SOFTWARE REQUIREMENT SPECIFICATION (SRS)

2048 GAME

Version 1.0

Syeda Sara Akif (CS-17002)

Mahrukh Khan (CS-17003)

Ayesha Nadeem Akhter (CS-17030)

Department of Computer and Information Systems
NED University of Engineering and Technology

Submitted to
Ms. Fakhra Aftab

Novemeber 22, 2019

Contents

Chapter 1

Introduction

The aim of the Software Requirements Specification (SRS) document is to gather and analyze essential aspects of the software as well as to give its in-depth insight. The first chapter of this document covers the project feasibility along with the preliminary idea about the software whereas the introduction of this chapter provides a brief overview of the entire document. In general, any software project is a project focusing on the creation of software along with its proper documentation and consequently the success of that project can be measured by the resulting software. As Gaming is the fast growing field of software engineering and development, "2048 game" has been chosen by us as our software project.

1.1 Document Purpose

This document is intended to give a comprehensive overview of our project '2048 game' by presenting a thorough description of all characteristics of the game. The document not only specifies the user interface but also the guidelines to interact with through it with the game, the requirements along with the constraints of how the software project must behave at different conditions or inputs. The SRS also details all the features upon which 2048 is designed with reference to the manner and importance of their implementation. This document is intended for all the stakeholders of the game.

1.2 Product Scope

The 2048 Game is an interactive computer game with a textual interface which entertains the user by helping the user to pass a little time without getting bored. 2048 is a single-player game where the player makes actions

using the four arrow keys of a keyboard on a 4 x 4 grid. Initially, the grid is empty except two to three tiles placed on the board randomly. As the player makes a move, the program spawns a tile on the grid randomly while all tiles slide towards the edge of the chosen side if able. If two tiles with the same numerical value collide, they merge into one tile having twice the merged value. The player wins if a tile with value 2048 is spawned, and loses if no more moves are available i.e the grid is filled completely.

1. **State-based Definition** : Player input: actions (up, down, left and right) based on the current state Computer input: adds tiles on the grid randomly based on the current state Output: result state, score (instant) while playing the game and a final list of player scores at the end of the game. Terminal state (success): One '2048' tile Terminal state (failure): All adjacent tiles occupied and have different values, i.e. no more moves available

2. **Evaluation Metrics**: The main goal of this project is to get a '2048' tile. However, that is not the only evaluation metric in our project. Besides achieving a 2048 tile in the grid, there is also a bar displaying player score that keeps adding up whenever the player matches two tiles into one. The winning of the game is a 2048 tile but the score value serves as the icing on the cake.

1.3 Intended Audience and Document Overview

The SRS document also gives testers and project managers a way to ensure the game's adherence to original vision. Although the document may be read from front to back for a complete understanding of the project, it was written in sections and hence can be read as such. For an overview or general description of the document and the project itself, see Overall Description **(Section 2)**. For a detailed description of the game features, specific requirements that the system is expected to deliver and their interaction with the user, see Specific Requirements **(Section 3)**. This section is written primarily for the developers and describes details in technical terms. The technical standards according to which the game will be developed are laid in Other Non-Functional Requirements **(Section 4)**.

Both sections of the document describe the same software product in its entirety, but are intended for different audiences and thus use different language

