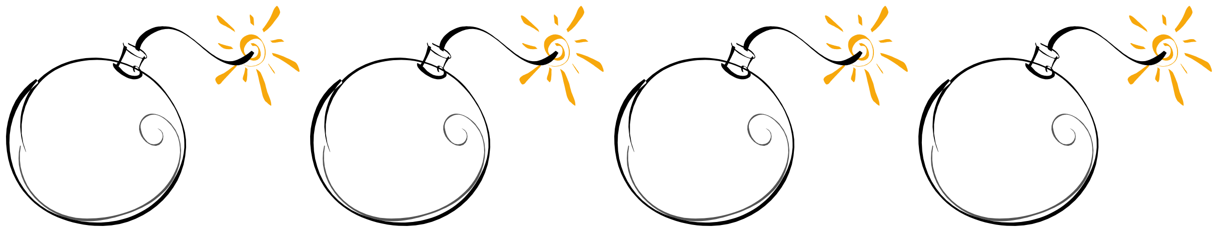
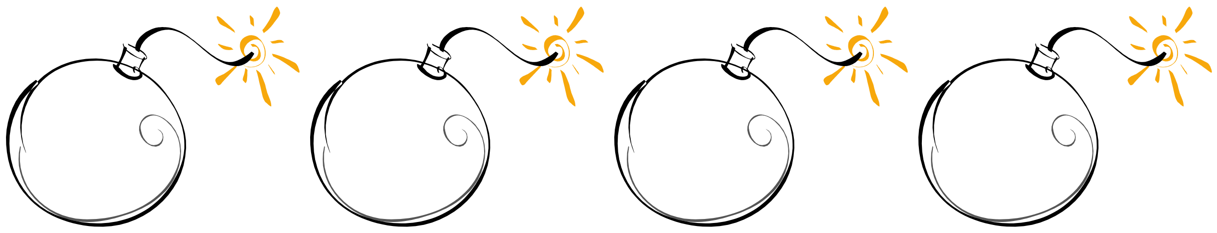


SOFTWARE REQUIREMENTS SPECIFICATION

Project Bomberman: Reloaded



FEBRUARY 9, 2014

TEAM RETRO REVOLUTION

Written by:

Hang Kang, Simon Krafft, Qianyu Liu, Alistair Russell, Sean Ryan, Wayne Tam

Table of Contents

1. Introduction

1.1 Purpose

The objective of this Software Requirement Specification (referred to as SRS thereafter) document is to help guide the complete design and development of a gaming system (referred to as the system thereafter). The SRS will be based on the client's written description of the desired product. Overall, this document will enumerate the features, core components and constraints to create an action, maze-based game similar to those of the original *Bomberman* franchise developed by Hudson Soft first published in 1985.

1.2 Intended audience

This document is intended for all members the software development team as a reference to the system's requirements and specifications. This document is also created for client to have a clearer perception of the gaming system and its functions. The Specific Requirements section constitutes the vital components that will be implemented in the Bomberman game.

1.3 Project Scope

The focus of this project is to develop a maze-based game similar to the original *Bomberman* video game franchise. The game will user to control a character and unlock levels. The game is designed to increase difficulty over time and will allow user to save and load game at each level. The game should support single player mode as well as two-player mode with alternating turns. The main objectives are to design a game software that functions and runs smoothly while implementing all the fundamental features mentioned on the document. The global scope of the project is to learn to collaborate on a major project with different people.

1.4 Definitions, Acronyms & Abbreviations

Client:	refers to the person who requests the initiation of the project and decide the key necessary requirements of the software gaming system.
Game:	refers to *Bomberman: Reloaded*, the current game in development. The game is also sometimes referred as the application.
User:	refers to the person(s) who uses the software system and provide input when needed. The user will also sometimes be referred as the player in the context of the actual game.
Bomberman:	refers to the main character which is controlled by the player
A.I.:	refers to the enemy or enemies in the game of which the player must destroy in order to pass the game.
Bomb:	refers to the weapon the character plants on the screen which as the ability to destroy the Bomberman itself, A.I.s, blocks and other objects.
Power-up(s):	refers to obtainable item(s) in the game for the player to provide additional gameplay advantages.
Square:	refers to the unit of measurement for length in this game. For example, 1x1 square is the size of Bomberman character.
BGM:	background music, also referred as music
SFX:	special sounds effects
GUI:	Graphical User Interface, refers to the user interface that allow the user to interact with the system.
JFC:	Java Foundation Class, refers to the graphical framework for building portable Java-based graphical user interfaces
API:	Application Programming Interface, specifies how software components should interact with each other

1.5 References

The link below and its associated sites are studied to have a better understanding of the rules of the game, properties of power-ups and the behavior of the Bomberman and his enemies.

“Bomberman.” Internet: http://strategywiki.org/wiki/Bomberman, Nov. 17, 2013 [Feb. 2, 2014]

1.6 Overview

The rest of this SRS document is consisted of two sections: overall description and specific requirements. The overall description (Section 2.x) will give general product perspective, functions and use cases, user characteristics, constraints and assumptions of the project. This section is primarily written for the client and other non-developers to have a basic knowledge of the development phase. The overall description is also used as a reference for the specific requirement.

On the other hand, the specific requirements (Section 3.x) is mainly written for the development team to guide through the completion of the design and implementation phase. This section contains details of functional and non-functional requirements, organized by categories. The document will also include any technical information about the graphics, external interface requirements, and other necessary requirements. The contents of this section is based on the client’s description of the desired final product. All specific requirements will be numbered to facilitate further references.

2. Overall Description

2.1 Product Perspective

The project will be a remake of the *Bomberman* game from the original series of the same name. The design and appearance of the game will have added personal touches from the development team. However, the overall emphasis is to design a Bomberman game with all the core functions and features notable in the original Bomberman games.

