Team Project Plan For Shell Shocker

Table of Contents:

Who: The Shell Shocker team has the following roles:

- Richard Dong; Lead Programmer, Development Manager- Implements coding that meets the requirements for the majority of the game. The programmer will do this while following ethics of Java and code documentation.
- Patrick Harold;

  Graphics Designer- Works alongside the Lead Programmer. The designer creates the team's original game graphics to be creative and entertaining.

  Project Manager- The person who solves the problems of consistent reporting, risk mitigation, timeline, and cost control. The manager will be responsible for creating a flawless integration of different aspects of the project, documenting and creating a presentation for the final product.
- Chance Hofmann; Database Manager, Developer- Works alongside the lead programmer to create a database for storing player profiles and information in an efficient manner.
- Tharindu Wanigatunga; Training- Records proper documentation for the game code in a neat and efficient way. The training role also requires the creation of a instructor or computer-based training solution designed to help users better understand how the system works and how the system may be used.

What:

This project will be a platform-based action-adventure game in the style of classic 8-bit programs. The program will utilize a database which will store player login information as well as save information from playing the game. The game involves user interaction with non-playable enemies in a grid-based map system. Users have a character progression system involving experience points, which are gained in battle and carry the ability to level up a character. Leveling up allows the user's character to have upgraded weapons. The game will contain an interaction menu detailing a character's inventory and statistics that will present itself whenever the game is paused. Also, the entire game will have a storyline, following the path of a hero as he attempts to save his planet.

Why:

Shell Shocker's primary objective is to make an entertaining platform-based game in the style of classic 8-bit programs. Shell Shocker will be a homage to many of the early games that built the industry. It will incorporate attributes of the games they played in the 1980s and 1990s to remind veteran gamers of what made those games so special to those that played them.

When:

In order to complete this project and facilitate time management, the Shell Shocker team will use a project timeline. There are several major deadlines that the Shell Shocker team will adhere to for maximum efficiency:

November 6, 2013: Project plan and timeline submission
December 20, 2013: Amended project plan and timeline submission
January 1, 2014: Game code completed
February 7, 2014: Final project submission
March 21, 2014: Presentation scheduling

Scope Management:

The 8-bit, side-scrolling RPG game is about a story of the human colony of Azaran. Azaran, a peaceful society, will face an alien invasion by the villainous Zorths. The Zorths are a violent species that threatens to kill everyone on the planet. The Zorths depend on a large mothership that currently resides in space. The ship must be destroyed to save the human civilization and may be done so on account of its primary weakness: its main reactor is highly combustible and can be reached from the external of the ship. However, the journey to the reactor is treacherous. Minions and gun towers are in the passageways and tunnels that lead to the reactor. Additionally, at the reactor, large enemies guard and prevent attackers from disabling the reactor. The humans launch an attack on the mothership in hopes of stopping an invasion of their home planet. One spaceship pilot is generous enough to drop the user off with an attack vehicle called the Ragnar, which has special weapons for fighting and rocket boosters for flight. The user's job is to pilot the Ragnar to the reactor and destroy the Zorth mothership by destroying the main reactor. Thus, the planet Azaran and the humans who inhabit it may be saved.

Requirements Management:

The Shell Shocker game is a side-scrolling action adventure role-playing game in an 8-bit style for a single user. It focuses on the invasion of the human colony Azaran by the Zorths, a race of planet-destroying aliens. In the game, the humans living on Azaran mount a counterstrike against the Zorth mothership. The user is involved in the strike as the pilot and commander of the Ragnar, a flying tank-like machine used by the military of Azaran. Equipped with rocket boosters for flight control and weapons for combat, the Ragnar must disable the mothership by boarding and destroying its main flight reactor.

The game contains a set of three individual checkpoints that must be unlocked. All unlocked checkpoints are saved to the game profile. Through interaction with a database, the user has the ability to save current score, player health, checkpoint location, and other character-pertinent information into a file. The information can be used later as the user returns to the same game file. A user will have the option to delete profiles.Throughout the game, a cumulative point score and a high score will be presented to the user. The high score is the maximum score among all game profiles saved in the database.