1.4 Definitions, Acronyms and Abbreviations

- **PUZZLE:** A puzzle is a game, problem, or toy that tests a person's ingenuity or knowledge. In a puzzle, the solver is expected to put pieces together in a logical way, in order to arrive at the correct or fun solution of the puzzle.
- **USE-CASE:** In software and systems engineering, a use case is a list of actions or event steps typically defining the interactions between an actor and a system to achieve a goal. The actor can be a human or other external system.
- **DFD:**A data-flow diagram is a way of representing a flow of a data of a process or a system. The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow, there are no decision rules and no loops.
- **USER EVENT:** An action initiated by the user or the computer. For example, a user event would be any mouse movement or keystroke.
- **INTERFACE:** An interface can be thought of as a contract between the system and the environment. In a computer program, the 'system' is a function or module in question, and the 'environment' is the rest of the project. The interface formally describes what can pass between the system and the environment.
- **CONSTRAINT:** A constraint is a restriction on the degree of freedom you have in providing a solution.
- **GUI**: Graphical User Interface is a system of interactive visual components for computer software. A GUI displays objects that convey information, and represent actions that can be taken by the user.
- **THROUGHPUT:** In general terms, throughput is the rate of production ot the rate at which something(data specifically) is processed.
- **NON-FUNCTIONAL REQUIREMENTS:** In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. They are contrasted with functional requirements that define specific behavior or functions.

1.5 Document Conventions

In general, this document follows IEEE formatting requirements.

- Figure captions are centered.
- Page numbers are displayed at the end of the page in the center.
- The size of each paper is a4.
- Font size is kept 11pt.
- The **default font for LaTeX**, Knuth's Computer Modern, is used throughout the document.
- Starting alphabet of every word in a heading is kept Capital.
- The headings are default sized bold.

1.6 References and Acknowledgments

- ANSI/IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications
- ANSI/IEEE Std 1233-1996, IEEE Guide for Developing System Requirements Specifications
- `https://www.slideshare.net/Axphey/the-complete-srs-documentation-of-our-dev`
- `https://www.slideshare.net/NadiaIIT/final-project-report-of-a-game`
- "SOFTWARE ENGINEERING" by Ian Sommerville (10th Edition, PEARSON)
- "SOFTWARE ENGINEERING A Practitioner's Approach" by Roger S. Pressman (7th Edition, McGraw Hill Higher Education)

Chapter 2

Overall Description

2.1 Product Perspective

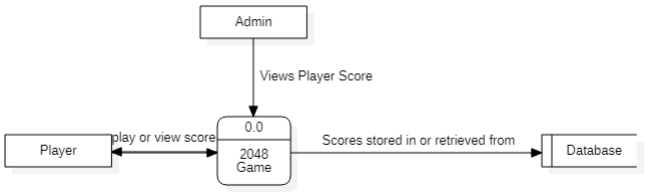
2048 is a single-player sliding block puzzle game designed, originally by an Italian web developer, Gabriele Cirulli. The game's objective is to slide numbered tiles on a grid to combine them to create a tile with the number 2048. Its user-friendly interface makes it easy for people of all ages to play this game with minimal difficulties.

2.2 Product Functionality

The way this single-player game functions has been mentioned below.

- Player will have to enter his name before playing the game.
- Player will then be directed to 4x4 block puzzle which would, initially, contain only tiles with the number 2 written on both tiles.
- Player will then use arrow keys to slide the tiles up, down, left and, right within the 4x4 block.
- Upon every move, a new tile would appear in the block with the number 2 written on it.
- The game will get over if the player runs of the available spaces for sliding the tiles in his endeavor of creating the tile with the number 2048.
- Upon every valid move, player's score will be updated.

The following Level-0 DFD illutrates the idea.



2.3 Users and Characteristics

Since, it is a single player game, therefore, only a single player will be communicating with the system at a time. Moreover, it is to be noted that the players are expected to be windows literate. This means that the players must be aware of the fundamental components of the computer system and should be able to use button, pull-down menus and similar tools. This is because the developers of this game have such people on their priority list of satisfaction. In other words, people who are devoid of the knowledge (regarding the above-mentioned subjects) are, unfortunately, not considered as an audience for this application.

2.4 Operating Environment

This game is, in fact, a desktop-based application and has been developed on the Windows Operating System. The player will be required to have Windows 7 or above version of the Windows Operating System to play this game along with a keyboard, a mouse and a speaker (which are the only hardware requirements for this game).