2.2 Product Features and Functions

2.2.1 Menus

The game will have a main menu and an in-game menu. The main menu is an interface first presented when the user open the game where the user can start the game. The main menu contains features including but not limited to: start new game, load game, select levels and options. The in-game menu is an interface presented only when the user press pause during the game. The in-game menu allows user to change some of the options, such as volume of background music and SFX. The in-game menu will also display features such as score, time and lives remaining.

2.2.2 Game Play

Game play refers to the overall functions and features of the game which will allow the user to control the Bomberman's movement, go through obstacles and A.I.s, and to proceed to the next level. The player will be able to place bomb, destroy blocks and defeat A.I.s to gain points. The player will also have the opportunity to pick up power-ups that gives player additional gameplay advantages. The user's score, time, and lives remaining will be displayed on the screen as well as the in-game menu when the game is paused.

2.2.3 Options

The user will be allowed to change settings through the main menu or the in-game menu. The option function will allow the user to adjust features such as background music volume and SFX volume. Controls and properties of all power-ups will also be described in the option menu as a guide, but not available for the user to change.

2.2.4 Exit

This function will exit the game.

2.3 Use Cases

The below use cases define the interaction between the user (player in the context of the game) and the Bomberman game system. All use cases are essential to the development unless otherwise indicated.

2.3.1 Use Case: Main Menu Navigation

The main menu is presented when the user launches the application or when the user selects “Return” in any of the submenus or “Quit” the in-game menu. All use cases under main menu navigation is initiated by the user, while the submenu differs corresponding to the option the user selects. The user can choose from six options in the main menu: “New Game”, “View High Score”, “Load Game”, “Select Levels”, “Options” and “Credits”.

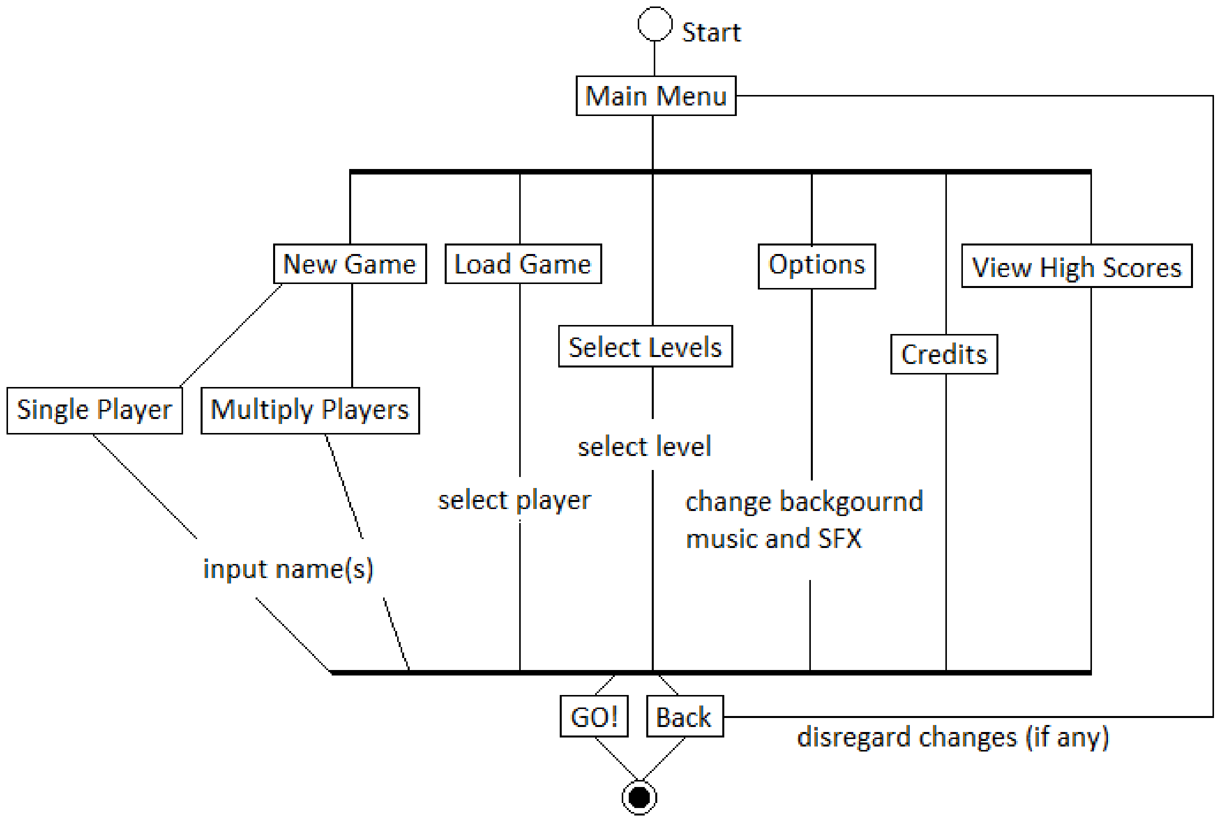


Figure 1: Use Case Main Menu

2.3.1.1 New Game

Use Case Name:	**Navigating New Game**
Participating Actor(s):	User
Entry Conditions:	User select "New Game" from the main menu
Flow of Events:	1. System displays the "New Game" menu with options to choose between single player mode or multiplayer mode 1.1. If user selects single player mode 1.1.1. Application prompts user to enter player name 1.1.2. User enters character name 1.2. Else if user selects multiplayer mode 1.2.1. Application prompts user to enter two player names 1.2.2. User enters two characters names 2. If user selects "Go!" 2.1. Application runs level from level 1 2.2. Bomberman character is displayed on the top left corner of screen 3. Else if user selects "Back"

	3.1. Application disregards all input 3.2. Application returns to the main menu
Exit Condition	1. User selects "Go!" (see 2.3.3 Play Game), or 2. User selects "Back"
Feature Requirements:	The default player name should be "Player 1" and "Player 2" Player name(s) can be only in letters (case sensitive, upper and lower case allowed) and/or numbers. The system will not accept space and other symbols as player name (and will asks for user's input again). Player name(s) should be at least 1 character and at most 10 characters. Player 1 name must be different from player 2 name (in multiplayer mode)

2.3.1.2 View High Score

Use Case Name:	**Navigating View High Score**
Participating Actor(s):	User
Entry Conditions:	User select "View High Score" from the main menu
Flow of Events:	1. System displays the "View High Score" menu 1.1. Application displays a top 10 list of player names and their corresponding scores in descending order from top to bottom 2. User view high score 3. User selects "Back" and returns to the main menu
Exit Condition	User returns to the main menu
Quality Requirements:	The system should automatically update and sort the list of scores after every game.
Exception:	The "View High Score" menu should be disabled when there is no score stored in the game system.

