

Requirements and Specification Document

By: Christian Hew, Sunaina Gandotra, Shayla Moore, Ned Taylor

Table of Contents

Purpose and Scope	1-2
High Level Description	2-3
Glossary	3-4
Functional Requirements and Use Cases	4-9
Technology Used	10
Other Requirements	10-11
Work Breakdown Structure	12-13
Development Iterations	13-14
Effort Required	14-16

SRS 1. Purpose and scope

Goal:

The goal of this application is to be able to take images, break those images into pieces and create a solvable tile sliding puzzle. If the images are user taken, then those images should be able to be sent to other users who also have the app.

Scope:

This puzzle sliding application is an android based game. The scope of this application will be to determine how the images are broken into pieces, how those

images are stored and transferred between players. It will also include determining the proper/legal movements within the game and what constitutes a winning condition. The scope will also include determining the difficulty of the puzzle, as well as navigation between screens within the application. Outside of the scope will include: 1) restricting the user to what types of images(pictures) they can use, i.e.: what the user takes pictures of in order to turn that picture into a puzzle. 2) Determining how the user will connect to the internet in order to send the take image, i.e.: they can use Wi-Fi, or their phone's data plan.

Stakeholder	Object of stakeholder
User/player	Wants the game to be smooth and an enjoyable experience
Software developers	Wants to develop a fully functional game that allows for interaction between users, also wants to meet all the client requirements
Software testers	Wants modular pieces to the program so that it is easier to test the components of the application, and ensure that all pieces work together fully.
Relatives/friends of users	Wants to be able to interact through the app by sending images to their friends, for their friends to solve. They also want the interaction to be seamless.
Game retailers / app store	Wants to be able to market the application and build a revenue from multiple downloads of the application
Client/ commissioned us for project	Wants to have a fully functional application that meets all of the requirements
Competition game sellers	Does not want this application to succeed because it would cause a loss of profit to whatever application they have on the market at the current time.

SRS 2 a. High level description

We are creating a puzzle game. With this game a user has the option to choose the difficulty of the puzzle and the image to solve. The user can either upload an image from their gallery, take a new photo with their camera, send a friend an image to solve, or solve a friend's puzzle. The user's score is tracked and saved within the game and their highest scores are displayed. The user must swap tiles on the board and when the puzzle matches the original image the game is completed. At any time during the game the user can click a button for help, giving them instructions on how to play, click a button for a hint, displaying the original image as reference in a small window, or click a button to give up and end the game.

SRS 2 b.

