SE 3XA3: Software Requirements Specification
Group309-2048

Yanting, Cao
400067576, caoy31

Chris, Fu
400084178, fus13

Shengchen, Zhou
400050783, zhous20

March 13, 2020

Contents

List of Tables

List of Figures

1 Project Drivers

1.1 The Purpose of the Project

2048 is a fascinating puzzle game and quickly drew public's attention as soon as it came out. Nowadays, there are still people addicted to the game and play it while they have spare time. However, sometimes the game can be annoying since it has randomness of generating tiles and limited size of grid, which make it super hard to go beyond 2048. With the main purpose of encouraging players to challenge the records, we decided to add some fun features and make some adjustments. For example, the change of grid size and addition of special tool like bomb/switch. It is possible that we add another game mode in which players try to reach 2048 in a time as short as possible. Thus, our project provide a fresh experience of 2048 more interesting features and provide more goals for players.

1.2 The Stakeholders

1.2.1 The Client

Client of the project includes the professor, TA and other individual who review the project and give advice about the project during its development.

1.2.2 The Customers

Customers of the project includes who access and play the game by different motivation. They should can access internet, use browsers and they should be interested in the project.

1.2.3 Other Stakeholders

Other stakeholders include those who developing the project, which are the group members, other group who developing similar project. Stakeholders also include those who do not involve in developing of the project and have potential to be users of the project. Since the project is based on existing product(the game 2048), there's no need for further market analysis or other similar process.

1.3 Mandated Constraints

1.3.1 Solution constraints

Description: The project shall run on most mainstream operating system, which including but not limited to recent version of Windows, linux, Mac OS and other operating system who has ability to run a web browser.

Rationale: The project is based on javascript, thus it requires its user can access appropriate version of JavaScript.

Fit criterion: The project demands its user have JavaScript on their computer, the project shall provide download page for JavaScript if no JavaScript on target computer.

Description: The project shall require steady internet connection to operate.

Rationale: The project is only playable under appropriate internet conditions(enough internet speed) for users to load the game and use some of its features.

Fit criterion: The project shall detect internet condition and give advice to the user on whether they can have a decent experience.

1.3.2 Partner of Collaborative Applications

The project has no direct partner of collaborative applications, but it do requires browsers, internet JavaScript and a terminal to operate.

1.3.3 Anticipated Workspace Environment

The project is preferred to be working in libraries for better communication, however, working at home is not restricted.

1.3.4 Off-the-shelf Software

The project requires JavaScript and web browser to operate.

1.3.5 Budget Constraints

The expecting budget spent is 0 dollar.

1.3.6 Schedule Constraints

The project must be finished by the end of the term.

1.4 Naming Conventions and Terminology

1. The project/the product: the group project we are working on.
2. WB: web browser.
3. Terminal: A computer or phone.
4. JS: JavaScript.
5. FR: Functional requirement
6. NFR: Non-functional requirement
7. git: Git lab
8. Client: The people that the product developed for
9. Program: The coding that execute the game.
10. HTML: Hypertext Markup Language

1.5 Relevant Facts and Assumptions

The game we use is open sources and so do our development tools(git, and JavaScript), thus there's no potential legal issues.
We assume that during the develop process all syntax/text and pictures are open source and can be use freely.
All requirements are based on the criteria outlined for proper use of the product. If definitions change during development, requirements will change accordingly.
All requirements are written under the assumption that our product is able to run on users' hardware - the user's hardware has asupports the technology of our product
It is assumed that the product is running in an environment with internet connection.

Table 1: Work Partitioning

Event Name	Input/Output	Summary
Player starts game	Game interface output	Game interface should be displayed when player starts playing
Player moves tiles	keyboard input	Tiles move in the direction
Player restarts game	mouse input	Player choose to start a new game
Player uses special tool	mouse input	Player choose to use their switch or bomb tool
Player closes the game	mouse input	Game progress saved when game closed.

1.5.1 User characteristics

Intended users do not have to meet a strict set of requirements to use our product. Users can be a PhD student or no education. They are not required to have past experience of similar types of games either. In general, it is considered that the user should:
1. Have a hardware with a web browser to run our product on, i.e. desktop, laptop, etc.
2. Have basic knowledge of operations for computer with access to internet
3. Be able to read FAQs and understand how to use the product

2 Functional Requirements

2.1 The Scope of the Work and the Product

2.1.1 The Context of the Work

See figure 1.