2.3.1.3 Load Game (desired implementation)

Use Case Name:	**Navigating Load Game**
Participating Actor(s):	User
Entry Conditions:	User select "Load Game" from the main menu
Flow of Events:	1. System displays the "Load Game" menu with options to choose saved game 1.1. Application prompts user to select a player 2. User selects player and "Go!"

	2.1. Application runs and continues the game from the last point the game is saved 3. User selects "Back" 3.1. Application disregards all selection (if any) 3.2. Application returns to the main menu
Exit Condition	1. User selects "Go!" (see 2.3.3 Play Game), or 2. User selects "Back"
Feature Requirements:	Application can save at most 2 game at any given time. Otherwise the user must replace the new game with the existing one. The "Load Game" menu should be disabled when there is no game stored in the system.

2.3.1.3 Select Levels (desired implementation)

Use Case Name:	**Navigating Select Levels**
Participating Actor(s):	User
Entry Conditions:	User select "Select Level" from the main menu
Flow of Events:	1. System displays the "Select Level" menu with options to choose levels to continue the game 1.1. Application prompts user to select a level 1.2. User selects level 2. If user selects "Go!" 2.1. application runs selected level 3. Else if user selects "Back" 3.1. Application returns to the main menu
Exit Condition	1. User selects "Go!" (see 2.3.3 Play Game), or 2. User selects "Back"
Feature Requirements:	Application should have 10 levels with increasing difficulty Each "level" should be enable only when that level has already been unlocked (by any player)

1.3.1.4 Options (Music and SFX)

Use Case Name:	**Navigating Options**
Participating Actor(s):	User
Entry Conditions:	User select "Options" from the main menu or the in-game menu
Flow of Events:	1. System displays the "Option" menu with options to change background music volume, SFX volume. The controls and explanation of all power-ups will also be shown here. 1.1. User adjust background music volume,

	1.2. User adjust SFX volume, or 1.3. User view controls and power-ups 2. User selects "Back" 2.1. Application save changes (if any) 2.2. If user accesses "Options" in main menu 2.2.1. User returns to the main menu 2.3. Else if user accesses "Options" in in-game menu 2.3.1. User returns to the in-game menu
Exit Condition:	User returns to the main menu or the in-game menu (depending on the access point)
Feature Requirements:	System should save the setting changes for the remainder of the application session. Option interface should be clean and easy to navigate: user should be able to complete setting change within 30 seconds 9 out of 10 tries.

2.3.1.5 Credits (desired implementation)

Use Case Name:	**Navigating Credits**
Participating Actor(s):	User
Entry Conditions:	User select "Credits" from the main menu or when player finishes the entire game (unlock all the levels)
Flow of Events:	1. System displays list of names of participants and their role in the development process including, but limited to, the client and the members of the development team. 2. Credits ends and automatically returns to the main menu, or 3. User selects "Back" and returns to the main menu
Exit Condition:	User returns to the main menu

2.3.2 Use Case: In-Game Menu Navigation

The in-game menu is presented when the user pauses the game. When paused, all features (time, A.I.s) will stop moving, and the user will not be able to control the Bomberman. All use cases under in-game menu navigation is initiated by the user, while the submenu differs corresponding to the option the user selects. The user can choose from three options in the in-game menu: "Save Game", "Resume" and "Options".