2.5 Design and Implementation Constraints

The design and implementation constraints include;

- **Size Of The Application:** The entire application would occupy around 2MB disk space.
- **Operating System:** The game would be played on Windows Operating System(Windows 7 and above) only.
- **Hardware Components:** A basic desktop computer or a laptop is necessary for the player to play the game.

- **Medium of Communication:** The user interface of this game would be in English Language. Hence, only the people who are aware of this language would be able to play.
- **Performance:** The application would respond to the user's move within 2 seconds of the user event.
- **Security:** The user will be able to view his score only. However, high score can also be viewed (without viewing the name of the high scorer).

2.6 User Documentation

While designing the game and implementing it, following sources would be referred for help:

- The working of the game would be studied at `https://en.wikipedia.org/wiki/2048_(video_game)`.
- For help while implementation, online help would be sought at `https://www.codeproject.com/Articles/1173541/A-Walkthrough-to-Implement-Game`.

2.7 Assumptions and Dependencies

2.7.1 Assumptions

As mentioned before, the application would be developed on Windows 10 Operating System. Currently, it is being assumed that the game would be played easily on Windows 7 or above versions of the Windows Operating System.

2.7.2 Dependencies

Since it is desktop application, therefore it would be highly dependent on a desktop computer or a laptop (with Windows 7 or above installed). In case of desktop computers, power failure may have an adverse effect on the working of the game thereby causing it to lose the player data/score(if they were not saved properly).

Chapter 3

Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The user interface contains three GUI windows with which, the user will interact.

1. **The Main Menu Window;** Figure 3.1 shows three options. The user can click on these options using the mouse.

2. **On Clicking 'Play' Option;** after the user has entered his name, the current window will display a new page, The game window: Figure 3.2, on which the user can play his game. In this window, the user will use his keyboard arrow keys to play the game whereas the new game can be clicked on to play the game from the beginning.

3. **On Clicking The 'Scoreboard' Window;** the user will be directed to the scoreboard window, Figure 3.3 The dustbin or the delete option can be used by clicking on it, but the user will have to enter the correct password using his keyboard first. The back-button in any window will lead to the previous window while the quit option in the mainmenu window will shut the game application.

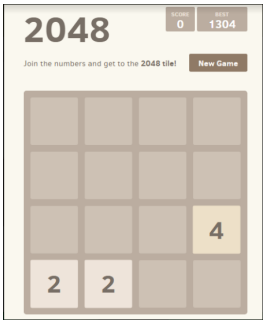


Figure 3.1: The Game

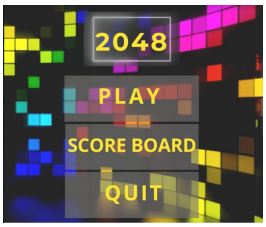


Figure 3.2: Main Menu

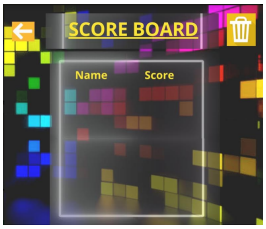


Figure 3.3: Scoreboard

3.1.2 Hardware Interfaces

Hardware Interfacing would be done by;

- **Peripherals:** The peripherals include keyboard, mouse, CD-ROM drives, etc.
- **Display Adapter:** The software requiring computer graphics display, like graphics editors.
- **Secondary Storage :** Hard disk requirements vary depending on the size of the gaming application as mentioned in **section-2.5**
- **Processing Power :** Central Processing Unit(CPU) is the processing power for any system.

3.1.3 Software Interfaces

Since it is a desktop application, therefore, our software interface will consist of operating system and database only.

- **Operating System :** Windows Operating System (Windows 7 or above) will correctly support this gaming application.
- **Database :** The database, where the player's records will be stored, is implemented using MySQL.

3.1.4 Communications Interfaces

This application is a desktop application. This means that it would not require any bluetooth or internet connection to communicate with the user/player. Therefore, it does not require any communication interfaces. The game can be played easily while being offline.

3.2 Functional Requirements

The Functional Requirements for this application are mentioned below.