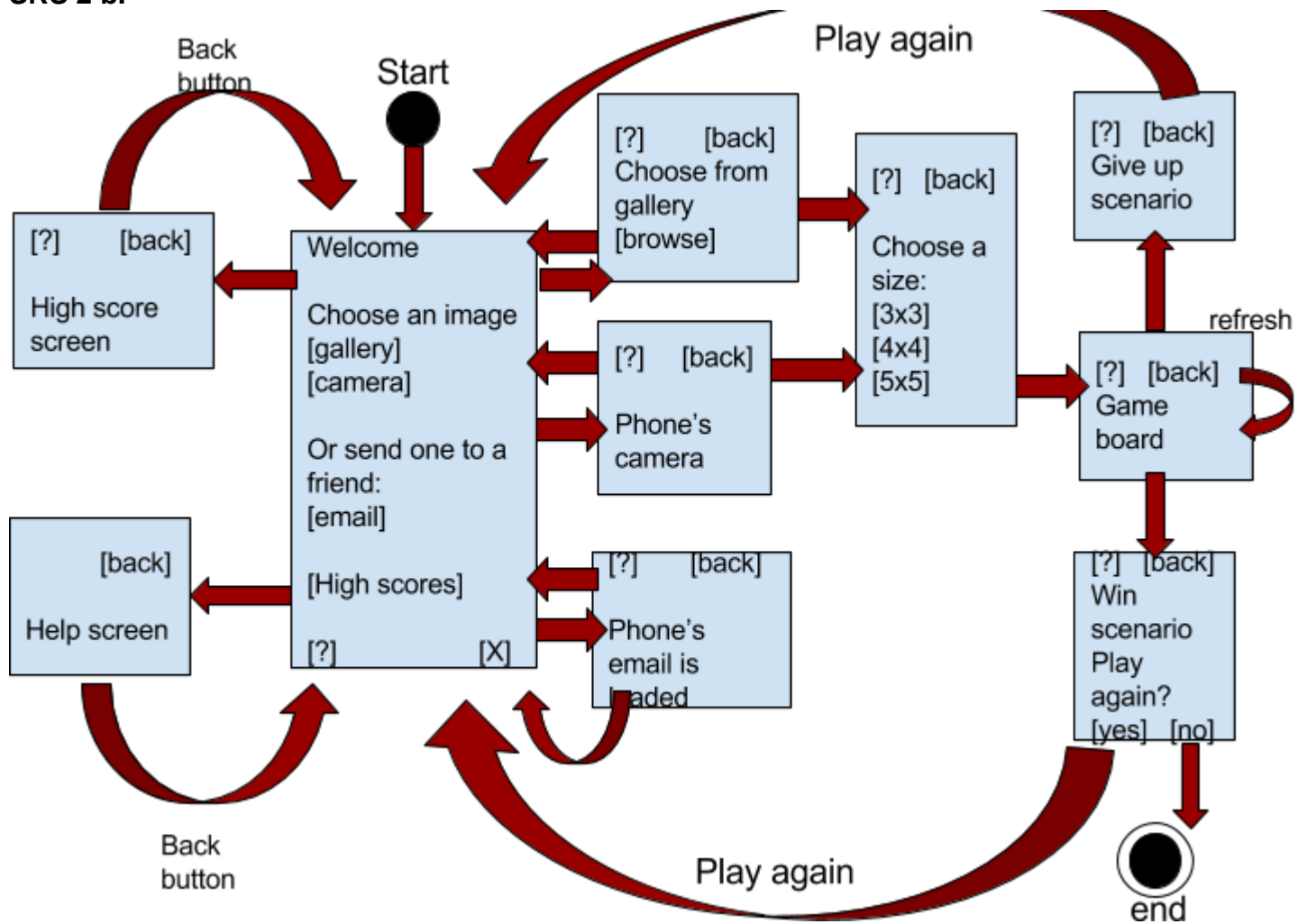




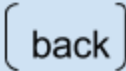


Diagram Key

	Signifies a clickable button
	Transitioning between screens
	Exit button to exit the app
	Help button
	Returns to previous screen

SRS 3. Glossary

Term	Definition
Player	A person who can interact with the game application that has been started and is not terminated
User	A potential player of the game.
Client	Refers to the person who requested the game
Gallery	Refers to the gallery of photos on a player's device.
Puzzle Difficulty	Refers to the size of the grid. For example, an easy puzzle is 3x3, a medium is 4x4, and a hard puzzle is a 5x5.
Swapping	Switching the location of a blank tile with an image tile that is either above, below, to the left, or to the right of the blank tile.

SRS 4. Functional Requirements

1. The primary actors in this program will be the players of the game. As a player of the game, there are three major goals for a successful game. Firstly, a player wants to be able to solve a puzzle from their gallery. Second, a player wants to be able to send a

picture to another phone for their friend to be able to solve. Finally, a player wants to be able to receive photos from their friends through the game and solve them.

2. Our program will have the following features:

- a. A player will be able to choose between three options on opening screen:
solve an existing puzzle, send a puzzle to a friend, or solve a puzzle that a friend has sent them
 1. If the player clicks the “solve an existing puzzle”, the gallery of their pictures will appear and they will choose a photo to solve
 2. If a player picks the “send a picture to a friend”, the camera will appear and they will take their picture and email it to their friend. When the second user receives the email, they must save it to their phone’s gallery in order to access this from the app.
 3. If the player chooses to solve a puzzle that was sent to them, the program will open the phone’s gallery so the user can select a photo. This photo will then be used for the player’s puzzle and the player will be redirected to the game board.
- b. A player will have access to a “back” button to bring them back to the welcome screen no matter where they are.
- c. A player will have a “refresh” button, which will restart their game. (the board will re shuffle and allow the player to resolve the puzzle)
- d. After the player chooses to solve a puzzle, they will have the option to choose their difficulty. Difficulties are categorized by grid size.

e. When the player completes a game correctly, a message will appear telling them that they have won. Two buttons will also appear 1) play again(yes), which will send the user to the home screen 2) exit button(no) which will allow the user to exit the application

f. Our program will count the number of moves it takes to solve a puzzle and record that. In the main screen the user can see the high scores (i.e.: lowest number of moves) for each puzzle difficulty.

g. Our program will split a user chosen image and break that image into several parts based on the difficulty the user has selected) (easy - 3x3. Medium - 4x4. Hard - 5x5)

h. our program will allow the user to swap the location of the tiles by swiping the tiles to another tile within bounds. (within bounds: not off the screen, and must be directly above, below, left, right of the tile).

3. Our features, listed above, are ranked as follows from most important to least:

- F (must have)
- G (must have)
- A.1 (must have)
- A.2
- A.3
- D
- E

- B
- C (future work)
- F (future work)

Fully dressed use case 1: Playing a 5 by 5 puzzle from gallery

Primary actor: game player

Precondition: Player has opened the app and have gotten to the welcome screen

Stakeholders and interests: Game player wants to play a 5 by 5 game of a picture that they picked from a gallery of pre-existing photos.

Preconditions: The player has opened the application and has a gallery that is stocked with photos.

Main Success Scenario:

1. Player selects the button to solve a puzzle from the pre-existing gallery
(Alternative flow 1: gallery is empty)
2. Player chooses photo he or she wants to solve from the gallery.
3. Player chooses the “5 by 5” button for the size.
4. Player is directed to the game board and begins to solve the puzzle (The player will solve the puzzle by sliding tiles into the correct location, i.e.: forming the image in its completeness)
5. When the player wins, he or she is given the option to play again (Alternative flow 2: Player gives up)

Alternative Flows:

1. If the gallery is empty, an alert will prompt the user to pick a different option on the welcome screen
2. If the player gives up, they will be redirected to the welcome screen where they have the option to play a different puzzle.