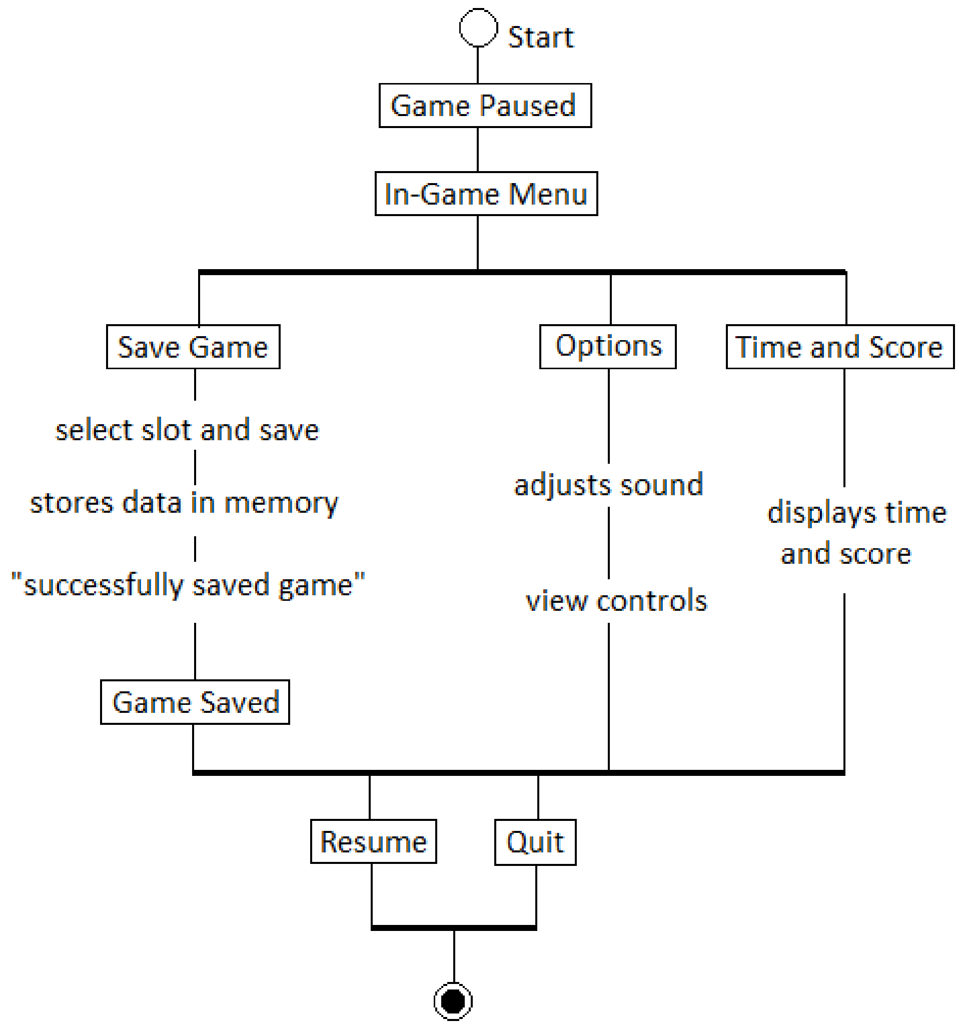


Figure 2: Use Case In-Game Menu

1.3.2.1 Save Game (optional implementation)

Use Case Name:	**Navigating Save Game**
Participating Actor(s):	User
Entry Conditions:	User select "Save Game" from the in-game menu
Flow of Events:	1. Application prompts user to select a slot to save game 2. User selects slot and click "Save", or 2.1. Application stores game with score in memory 2.2. Application verifies "Successfully Saved Game" 2.3. User returns to in-game menu after save game completed 3. User selects "Back" 3.1. Application disregards any changes 3.2. User returns to the in-game menu

Exit Condition	User returns to the in-game menu
Exceptions:	When application already has 2 game stored, the application will asks whether user wants to replace the new game with the existing one. If "Yes" – application saves game and returns to the in-game menu If "No" – application returns to the in-game menu without saving The system only allows save game when it is in single player mode

1.3.2.2 Resume

Use Case Name:	**Selecting Resume**
Participating Actor(s):	User
Entry Conditions:	User select "Resume" from the main menu
Flow of Events:	1. Application exits in-game menu and returns to the game interface 2. Application display a count-down from 3 (during this time the game should still be paused) 3. When count-down finishes, game continues
Exit Condition	User exits in-game menu and plays game (see 2.3.3 Play Game)

2.3.2.4 Options (Music and SFX)

See 2.3.1.4 (Navigating Options)

2.3.3 Use Case: Play Game

This section describe the flow of events when the user plays the game. All use cases under play name is initiated by the user and based on user's input.

User has the choice of choosing Single Player mode and Multiplayer mode. Single player mode requires user to input one player name (default: Player 1), and allows user to save game (and thereby load game). The system will also automatically saves user's high score when the game finishes and display the player's name and corresponding score in the "View High Score" submenu of the main menu (if the score is within the top 10. See 2.3.1.2 View High Score). The single player mode is story-based, and will follow the similar storyline to that of the original *Bomberman* game with introductory clips, cut scenes and ending clips.

On the other hand, multiplayer mode requires users to input two player names (default: Player 1 and Player 2). Two players will play in alternating turns. Multiplayer mode will contains less features and functions in comparison to single player mode. Multiplayer mode will not be story-based. In addition, the system will not allow users to save game (and thus the user will also not able to load multiplayer game). The system will also not save any scores in memory.

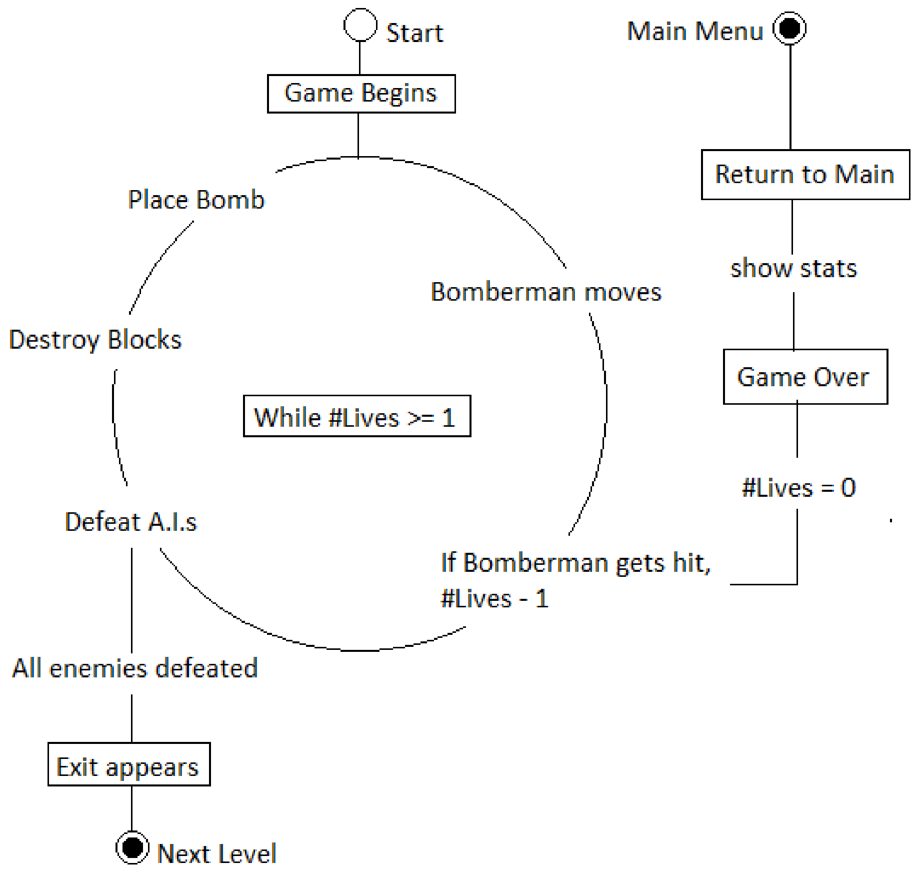


Figure 3: Use Case Play Game (Single Player)