2.1.2 Work Partitioning

See table 2.

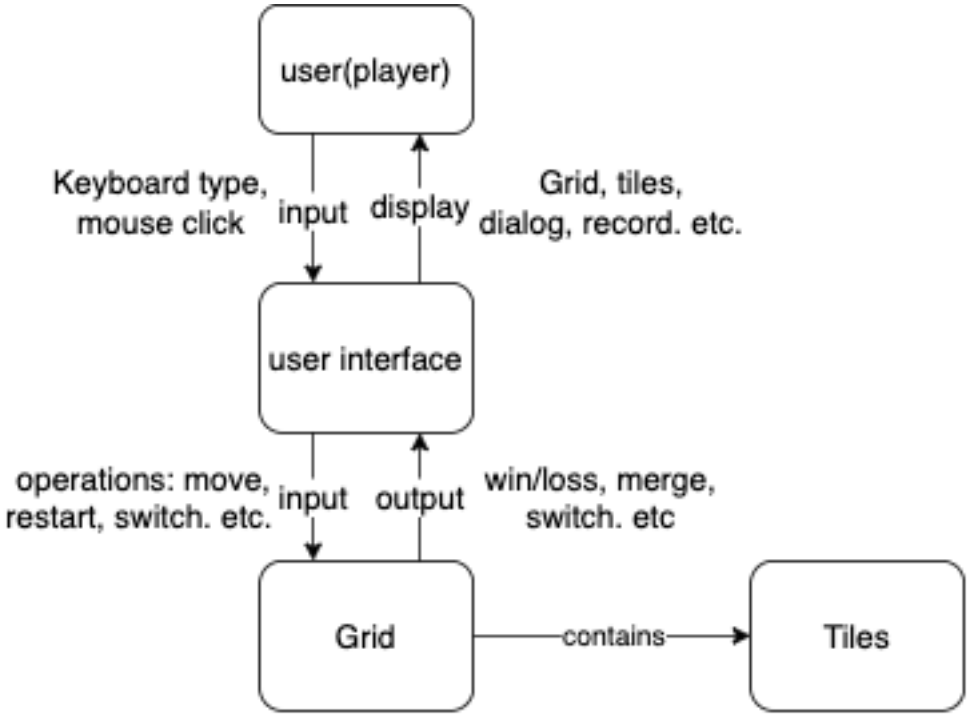

Figure 1: Context of work

2.1.3 Individual Product Use Cases

See figure 2.

2.2 Functional Requirements

2.2.1 Grid

- FR1: The game is played on a 5x5 grid.
- FR2: Every cell in the grid holds and only holds a single tile.
- Fit Criterions/Test Cases for grid:
  1. Execute the game and a 5x5 grid with two tiles with value of two or four is displayed.
  2. When playing, the grid displays all cells in the grid with single tile in each cell.

2.2.2 Tile movement

- FR3: Tiles are moved within the 5x5 grid and cannot be moved outside the grid.
- FR4: Movements can go into the upper, lower, left and right directions.
- FR5: Movements are only successful when adjacent tiles in the direction of the movement have the same value, or when there are tiles can be moved in the direction with a distance larger than 0.
- FR6:A movement should affect every tile on the grid that can be moved in the direction or can be combined together.
- FR7: A movement should send the tiles as far as possible in the given direction.
- FR8: The player can move tiles on the grid using the four arrow keys on the keyboard.