A menu system allows the user to navigate through many game features, including a tutorial section for new users, an area in which a new game profile can be created or to which an old saved file

can be accessed, and an exit to the program. The creation of a new profile requires users to enter a name not previously saved to create a player profile and to save high scores. The access of old files allows returning users to resume previous play at different unlocked checkpoints.

The game features many characters in the form of moving vehicles. The main vehicle piloted by the user, the Ragnar, journeys its way through the Zorth mothership. The path that the Ragnar takes has levels and platforms that carry enemies The Ragnar interacts with the enemy characters whose main goal is to rebuff any progress made by the user. Using weapons such as bombs, lasers, cannons, and shell packages, the Ragnar plows through the opposing forces. Enemies are categorized into 4 groups: "strong-minded" minions, "weak-minded" minions, stationary towers, and a guardian. The "strong-minded" minions set out to track down and destroy any invaders. The "weak-minded" minions are vehicles that move back and forth (with the goal of dodging the user's attacks) while guarding certain levels or platforms. They attack when motion of the user is sensed. Towers, which are stationary and placed next to entrances and the along the edges of pathway walls, shoot at enemy targets. Lastly, the guardian protects the main reactor; it uses a variety of weapons for attacking and has several shields for defense. Along the way to the reactor, certain packages exist, which provide replenishments for the Ragnar's health, shield, and munitions. The Ragnar acquires these packages by making contact with the containers.

The player is capable of "leveling up" and "leveling down" in several ways. Along the path to the reactor, if the player reaches one of two checkpoints, the Ragnar's weapons systems are upgraded to impact an increased amount of damage as the enemies become more numerous. However, when the Ragnar loses enough health and reaches what is defined as a critical level, the Ragnar's weapons systems are downgraded to its initial level.

Points are awarded to the user in a variety of situations. The most common way of gaining points is by destroying enemies, which are worth a different quantity of points based on their type. Additionally, specific point-garnering packages appear throughout the game that grant a given amount of points when touched by the Ragnar. Moreover, points are deducted if the user's health points reach a certain critically low level. The points are also deducted the longer it takes to complete the game.

Whenever a user presses the arrow keys up, right, or left, the Ragnar attempts to move accordingly. Gravity also affects the Ragnar's movements whenever the vehicle is airborne and rocket boosters are not in use. The Ragnar is not able to move through walls, enemies, platforms, and other solid objects (not including the special packages). The view shown to the user is only a partial view of the entire map of the game, and whenever the Ragnar moves, the view screen changes in a similar amount; the Ragnar always remains left of center in the game display. The logic of the game's platforms are simple: whenever an object attempts to cross the platform either due to gravity or to keyboard commands, the movement is denied. Restricting movement prevents objects from falling through floors and from going through ceilings.

In conclusion, the Shell Shocker game is a side-scrolling action-adventure game developed in the 8-bit style. Through the use of multiple controls, level-up conditions, and score earning, the game encompasses a unique and enticing experience.