2.3.3.1 Single Player Mode

Use Case Name:	**Play Game – Single Player**
Participating Actor(s):	User
Entry Conditions:	1. User selects "Go!" (from "New Game", "Load Game", or "Select Levels"), or 2. User selects "Resume" in in-game menu
Flow of Events:	1. System starts (or continue in case of loading and resuming) 1.1. User input is enabled. User can use command keys to control the Bomberman character 1.2. All A.I. starts moving 2. While player's number of lives is greater than zero, user control Bomberman character and player plays game by destroying blocks and defeating A.I.s 2.1. If A.I.s or blocks get hit by bomb 2.1.1.1. The A.I. or block disappear from the screen 2.1.1.2. Points are added to the score 2.2. If player gets hit by bomb or A.I.s 2.2.1.1. Number of lives deducts 1 2.2.1.2. Player is teleported back to the top left corner 2.2.1.3. Player is free from damage and placing bomb for 2 seconds. 3. If user completes level (after defeating all A.I.s and exits level) 3.1. If user completes the last level, or 3.1.1. System displays "You Won!" message 3.1.2. System displays player's score 3.1.3. System saves score to memory 3.1.4. User selects "Back to Main" and returns to the main menu 3.2. Player proceeds to the next level 3.2.1. Points are added to the score 3.2.2. New level unlocks (user can now load from this level) 3.2.3. New level begins (loop back to 1) 4. If number of lives equals zero 4.1. System displays "Game Over" message 4.2. System displays player's score 4.3. System saves score to memory where it will be updated in

	the "View High Score" menu (if the score is in the top 10) 4.4. User selects "Back to Main" to return to the main menu (see 2.3.1 main menu) 5. Else if user pauses the game 5.1. User exits game play interface 5.2. User enters in-game menu interface
Exit Condition:	1. User selects "Back to Main" after losing to completing game (see 2.3.1 Main Menu), or 2. User pauses the game and enters in-game menu interface (see 2.3.2 in-game menu)

2.3.3.2 Multiplayer Mode

Use Case Name:	**Play Game – Multiplayer**
Participating Actor(s):	Two users: Player 1 and Player 2
Entry Conditions:	1. Either User selects "Multiplayer Mode" and "Go!" from "New Game", or 2. User selects "Resume" in in-game menu
Flow of Events:	1. While both players have more than one life 1.1. Player 1 plays one level (see 2.3.3 Play Game – Single Player) 1.2. Player 2 plays the same level (see 2.3.3 Play Game – Single Player) 2. If a player has no remaining lives 2.1. The system presents "<player name> Game Over" message 2.2. The other player continues game 3. If both player have no remaining lives 3.1. System presents "<player name> Game Over" message 3.2. System display to player names and their corresponding scores as well as a combine score of both players 3.3. User selects "Back to Main" to returns to the main menu
Exit Condition	1. User selects "Back to Main" after losing to completing game (see 2.3.1 Main Menu), or 2. User pauses the game and enters in-game menu interface (see 2.3.2 in-game menu)
Exceptions:	1. System does not save game (and score) in multiplayer mode. 2. When one player has no remaining lives, system switches to the other player instead of giving the option to "Back to Main"

2.4 User Characteristics:

The game should be playable by any users at any technical skill level, regardless of age and experience. Instructions, controls and properties of power-ups will be described in the game. However, the user is expected to know the fundamental rules of the game beforehand. The game should not require any previous acquired knowledge about the game functions to be able to play the game. Other than simple controls for moving, placing bombs and pause game, the user should not expect to learn any other sets of commands.

2.5 Constraints

2.5.1 Memory Constraints

Since this game will be of small scope compared to most games on the market today, memory usage should not be a critical constraint. The current estimate is that the game should require less than 100 MB hard drive memory. Memory allocation and optimization will be used on a needed basis and implemented more if there is extra time after the game has passed its testing phase successfully.

2.5.2 Design and Implementation Constraints

This software is designed to be playable by users of all ages and skill levels. However, the user is expected to know some of the fundamental rules of the game. Users should not have access to the source code of the product. The bulk of the code will be written in java with the main IDE being Eclipse. The main GUI will be implemented through the Java Swing library import. Java Swing will be implemented through the JFC which will be the API for providing the GUI to the developers. The McGill web server will be utilized to publish the game to facilitate the availability to users. All corrections and addition of code must be posted to GitHub for transparency to all team members.

2.5.3 Time Constraints

The *Bomberman: Reloaded* is expected to be completed by late March. The game is expected to be released in its entirety to the client by the early April. The official release of the game will be depend on the decision of the client.

2.6 Assumptions and Dependencies:

- This product shall require an operating system that can recognize and run java code.
- This product shall require the utilization of java libraries imported through Eclipse.
- The graphics quality will be dependent on the quality and implementation of code.

3. Specific Requirements

3.1 Detailed Functional Requirements by Category

3.1. CT Controls

3.1. CT.1 Arrow keys (essential; not yet implemented; medium)

- Bomberman character's movements are only controlled by arrow keys.
- Bomberman character moves in the direction of the arrow key pressed.
- When the player presses and holds an arrow key, the character will move in the corresponding direction until the key is released.
- When another arrow key is pressed while an arrow key is pressed, the character will stop moving in the direction of the first arrow key, and move in the direction of the second arrow key.

3.1. CT.2 Space Key (essential; not yet implemented; medium)

- When space key is pressed or held, the character will place one bomb.

3.1. CT.3 ESC Key (essential; not yet implemented; medium)

- When ESC key is pressed or held, the game will pause. See 3.1. CT.5 Pause for specifications for pausing the game.

3.1. CT.4 Mouse (essential, not yet implemented, medium)

- User must navigate through the menu using mouse.
- User selects options by left-clicking the option.
- Right-click will have no effect to the game.