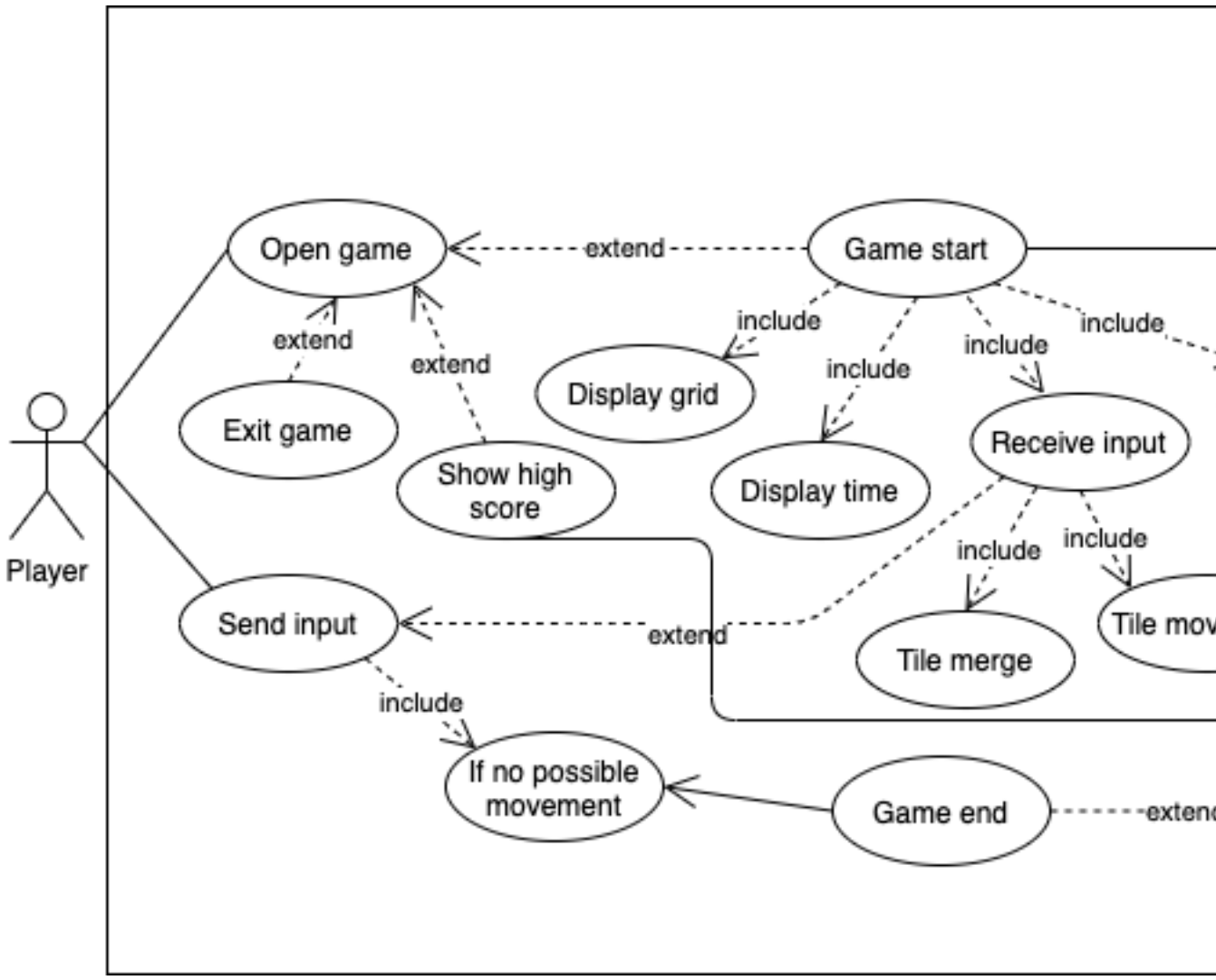

Figure 2: Use Case Diagram for 2048

- FR9: After each successful tile movement, the grid should randomly generate a new tile of value two with a chance of 80%, or a new tile of value four with a chance of 20%.

- Fit Criterions/Test Cases for tile movement:
  Players press an arrow key and the grid will:
  1. move all tiles that can be moved in the direction.
  2. merge all adjacent tiles with the same value in the direction.
  3. do nothing if no possible merge or possible movements.

2.2.3 Tile collision

- FR10: When two tiles with the same value collide, they will merge together into a tile that double the original value.

- FR11: When a tile collides with a tile of different value, it will not merge.

- FR12: A single tile can only be merged once during one movement.

- FR13: Merging two tiles will add the value of merged tile to the score.

- FR14: After merging, the merged tile should stay at the position as far as possible in the movement direction.

- Fit Criterion/Test Case for tile collision:
  Included in Fit Criterions/Test Cases for tile movement or explicitly defined in FRs above.

2.2.4 Game victory/loss stage condition

- FR15: When two tiles of the value 1024 merge into a 2048 tile, it will lead to a game stage of victory.

- FR16: When no more movements are possible, it will lead to a game stage of loss.

- FR17: After victory, the user can decide to continue or to restart the game.

- FR18: If the player decides to continue after victory, he can keep playing endlessly until there are no more possible movements.