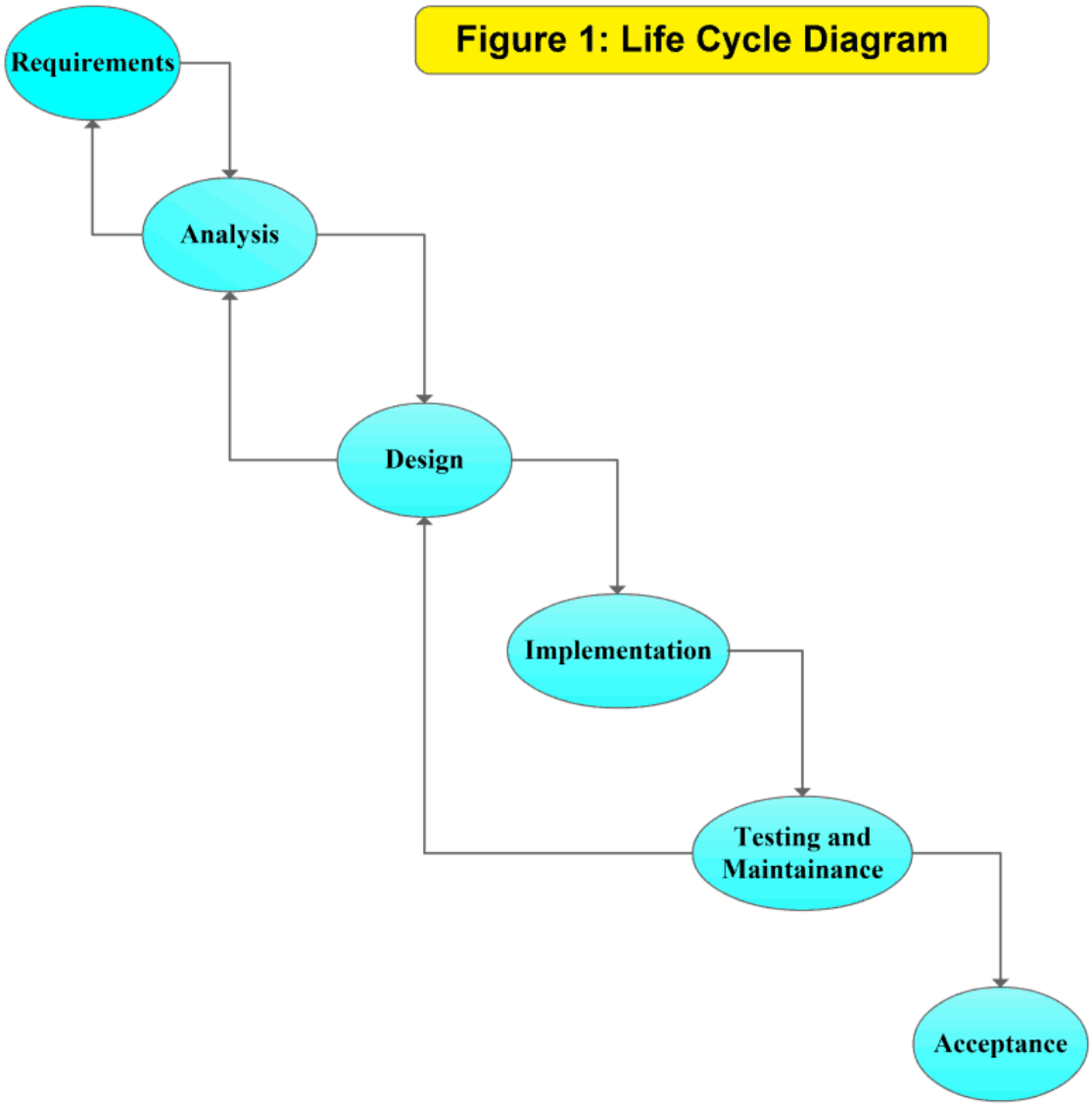

Figure 1: Life Cycle Diagram

Project

- Requirements
    - The guidelines for the requested game are given and will be fully understood.

- Analysis
    - The team completes the task of determining needs or conditions of the requested product. The requirements given by stakeholders - analyzing, validating, and managing software or system requirements - are all taken into account.

- Design
    - The game will be designed so that it is an action adventure Role-Playing Game in an 8-bit style. The player will control and manage his character as they progress through the game. The player will battle hostile non-playable characters as they play in real time and may defeat them and collect other items in order to gain points and level up.

- Implementation
    - Code will be written to meet design specifications a neat and efficient manner by the team of programmers. Documentation will occur; its purpose is to make the job of future maintenance and enhancement easier as code is easier to read.

- Testing and Maintenance
    - Testing will be done on the game to ensure that the created product runs smoothly in the way described in the description. Different tests will be run to find software defects. The first tests will be aimed at giving feedback for all new code. At an alpha stage, a test plan is written and bugs are reported. During a beta stage, clear assignments will be given to the testing team on a daily basis.

- Acceptance
    - The game will be available for use after extensive testing for any bugs that inhibit and decrease the appeal of game play. Requirements will have been met thoroughly.