3.1. MN Menus

Menu requirements contain any functional requirement that is related to the main menu and the in-game menu. Both main menu and in-game menu have submenus. Menu provides a platform for the user to interact with the game. The user will have to navigate through menus and submenus in order perform functions such as play game and change settings.

3.1. MN.1 Menu Graphics (essential; not yet implemented, medium)

- Menus (main and in-game menus) should be an interface with hyperlinks that connects to various submenus.

- Submenus should extend menus in term of graphics.

3.1. MN.2 Main Menu (essential; not yet implemented, medium)

- The main menu appears after the introductory clip or whenever the user selects options that leads to the main menu.
- The main menu contains the options that link to submenus. See 3.1. MN.4 – 3.1. MN.9 for the specifications of main menu submenus.

3.1. MN.3 In-game Menu (essential; not yet implemented, medium)

- The in-game menu appears when the user pauses the game.
- The in-game menu contains the options that link to submenus. See 3.1. MN.10 – 3.1. MN.11 for the specifications of in-game menu submenus.
- To exit in-game menu, user must selects "Resume". See 3.1. GP.2 for specification of pausing the game.

3.1. MN.4 New Game (essential; not yet implemented, medium)

- New game contains options for user to choose either single player mode or multiplayer mode.
- After user selects mode, the user must also input character name(s). See 3.1 GP.1 for the specification of name input.

3.1. MN.5 View High Score (essential, not yet implemented, difficult)

- View High Score displays the top 10 player names and their corresponding scores
- High Score should automatically updates its content after every game in single player mode

3.1. MN.6 Load Game (desirable, not yet implemented, difficult)

- The submenu allows the user to retrieve the history of game playing and start at the same place as the user had left the game.
- The "Load Game" menu should be disabled when there is no game stored in the system.

3.1. MN.7 Select Levels (desirable, not yet implemented, difficult)

- The submenu allows the player to select a level
- Each "level" should be accessible only when that level has already been unlocked (by any player)

3.1. MN.8 Options (desirable, not yet implemented, medium)

- The submenu that allows the player to adjust the volume of background music and SFX. See 3.1. AV for specifications of sounds and audio.
- System must save any changes made and implement the changes for the duration of the game play
- The submenu must display controls and descriptions of each power-ups in image format

3.1. MN.9 Credits (desirable; not yet implemented, easy)

- The submenu displays the list of names of participant in this project.
- The format of presentation is scrolling texts

3.1. MN.10 Save Game (desirable; not yet implemented, difficult)

- Save Game allows player to store his current state in the system.
- The system can save at most 2 game at any given time. Otherwise the user must replace the new game with the existing one.

3.1. MN.11 In-Game Options (desirable, not yet implemented, medium)

See 3.1. MN.8 Options for specifications of options

3.1. MN.12 Back (essential, not yet implemented, medium)

- All submenus of main menu must have the option “Back” to return to the main menu
- All submenus of in-game menu must have the option “Back” to return to the in-game menu
- In-game menu must have the option “Quit” to return to the main menu
- In-game menu must have the option “Resume” to return to the game. See 3.1. GP.3 for specification for resuming game

3.1. MN.13 Go (essential, not yet implemented, medium)

- “New Game” of main menu must have the option “Go!” to start new game, which should only be enabled when player inputs name for Bomberman character
- “Select Level” of main menu must have the option “Go!” to start new game, which should only be enabled when a level is selected.
- “Load Game” of main menu must have the option “Go!” to start new game, which should only be enabled when a game is selected.

3.1. MN.14 Sound (essential, not implemented, difficult)

See 3.1. AV for specifications of background music and SFX.

3.1. MN.15 Three Seconds Countdown (optional; no yet implemented; medium)

- When user selects “Resume”, application should pause game for 3 additional seconds before resume playing

3.1. BM Bomberman (Main Character)

3.1. BM.1 Graphics (essential; not yet implemented, easy)

- Bomberman image is based on the original *Bomberman* game.
- The size of image is a 1x1 square

3.1. BM.2 Actions (essential, not yet implemented, medium)

- Bomberman’s movements are controlled by the user based on user’s input. See 3.1. CT Controls for specification of command keys.
- Arrow keys will control Bomberman’s movements
- The default speed of movement is 1 square per second
- Bomberman cannot move through blocks.
- Bomberman cannot move out of the map
- Space key will allow Bomberman to place a bomb

3.1. BM.3 Life (essential; not yet implemented, medium)

- Bomberman will have 3 lives initially (when user starts a new game)
- Bomberman’s number of lives is displayed on the screen during the game play.
- Number of lives decreases when Bomberman is hit. See 3.1. GP.4 Hit Detection – Bomberman for specification of when Bomberman is hit.
- At the start of every level, the number of lives is reset to 3.

3.1. BM.4 Bomb (essential; not yet implemented; difficult)

- There is no limit to how many bombs the Bomberman can place.
- The Bomberman can place the second bomb immediately after the first bomb explodes. See 3.1. PU Power-Ups for specification of power-up that can increase the number of bombs the Bomberman can place at a given time.
- Bomb produces a cross-shaped explosion. See 3.1 AV.4 for specification of bomb explosion visual effect
- The default range of an explosion is 3x3 block-size

3.1. AI Artificial Intelligence (Enemy)

3.1. AI.1 Graphics (essential, not yet implemented, easy)

- Character image is based on the original Bomberman game.
- The size of image is a 1x1 square
- A.I.s images vary for different types of A.I.s

3.1. AI.2 Movement (essential, not yet implemented, difficult)

- A.I.s moves by itself without user input
- A.I.s cannot move through blocks.
- A.I.s cannot move out of the map

3.1. AI.3 Implementation (essential, not yet implemented, difficult)

- A.I.s will be randomly distributed across the first, third and fourth quadrant of the map
- The type distribution is based on level. See 3.1. LV.3 Difficulty for specification on how A.I.s should be distributed.