- FR19: When player continue the game after 2048, he will lose when there are no more possible movements.

- Fit Criterions/Test Cases for game stage condition:
  1. When no legal movements then game lost.
  2. When two tiles of 1024 merged to one 2048, then player win the game.
  3. After victory/loss the game user interface should display relevant dialogs for further operations.

2.2.5 Special tools

- FR20: After each time that player merged two numbers into a number that larger than or equal to 512, player will receive a special tool to help him proceed the game. The chance for bomb tool is 50% and the chance for switch tool is 50%.

- FR21: Bomb tool allows player to "destroy" a grid of 2x2 selected area, which will delete any tiles in that area and no other effects will be made

- FR22: Switch tool allows player to switch two adjacent tiles, after the switch, two tiles exchange their position and no other effects will be made.

- Fit Criterions/Test Cases for special tools:
  1. Player selects a 2x2 area and use bomb tool, all tiles in area deleted.
  2. Player selects two adjacent tiles and use switch tool, then two tiles exchange their position.
  3. Player receive a special tool each time a number larger than or equal to 512 made.

2.2.6 User Interface

- FR23: The interface should display a board to the user with a 5x5 grid on it.

- FR24: At the start of game, the grid should contain only two tiles with value of two or four.
- FR25: Tiles of different value should have different colours as a distinction.
- FR26: Player should be able to see their current score.
- FR27: Player should be able to see the highest score record.
- FR28: Player should be able to restart the game by clicking on a "new game" button.
- FR29: Player should be able to see the time consumed since the game starts.
- FR30: When victory is achieved, a dialog should pop up, tell player he won and ask whether continue or not.
- FR31: When game is lost, a dialog should pop up, tell player he lost and ask him if he wants to restart.
- FR32: After closing a game, the game progress should be saved and player should be able to start where it closed next time.
- FR33: The highest score record should be saved.
- FR34: Player should be able to see the time consumed.
- Fit Criterions/Test Cases for user interface:
  Explicitly defined in FRs above.

3 Non-Functional Requirement

3.1 Look and Feel Requirements

3.1.1 Appearance Requirements

The product will contain a clean background and not overpower any colors of in-game objects. The color of the 2048 board shall fit the

background color and the color of blocks inside the board will get darker as the number increasing. The product will have blocks for users to use tools located outside the 2048 board with colors different from the blocks inside the 2048 board, and the location shall keep the whole interface tidy. All the colors of the background, the 2048 board and the blocks shall be in red color system (from light red to brown). The whole game interface will be sized to fit properly on a web page.

3.1.2 Style Requirements

The product shall be positive and without any violence. The game shall be a simple style and fitting. The game shall provide the user a relaxing feeling rather than stressful.

3.2 Usability and Humanity Requirements

3.2.1 Ease of use Requirements

The game shall be easy to understand so that the users shall be able to clearly know how to play 2048 in one minute after they start the game.

3.2.2 Penalization and Internationalization Requirements

The game shall provide an intuitive user interface.

3.2.3 Learning Requirements

The gaming scope shall be able to be understood by all the potential users. The user shall know how to use computer and the internet browser.

3.2.4 Understandability and Politeness Requirements

The gaming scope shall be able to be understood by all the potential users. The user shall know how to use computer and the internet browser.

3.2.5 Accessibility Requirements

The game shall be able to be accessed and executed on the majority of web browsers.

3.3 Performance Requirements

3.3.1 Speed and Latency Requirements

The delay between operations shall not affect the user's enjoyment.

3.3.2 Safety-Critical Requirements

The game shall not compromise the user data or machine.

3.3.3 Precision or Accuracy

The game shall use integer values.

3.3.4 Reliability and Availability Requirements

The game shall be available to be played at any time when there is an internet connection.

3.3.5 Robustness or Fault-Tolerance Requirements

The game shall be able to operate without consideration of the physical state of the end users.

3.3.6 Capacity Requirements

The game shall not exceed the server load.

3.3.7 Scalability or Extensibility Requirements

The code is open source and allowed to be modified.

3.3.8 Longevity Requirements

The product shall be relevant for the lifetime of JavaScript functional browsers.

3.4 Operational and Environmental Requirements

3.4.1 Expected Physical Environment

The product is intended to be used anywhere there is a device contains a browser capabilities. The product shall be available where there is an internet connection.