- As the user/player runs the file, a screen, showing the Main Menu Window (as shown in Figure 3.2) should appear instantly.
- As the user clicks the *"Play"* button, he would be asked to enter his name.

- As the user click on the *"Go"* button, a screen (as shown in Figure 3.1) would appear. This would be the main screen enabling the user to play.
- The progress of the player would be lost if the player exits the game without completing it or without running out of available spaces for more tiles.
- The score of the player will be updated after every valid move.
- The user will be able to view his score throughout the course of playing the game as well as after the game gets over.
- Any previous moves of the player would not be reversed.
- The user/player can also view the current high score. However, the details of the high scorer would be kept private.

3.3 Behaviour Requirements

3.3.1 Use Case View

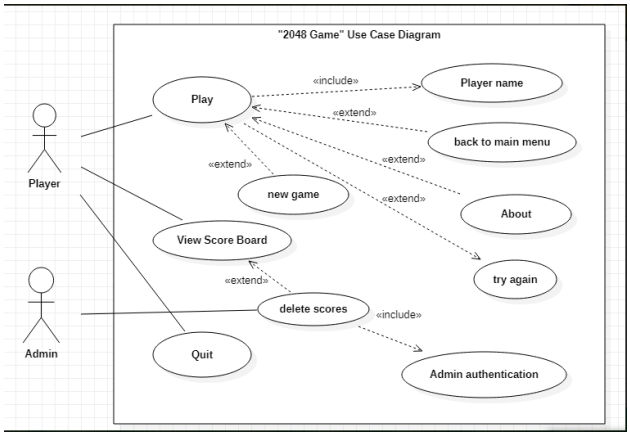


Figure 3.4: Use Case View

The description of the use cases given in Figure is given below.

- **USE-CASE "PLAY" :** When this button is pressed, the game screen will display the game window.

- **USE-CASE "PLAYER NAME" :** Upon pressing the Play button, the player will be prompted with a pop-up window where he must enter his name.
- **USE-CASE "BACK TO MAIN MENU " :** The screen window will go back to displaying the main menu.
- **USE-CASE "NEW GAME" :** If any moves have been made in the game, pressing this button will reset all and bring the game in its initial state.
- **USE-CASE "TRY AGAIN " :** When the player loses the game, a 'try again' button allows him to begin the game again.
- **USE-CASE "ABOUT" :** Any information such as developer names or game details will be displayed when user presses this button.
- **USE-CASE "QUIT" :** The user can exit the game when he clicks this button.
- **USE-CASE "VIEW SCORE BOARD" :** The player names in ascending order of their scores will be displayed.
- **USE-CASE "DELETE SCORES " :** If the actor is an admin, he can also delete the score records.
- **USE-CASE "ADMIN AUTHENTICATION " :** To make sure that only admin deletes the scores, the user will have to enter an admin password when he tries to delete scores.

Chapter 4

Other Non-Functional Requirements

4.1 Performance Requirements

Performance Engineering encompasses the techniques applied during SDLC to ensure that the non-functional requirements for performance(such as throughput, letency or memory usage) will be met. The instruction will be executed instantly within 2 seconds and if user shows any movement, it will be responded.

4.2 Safety and Security Requirements

Safety is at the inner core of the system, providing de facto better isolation. Security layers or different criticality levels are deployed around it. For security admin, password is to be entered to delete the score which is why, our system will be secure. Secondly, this application is a desktop application so it does not require internet. Consequently, it would be safe from hacking attempts.

4.3 Software Quality Attributes

- **To Ensure Reliability And Correctness:** Upon every user event, the user/player should see the effects within around two seconds.
- **For Adaptibility And Flexibility:** The game would save the player's record automatically.

- **In Terms Of Usability:** The GUI will be very intuitive for the player to use.
- **Data Integrity:** The score of the player is stored in the database at the back-end and will be retrieved from it and presented as ouput on the screen as the game gets over.