3.1. AI.4 Types (essential, not yet implemented, medium)

- A.I. 1:
  - Speed is 0.75 square per second
  - When defeated, 100 points are added to score
- A.I. 1 (smart):
  - Speed is 0.75 square per second
  - When defeated, 250 points are added to score
- A.I. 2:
  - Speed is 1 square per second
  - When defeated, 250 points are added to score
- A.I. 2 (smart):
  - Speed is 1 square per second
  - When defeated, 500 points are added to score
- A.I. 3:
  - Speed is 1.50 square per second
  - When defeated, 500 points are added to score
- A.I. 3 (smart):
  - Speed is 1.50 square per second
  - When defeated, 1000 points are added to score

3.1. AI.5 Intelligence (desirable, not yet implemented, difficult)

- Selected enemy (indicated in 3.1. AI. 4 Types) is smart.
- A smart A.I. has the ability to detect a bomb 1-square away.
- A smart A.I. has the ability to detect the player from 2-squares away and moves towards the player.

3.1. BL Blocks

3.1. BL.1 Graphics (essential, not yet implemented, easy)

- Block image is based on the original Bomberman game.
- The size of image is a 1x1 square

3.1. BL.2 Theme (optional, not yet implemented, medium)

- Images changes in each levels. See 3.1. LV.1 Graphics for specifications of level themes.

3.1. BL.3 Properties (essential, not yet implemented, medium)

- Block does not move for the duration of the game unless it is destroyed.
- The player cannot pass though blocks

3.1. BL.4 Implementation (essential, not yet implemented, difficult)

- Blocks will be distributed in patterns by design.
- From level 1 to level 5, destructible blocks will account for 100% of the all blocks.
- From level 5 and up, destructible blocks will account for 90% of all blocks, while indestructible blocks will account for 10% of all blocks.

3.1. BL.5 Types (optional, not yet implemented, medium)

- Destructible blocks: can be destroyed by bomb. See 3.1. GP Hit Detection – Blocks for specification of destroying blocks. 90% probability of appearing.
- Indestructible blocks: cannot be destroyed by bomb. 10% probability of appearing.

3.1. LV Levels and Maps

3.1. LV.1 Graphics (essential, not yet implemented, easy)

- Each level is a 15x15 squares gridded map.
- All levels contain A.I.s, blocks and power-ups. See 3.1 AI.3 and 3.1 BL.3, and 3.1. PU.2 Implementation for specification of constructing level map.

3.1. LV.2 Theme (optional, not yet implemented, medium)

- Each level has a different theme, distinguished by color of blocks (desirable).
- Color for each level is chose randomly.

3.1. LV.3 Difficulty (essential, not yet implemented, medium)

- There should be a total of 10 levels.
- Each level is comparatively more difficult than the previous. Primary method of measuring difficulty is by the type and number of A.I.s in each level.

3.1. LV.4 Exit (essential, not yet implemented, medium)

- Each level has an exit, where the player must pass through to proceed to the next level.
- Exit only appear when all the A.I.s have been defeated.

3.1. LV.5 Score and Time (essential, not yet implemented, medium)

- The game must display current time and score on screen when playing game.
- Score gained from one level is carried and added to the next level.
- When player loses a single player game, system must save the total score in memory.
- When player completes a single player game, system must save the total score in memory.
- System should organize all stored scores in a sorted manner (descending order) with their corresponding player names.
- System should not store scores from a multiplayer game.
- User must be able to see the list of top scores. See 3.1. MN.5 View High Score for specification of viewing high score.

3.1. GP Game Play

3.1. GP.1 Player Name Input (essential, not yet implemented, medium)

- The default player name should be “Player 1” and “Player 2”
- Player name(s) can be only in letters (case sensitive, upper and lower case allowed) and/or numbers. The system will not accept space and other symbols as player name (and will asks for user’s input again).
- Player name(s) should be at least 1 character and at most 10 characters.
- Player 1 name must be different from player 2 name (in multiplayer mode)

3.1. GP.2 Game Begins (essential, not yet implemented, medium)

- When starting a new game or level, the Bomberman character is set to the top left corner. See 3.1. MN.13 Go for specification of starting game.

3.1. GP.3 Pause (essential, not yet implemented, medium)

- When game is paused, all objects (including the player and A.I.s) must stop moving
- When game is paused, application stops timing

3.1. GP.4 Resume (essential, not yet implemented, medium)

- When game is resumed, system starts countdown. See 3.1. AV.4 for specification of system countdown.
- When countdown reaches zero all objects (including the player and A.I.s) start moving

3.1. GP.5 Hit Detection – Bomberman (essential, not yet implemented, difficult)

- If the Bomberman is is touched by an A.I. or when it is within the range of bomb explosion, the Bomberman is considered "hit".
- When the Bomberman is hit, its number of lives is deducted by 1
- When the Bomberman is hit, it is teleported to the top left corner of the map.
- Bomberman is free from damage for 2 seconds after it is hit.

3.1. GP.6 Hit Detection – A.I. (essential, not yet implemented, difficult)

- If an A.I. is within the range of bomb explosion, it is considered "defeated" by the Bomberman.
- When an A.I. is defeated, it is removed from the screen.
- Points are added to the score.

3.1. GP.7 Hit Detection – Blocks (essential, not yet implemented, difficult)

- If a block is within the range of bomb explosion, it is considered "hit".
- When a block is hit, it is removed from the screen
- Points are added to the score

3.1. GP.8 Consume Power-ups (essential, not yet implemented, difficult)

- If the Bomberman is at the same position of the power-up, it is considered that the player "consumed" the power-up
- When the Bomberman consumes a power-up, its corresponding properties changes. See 3.1. PU Power-Ups for specification of power-ups

- All power-up implementation is reset when player loses a life, with exception for bonus point power-ups.
- All power-up implementation is reset when player completes a level, with exception for bonus point power-ups.

3.1. PU Power-Ups

3.1. PU.1 Image (essential, not yet implemented, easy)

- All power-ups images will be taken from the original Bomberman game.
- All power-up images will be 1x1 square.