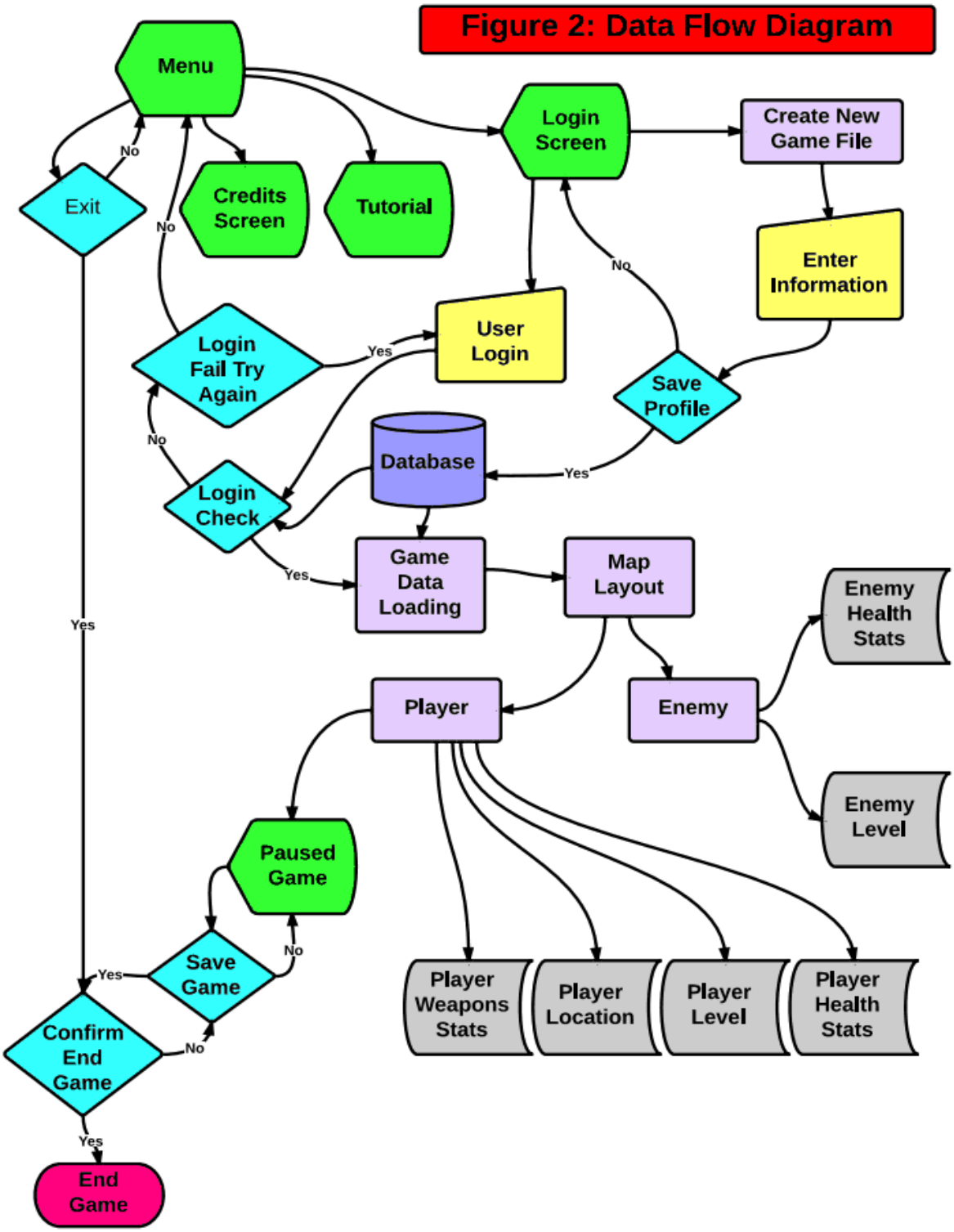
Figure 2: Data Flow Diagram
Menu
Exit
Credits Screen
Tutorial
Login Screen
Create New Game File
Enter Information
User Login
Login Fail Try Again
Save Profile
Database
Login Check
Game Data Loading
Map Layout
Enemy Health Stats
Player
Enemy
Enemy Level
Paused Game
Save Game
Confirm End Game
End Game
Player Weapons Stats
Player Location
Player Level
Player Health Stats
No
Yes

