for compatibility: test the game on different web browsers and devices.	Network failures: Test the game's behaviour when encountering network failures or connectivity issues. For example, deliberately disconnecting the internet connection during a game session, simulating slow or unreliable network conditions, or testing the game's offline mode functionality.

2) *White box testing*

White-box testing, also known as structural or code-based testing, involves testing the internal structure and logic of the software. Applied to our online Connect 4 game, here's how white-box testing can be used:

[10] **Code coverage testing:** White-box testing can ensure that all code paths and branches in the game's source code are executed and tested. This can be done using techniques such as statement coverage, branch coverage, and path coverage to achieve a high level of code coverage and identify any untested or unreachable code.

[11] **Boundary value testing:** White-box testing can focus on testing the game's behaviour at the boundaries of input values and game states. For example, testing the game's behaviour when the game board is empty, when it's almost full, or when it's completely full, to identify any potential issues or unexpected behaviours.

[12] **Unit testing:** White-box testing can involve writing and executing unit tests for individual functions or methods within the game's source code. This can help identify and fix issues at the function or method level, ensuring that each function or method performs as expected and returns the correct output given specific input conditions.

These tests ensure the correctness, robustness, and security of the game's source code and internal logic. White-box testing can complement black-box testing to provide a comprehensive testing approach for the game.

E. Advantages and Disadvantages of Agile [13]

Through taking part in this project whilst utilising the agile methodology, the author has noticed several strengths and weaknesses with the agile methodology.

One of the biggest strengths of the Agile methodology is its ability to be adaptable and responsive to change. Agile promotes flexibility and embraces changes in requirements, priorities, and even project goals during the development process. This contrasts with traditional methodologies that often follow a strict, predefined plan.

At the same time the author feels this can be its biggest weakness. For example, it can be difficult for developers to get on with the work as the user requirements and required features constantly change. One way this could be mitigated is applying "sign offs" to specific features which cannot be changed until the original design is complete. Allowing the developers time to develop the core features for the program.

Conclusion

In conclusion, the agile approach used in the software development project for the Connect 4 game proved to be effective in incorporating user stories, CRC cards, and testing strategies. The experiences gained during the design process provided valuable insights into the strengths and weaknesses of the approach, highlighting the need for thorough software testing in agile development.

II. BIBLIOGRAPHY

- [1] "History: The Agile Manifesto," agilemanifesto.org, [Online]. Available: <https://agilemanifesto.org/history.html>. [Accessed 11 04 2023].
- [2] "Principles behind the Agile Manifesto," [Online]. Available: <https://agilemanifesto.org/principles.html>. [Accessed 11 03 2023].
- [3] "Top 5 main Agile methodologies: advantages and disadvantages," [Online]. Available: <https://www.xpand-it.com/blog/top-5-agile-methodologies/#:~:text=There%20are%205%20main%20Agile,%2C%20Lean%20Development%20e%20Crystal..> [Accessed 11 04 2023].
- [4] "User Stories: What They Are and Why and How to Use Them," [Online]. Available: <https://www.nimblework.com/agile/user-stories/>. [Accessed 11 04 2023].
- [5] "Agile user story example: how to write user stories (template included)," [Online]. Available: <https://slickplan.com/blog/user-story>. [Accessed 04 11 2023].
- [6] "Class Responsibility Collaborator (CRC) Models: An Agile Introduction," [Online]. Available: <http://agilemodeling.com/artifacts/crcModel.htm>. [Accessed 11 04 2023].
- [7] R. Jakubov, "Interaction Diagrams: Sequence and Collaboration Diagrams," [Online]. Available: http://www.sci.brooklyn.cuny.edu/~kopec/CIS_763/CIS763_00/ClassPresent_1/UML_outline.htm. [Accessed 11 04 2023].
- [8] "UML Sequence Diagram Tutorial," 11 04 2023. [Online]. Available: <https://www.lucidchart.com/pages/uml-sequence-diagram>. [Accessed 11 04 2023].
- [9] "Black Box Vs White Box Testing," [Online]. Available: <https://www.practitest.com/qa-learningcenter/resources/black-box-vs-white-box-testing/#:~:text=The%20Black%20Box%20Test%20is,is%20carried%20out%20by%20testers..> [Accessed 11 04 2023].
- [10] "What is code coverage?," [Online]. Available: <https://www.atlassian.com/continuous-delivery/software-testing/code-coverage>. [Accessed 11 04 2023].
- [11] "Boundary Value Analysis and Equivalence Partitioning Testing," [Online]. Available: <https://www.guru99.com/equivalence-partitioning-boundary-value-analysis.html>. [Accessed 11 04 2023].
- [12] "unit testing," [Online]. Available: <https://www.techtarget.com/searchsoftwarequality/definition/unit-testing#:~:text=Unit%20testing%20is%20a%20software,independently%20scrutinized%20for%20proper%20operation..> [Accessed 11 04 2023].
- [13] "Advantages and Disadvantages of Agile Project Management," [Online]. Available: <https://activecollab.com/blog/project-management/agile-project-management-advantages-disadvantages>. [Accessed 11 04 2023].

FINAL GRADE

84/100

GENERAL COMMENTS

Instructor

Eval of Design (20): 18

User Stories (20): 19

CRC (20):18.5

SW Testing (20):10.5

Testing (20): 18

Feedback:

=====

Very good design evaluation and examples to illustrate the approach. Could have provided a little more evaluation the use of CRC cards.

You needed to provide more evaluation of black/white box testing.

Tests required for the application are appropriate, the discussion of unit testing could have been expanded upon.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11