3.1. PU.2 Implementation (essential, not yet implemented, difficult)

- There is at least 1 power-up and at most 10 power-ups per level.
- The type of power-ups will be distributed at random.
- Power-ups are randomly distributed in the beginning of the game.
- Power-ups will be hidden under where destructible blocks are positioned, randomly displayed when that block is destroyed.

3.1. PU.3 Types (essential, not yet implemented, difficult)

- Bomb Range: when consumed, range of bomb increased by 1 square. 10% probability of appearing.
- Bomb Number: when consumed, the number of bomb the Bomberman can place at a given time is increased by 1 unit. 10% probability of appearing.
- Speed: when consumed, the speed of Bomberman's movement increase 20%. 15% probability of appearing.
- Invincibility: when consumed, the Bomberman is free from damage for 15 seconds. 5% probability of appearing.
- Bonus 1: when consumed, add 100 points to score. 15% probability of appearing.
- Bonus 2: when consumed, add 250 points to score. 10% probability of appearing.
- Bonus 3: when consumed, add 500 points to score. 5% probability of appearing.
- Bonus 4: when consumed, add 1000 points to score. 1% probability of appearing.

3.1. AV Audio & Visual Effect

Background music and SFX will be present throughout the game. The main menu and the gameplay will feature retro style Bomberman themed music. Sound effect audio will be implemented during the gameplay for bomb explosions, block destructions, A.I. death, etc.

3.1. AV.1 Sound Controls (desirable, not yet implemented, medium)

- User is able to turn on and off background music in “Options” in main menu and in-game menu.
- User is able to turn on and off SFX in “Options” in main menu and in-game menu.

3.1. AV.2 Background Music (essential; not yet implemented, medium)

- Background music will be taken from the original Bomberman game with reference.
- System plays a continuous looping retro Bomberman game theme music when user is in main menu.
- System plays a continuous looping retro Bomberman game theme music when user is game play and in in-game menu.
- Background music differs by levels. When player advances to the next level, background music should change too.

3.1. AV.3 SFX (optional; not yet implemented; medium)

- SFX will be taken from the original Bomberman game with reference.
- Level start: when the user begins a level, the system plays a 1-second sound
- Bomb explosion: when a bomb explodes, the system plays a 1-second sound portraying the explosion.
- Block destruction: when blocks are destroyed, the system plays a 1-second sound depicting its destruction.
- Level completion: when the user passes a level, the system plays a 1-second sound
- Level failure: when player loses, the system plays a 1-second sound
- Menu selection: when the user selects an option in the menus or submenus, the system plays a 1-second sound.

3.1. AV.4 Visual Effects (desired, not yet implemented, difficult)

- Visual effects will be taken from the original *Bomberman* game.
- Power-ups: the system plays a 1-second animation indicating that the Bomberman has consumed a power-up.

- Bomb Explosion: the system plays a 1-second animation portraying the explosion of bomb
- Countdown: the system plays a 3-second animation portraying a countdown after player resume from the in-game menu

3.1. SL Storyline

Storyline requirements contains any functional requirement that is related to the function of the plot for *Bomberman: Reloaded*. Storyline includes, but limited to, introductory clips, cut scenes, and ending clips.

3.1. SL.1 Graphics (desirable, not yet implemented, easy)

- Clips will either be video clips taken from the original game with appropriate reference or story in the form of scrolling texts

3.2. SL.2 Introductory Clip (desirable, not yet implemented, easy)

- An introductory clip will play every time when user starts new game in single player mode.
- The introductory clip should be less than 30 seconds

3.2. SL.3 Ending Clip (desirable, not yet implemented, easy)

- An ending clip will play when user completes the last level in single player mode.
- The ending clip should be less than 30 seconds

3.2. SL.4 Cut scene (desirable, not yet implemented, easy)

- A cut scene clip will play every time when user completes a level in single player mode.
- The cut scene clips should be less than 5 seconds
- Exception: when user completes the last level, see 2.3. SL.2.2 Ending Clip

3.2. SL.5 Skip Clip (desirable, not yet implemented, medium)

- There must an function for user to skip the introductory and ending clips

3.2. SL.6 After Skip (desirable, not yet implemented, easy)

- When user skips introductory clip, application runs main menu
- When user skips ending clip, application runs credits

3.4 External Interface Requirements:

3.4.1 User Interfaces:

The game software requires the use of keyboard to play the game. Mouse is not required but recommend to navigate and interact with the menus. The standard controls will allow the user to load a game, save a game and play the game. The user also can bring up a menu for additional options such as adjusting the volume of music and SFX. The game will be organized through a main story line that will progress by different playable levels, each of different variety and increasing difficulty.

3.4.2 Hardware Interfaces:

There will be no specific hardware interface for this game as it should be able to run on nearly any operating system, the main ones being Windows, Mac, and Linux. If a certain operating system proves more effective at running and organizing the functions of the game it should be noted and evaluated for further improvement if necessary.

3.4.3 Software Interfaces:

The Java designed game will be capable of being run on any version of Windows, Mac or UNIX. More information will be added to this section if necessary when further details are revealed and more knowledge is gained by the developers.

3.5.4 Communications and Error-Handling Interfaces:

Main communication will be through pop-up windows displayed in the event of errors or other important messages. This game will not require internet connection.

3.6 Design Constraints

This software is designed to be enjoyable for all users.

See more in 2.5.1 Memory Constraints and 2.5.2 Design and Implementation Constraints

3.7 Other Requirements

3.7.1 Reliability

The system is only allowed to crash at most 5% of the time.

3.7.2 Availability

The game will be free and be able to run on any computer with Java installed.

3.7.3 Security and Privacy

The game should not request, collect nor exchange any user information from the user or from the computer on which it is installed.

3.7.4 Maintainability

Any fixes or updates should be released and implemented as a separate software release. Separate download and installation is required.

3.7.6 Other requirements

The software will be package with proper documentation in text file. The documentation includes, but limited to: terms and conditions, instruction on installation, basic rules of the game and a link to which the user can find more information.