Figure 3: Storyboard

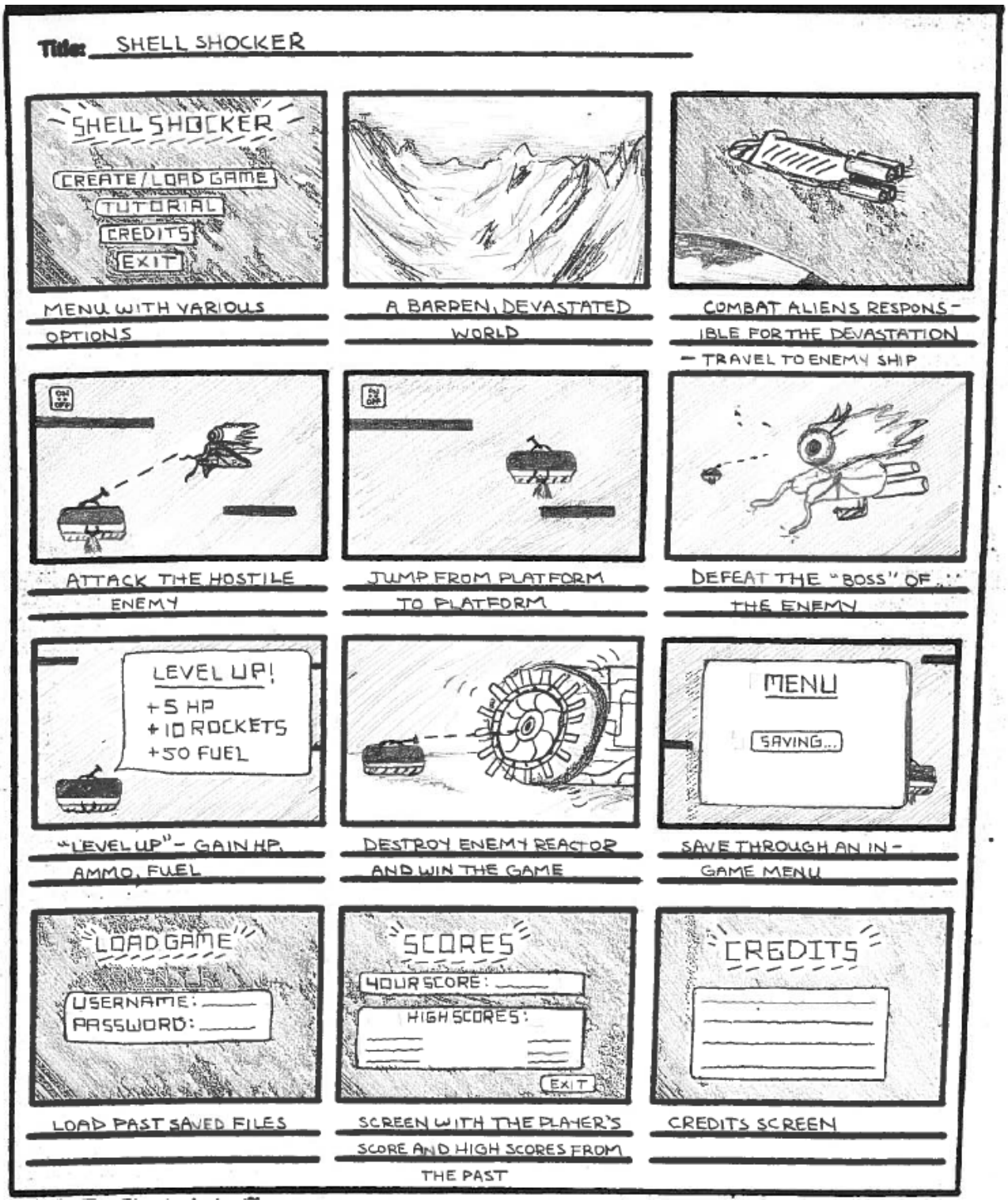

Figure 4: UML Diagram

- See the file titled UML Diagram in the project plan folder due to its large size.

Quality Management:

Quality will be ensured with periodic quality checks to make certain that the program runs as expected without glitches or errors. These checks include gameplay testing, the use of multiple programmers to address coding errors, multiple backups of important files, and compatibility checks on multiple systems. After the game is initially implemented, repetitive tests and modifications of the code will occur to insure a more succinct end product.

Resource Management:

The school system will provide all resources, specifically from the computer science lab, including computers, all necessary software, and internet onsite. When team members are not at school, all resources will be handled by the team members themselves. If any the purchase of any new equipment becomes necessary over the course of the project then the cost will either be split, or be purchased by a single team member who will retain ownership of the item after the project has concluded.