3.4.2 Requirements for Interfacing with Adjacent Systems

The product will be run on most web browsers. The medium will be the host of the website.

3.4.3 Productization Requirements

The product must be hosted by a website for local use.

3.4.4 Release Requirements

The product will be revised yearly and updated according to changing demands and needs of the users.

3.5 Maintainability and Support Requirements

3.5.1 Maintenance Requirements

This product will take maintenance once the clients find problems.

3.5.2 Supportability Requirements

All the potential users shall be able to run the game on their web browsers.

3.5.3 Adaptability Requirements

This game shall be accessible by all the potential users.

3.6 Security Requirements

3.6.1 Access Requirements

This game shall allow every potential users to access.

3.6.2 Integrity Requirements

The game shall not accept invalid user input.

3.6.3 Privacy Requirements

The game shall not access any data outside the game.

3.6.4 Audit Requirements

N/A

3.6.5 Immunity Requirements

N/A

3.7 Cultural Requirements

This game shall not be offensive to religious or ethnic groups.

3.8 Political Requirements

This game shall not be offensive to Political issue.

3.9 Legal Requirements

3.9.1 Compliance Requirements

Personal information shall be implemented so as to comply with the Data Protection Act.

3.10 Standards Requirements

The product shall comply with McMaster standards. See https://libguides.mcmaster.ca/stand

4 Project Issues

4.1 Open Issues

Our Project about improving the game play of 2048 is not complete.

4.2 Off-the-Shelf Solutions

4.2.1 Ready-Made Products

JavaScript

4.2.2 Reusable components

Most of the origin source code we found on GitHub.

4.2.3 Product That Can Be Copied

This project is an open source project. The original implementation can be relied upon as a prototype and source code.

4.3 New Problems

4.3.1 Effects on the Current Environment

The modified project shall not be different from the original version on effecting the current implementation environment.

4.3.2 Effects on the Installed Systems

The modified project shall not have effects on the installed system. This product is implemented on web browsers.

4.3.3 Potential User Problems

There might be potential users may feel as eye soreness or eye strain after they played for a long time.

4.3.4 Limitations in the Anticipated Implementation Environment That May Inhibit the New Product

With adding tools like booms and switch blocks, the top limit of the game might increase a lot so there might be problems that the number of the total score reaches the limit of integer (2,147,483,647 for 32-bit users).

4.3.5 Follow-Up Problems

There will be risk if there comes the new technologies that improves the web browser but will make our game not portable.

4.4 Tasks

4.4.1 Project Planning

See table 2.

4.4.2 Planning of the Development Phases

1. Model implementation: This is the method we will use to set up the game and design every model interacts with each other. This shall be done before starting coding because all the project will be based on the model.
2. Programming implementation: We will use JavaScript to modify and implement this game and use mocha as unit test.
3. Maintenance and Releasing: After coding phase is finished, the rest is to release the game and let user enjoy it. There might be users able to find bugs or other problems from the game, then there will be maintenance to make the game better. If there is no report of problem, the project will be maintained yearly as web browsers might keep upgrading.

4.5 Migration to the New Product

4.5.1 Requirements for Migration to the New Product

N/A

4.5.2 Data That Has to Be Modified or Translated for the New System

Unless no more web browser uses JavaScript, there is no need to modify or translate data.

4.5.3 Risks

N/A

4.5.4 Costs

The common web browsers, JavaScript and Visual Studio Code are all free.

4.6 User Documentation and Training

4.6.1 User Documentation Requirements

N/A

4.6.2 Training Requirements

N/A

4.6.3 Waiting Room

This is an open source project. Everyone can modify or adding functions at any time.

4.6.4 Ideas for Solutions

This is an open source project. Users as well as game designers who find the game have problems can share their ideas for solutions and modify the program at any time.

5 Appendix

5.1 Revision

See table 3.

Table 2: Schedule: Development Plan

Task	**Completer Role**	**Time**
Model Implementation	Software Engineers	Feb 9th
Model Revision	Client	Feb 10th
JavaScript Design	Software	Feb 28th
Implementation	Software Engineers	Mar 13th
Revision	Client	Mar 16th
Release	Software Engineers	Apr 6th
Maintenance	Software Engineers	

Table 3: **Revision History: Requirement Document**

Date	**Version**	**Notes**
Feb 2	Draft	Outlined all sections, templates, responsibilities
Feb 9	0	All sections inserted and completed