Financial Management:

Costs will usually be split between team members if the item purchased is not very expensive (i.e. less than $50). Occasionally, items will require purchasing that exceed this limit. In such situations, one team member will volunteer (or if nobody volunteers, one will be assigned) to purchase the item. The person who purchases the item will retain ownership. Any personal items that are broken by other team members will be reimbursed for compensation purposes.

Communications Management:

The primary means of communication between team members will be e-mail. Google Drive will be used in order to share project documentation and presentations. Dropbox will also be used to share necessary files.

Project Change Management:

If any changes to the project must be made from the initial plan then they will be addressed by the entire team and a decision will be made in order to ensure that the project still meets all requirements and that the proposed changes will be consistent with the rest of the project's implementation.

Risk Management:

Risk: Hardware failure or loss of data
Projected Solution: Frequent uploading of files to an online cloud server (such as Google Drive), as well as keeping multiple copies of each file between team members and also, the team will keep detailed copies of past revisions. These steps should minimize the consequences of hardware failure or data loss.

Risk: Compatibility Errors
Projected Solution: The program will be tested on multiple different machines with different capabilities and operating systems. Using multiple tests will ensure that each system tested will be compatible. If certain operating systems do not support the program, different revised versions will be offered.

Risk: Logic or Coding Errors
Projected Solution: A primary programmer and a secondary programmer will be available to help identify and correct errors from different perspectives.

Risk: Gameplay Flow and Function
Projected Solution: Each team member will assist in the creative process of the game's creation and will test the game to ensure that each of its aspects function as intended. Testing will occur in multiple phases and everyone will participate in these tests.

Procurement Management:

All team members will provide the essential instruments and tools necessary to complete their respective tasks. The tasks should not be a persistent issue because most of the equipment needed for completion of the tasks will be either provided by the school system or can be obtained legally online. However, if something expensive must be purchased, then the cost will either be split accordingly or will be purchased by a single team member. That team member will also retain ownership of the item after the project has concluded.

Schedule Management:

Table 1: Requirements and Analysis Phases

Start Date	Date Deadline	Task	Person Responsible	Finish	Verify
9/24	**9/27**	**General Description –** Overview of game, abilities, constraints	**Richard, Patrick**		
10/1	**10/4**	**Game meets specifications** (8-bit title, 1980s style)	**Richard**		
10/4	**10/7**	**System Design** Lifecycle determined	**Richard**		
10/4	**10/7**	**Meeting** with "Expert"	**Richard, Chance**		
10/4	**10/7**	**Data Flow Diagram**	**Patrick, Chance**		
10/4	**10/14**	**Review Requirements** and ask questions	**Everyone**		
10/4	**10/14**	**Story Board of Game**	**Patrick**		
10/5	**11/4**	**Project Plan (including its Function)**	**Everyone**		
11/8	**11/8**	**Project Plan and Timeline Submission**	**Everyone**		
10/4	**11/25**	**Class** list	**Richard**		
10/4	**1/1**	**UML Class Diagram**	**Richard**		

Table 2: Documentation Phase

Start Date	Date Deadline	Task	Person Responsible	Finish	Verify
9/27	**9/30**	**Teammate** job descriptions	**Richard, Patrick**		
10/3	**10/27**	**Methods Chart**	**Richard**		
10/4	**10/28**	**Title page** with company logo	**Patrick, Chance**		
10/12	**11/10**	**Class Header Comments**	**Tharindu**		
10/18	**10/21**	**Games Publisher** [BitBlit Interactive (BBI)] is mentioned multiple time in do	**Chance, Tharindu**		
10/18	**10/21**	**Product Initiative** [Project Wayback] is mentioned multiple time in docs	**Chance, Tharindu**		
11/1	**11/10**	**Software and Hardware requirements listed**	**Patrick, Tharindu**		
12/6	**12/27**	**Database design**	**Richard, Chance**		
12/7	**12/22**	**Back-end tables, fields designed**	**Richard**		
12/10	**12/23**	**PowerPoint created**	**Chance,**		
12/10	**12/23**	**User Manual created**	**Tharindu**		
12/14	**12/22**	**Front-end tables, fields designed**	**Richard**		
12/21	**12/31**	**I/O Interfaces –** examples of input or output, screen shots	**Richard**		
1/1	**2/22**	**Notebook finished** with CD, Title page, Team documentation, and 5 tabs (PPt, Specifications, User manual, Source code, JavaDocs)	**Everyone**		
1/1	**2/22**	**Comments** verified for correctness	**Tharindu**		
2/1	**2/22**	**JavaDocs and/or docs created**	**Tharindu**		
2/7	**2/24**	**User Manual** copied to Adobe pdf	**Chance**		
2/15	**2/22**	**All software copied to CDs**	**Everyone**		
2/15	**2/22**	**Adobe docs created**	**Tharindu**		

Table 3: Design and Implementation Phases

Start Date	Date Deadline	Task	Person Responsible	Finish	Verify
10/3	**10/6**	**Controls and Basic Logic**	**Richard**		
10/12	**11/3**	**Create all class and Method numbs (header structures)**	**Tharindu**		
10/12	**11/10**	**Create all Class and Method Header Comments**	**Tharindu**		
10/15	**11/5**	**¼ of game complete**	**Richard**		
10/18	**10/26**	**Platform-Wall.txt and logic**	**Richard**		
11/5	**11/30**	**½ of game complete**	**Richard**		
11/30	**12/21**	**¾ of game complete**	**Richard**		
12/6	**12/9**	**Player Login** (username/password authentication)	**Richard**		
12/6	**12/9**	**Game state** retrieved/sent from/to a database for storage	**Richard**		
12/13	**12/15**	**Database** (may be local or remote)	**Chance, Richard**		
12/13	**12/15**	**All player and game states are stored in a database**	**Chance, Richard**		
12/19	**12/21**	**Client implements a Leader board** (listing players and scores in a database)	**Richard**		
12/21	**1/1**	**All of game complete**	**Richard**		
12/27	**1/1**	**All of basic game code complete**	**Richard**		
1/3	**1/13**	**Readme.txt** doc created	**Richard**		

1/4	**1/31**	**Code Working** – lock all work on the code	**Richard**		

Table 4: Testing and Maintenance Phases

Start Date	Date Deadline	Task	Person Responsible	Finish	Verify
11/22	**1/1**	**Unit Level Testing** (use another sheet to track)	**Richard**		
12/6	**2/7**	**Name methods and state the tests to be run (including sample data)**	**Richard**		
12/20	**1/1**	**White Box Testing** (use another sheet to track)	**Chance, Patrick**		
12/20	**2/7**	**Black Box Testing** (use another sheet to track)	**Everyone**		
12/21	**1/1**	**Game supports multiple players** (may/not simultaneously)	**Richard, Chance**		
12/21	**2/7**	**Functional Testing** (use another sheet to track)	**Everyone**		
12/21	**1/1**	**Game state and Player data stored in a database**	**Richard**		
1/1	**2/1**	**Project works correctly in NetBeans** (and version is noted in documentation)	**Richard, Tharindu**		
1/17	**1/27**	**PowerPoint** verified that it is updated to match the finished project	**Chance**		
1/17	**1/27**	**User Manual** verified that it is updated to match the finished project	**Chance**		
2/1	**2/7**	**All Hardware** checked and working	**Everyone**		
2/1	**2/7**	**All CDs** checked and working correctly	**Tharindu**		
2/1	**2/7**	**All documentation** checked and ready to upload	**Tharindu**		
2/1	**3/2**	**Beta tested** by _____, _____	**Everyone**		
4/21	**4/21**	**Presentation practice final** (1st run through, timing run, final)	**Everyone**		

Citations:

Bouge, R.. N.p.. Web. 26 September 2013.
<http://zimmer.csufresno.edu/~sasanr/Teaching-Material/SAD/breaking down software development roles.pdf>.

James Jo, . N.p.. Web. 4 Nov 2013.
<http://www.techknol.net/2013/04/software-development-life-cycle.html>.

Perks, M.. N.p.. Web. 4 Nov 2013.
<http://www.ibm.com/developerworks/websphere/library/techarticles/0306_perks/perks2.html>.