Notes and Issues:

- We should note that the user has no limit on the number of moves before they have to give up; so Alternative Flow #2 is only possible if they click the give up button.

Fully Dressed use case 2: Player 1 wants to send a picture to Player 2 and Player 2 wants to solve it

Primary actors: Player 1, who is sending the photo, and Player 2, who is solving the photo

Precondition: Both players have their applications open and on the welcome screen.

Stakeholders and interests: Player 1 wants to use his or her camera to take a photo and send it to Player 2 to solve. Player 2 wants to receive the photo, pick the grid size for the puzzle, and solve the puzzle.

Main Success Scenario:

1. On the welcome screen, player 1 chooses the option to send a picture to a friend.
2. After choosing that option, the camera opens and the player takes a photograph.
3. The photograph is sent to the email of the other user (Alternative Flow 1: Email fails to send)

4. An alert appears on the screen of Player 2 telling him or her that a picture has been sent to them and asks if they want to solve it. (Alternative Flow 2: Player 2 says no to solving the picture)
5. Player 2 is directed to a page where they can choose the grid size.
6. Player 2 solves the puzzle and is alerted that they have won. (Alternative Flow 3: Player 2 gives up)

Alternative Flows:

1. If the email fails to send, Player 1 is alerted and is redirected back to the welcome screen where they can choose to send a photo to Player 2 again if they choose to.
2. If Player 2 gives up, they are redirected to the welcome screen.

Notes and Issues:

- Note that both players must have the application open for this process to work.

Casual Use-Cases:

1. If the player wants to review the rules of the game or are lost at any point, they will click the help button that will be present on every screen. An alert will appear that will contain the rules and how to play the game. When the player is done reading over, they click "Done" and the alert will disappear.
2. If the player wants to look up the high scores of the game, which correlate to the lowest amount of moves needed to complete a given puzzle at a certain difficulty, they click the "High Scores" button and a record of the top scores appear. When they are done looking, they click the "Back" button and they appear on the welcome screen again.

Problem / Problem Domain:

The main problem that can be run into with developing this application is making the game device independent. In other words, the board size might look perfect on an Android phone, but not look correct on a tablet. Creating the board size dynamically is extremely important in achieving this goal. In addition, another problem is the issue of user error. For example, the player entering the wrong email to send a photo to would result in an error because the user might have sent the image to the wrong person. In the case of an incorrect email address.

5. SRS - Technology Used

List technology requirements

The technology requirements for our app will be that the app runs on an Android phone.

- The app must be able to run primarily on the Android OS marshmallow. Android phones of multiple screen sizes without warping the contents of the app. Another technology requirement is that a user of the app may be able to send an image to another app user in order to create a puzzle with it. This requires two Android phones in this situation.
- Another system that our app will possibly interact with is a database system in order to transfer photos from in between app users. If not the database and the server, then email will be used as the technology to transfer emails.
- Our system will also have to interact with the android studio development environment, this is because our app will be created using this environment.

6. SRS - Other Requirements

Performance and scalability related requirements

- Scalability-wise, if the database route is pursued for the handling of sending images between users, it should be able to handle increasing amount of traffic on

the server that will handle it. We are not sure how we will do this yet. If the email route is pursued, it will, possibly, be easier to handle scalability issues as the app will just send the picture in between users to their respective emails.

User and usability related requirements

- The app will be very usable as we will include instructions on how the game is played if the user needs help. Also, the interface will be attractive and readable for easy use. We will test this by having multiple users try the application and rate their experience and the look of the application from 1(dissatisfied) - 10(very satisfied)

Maintenance and portability related requirements

- Maintenance requirements will be that the code will be commented and stylistically written so maintenance can be allowed in the first place. The app will be tested vigorously for bugs and after the app is released, if there is a bug, the app will be corrected and maintained.

DEV 1. Software development process

The development process that we would use is agile development process. Through agile development, we begin by establishing our requirements and goals for our software. We then go through a series of integration and testing before moving to the next functionality of the software. When getting the base requirements completed, we demo our project for our client and retrieve any feedback for the necessary improvements. We then make our changes and continue with our testing. After all functionalities are complete we release our software and continue to retrieve feedback

from our users for necessary improvements and bug fixes. With new feedback, we continue to integrate and test new or improved functionalities of our game. This continuous process is essential for our game due to the image sharing amongst friends. By transferring data between phones, we open up our software to greater changes of bugs and functionality issues, especially as phone technology continuously improves.

Dev 2. Work Breakdown Structure (* these estimations give extra time to allow for setback in other areas, both with the project and outside of it)

1. Navigation elements - 2 hours
 - 1.1 implement button from home screen -> game screen - 1 hour
 - 1.2 implement button from game screen -> home screen - 1 hour
2. Image processing - 100 hours
 - 2.1 import images and store them in a location - 15 hours
 - 2.2 break images apart into - 25 hours
 - 2.3 store images into grid layout and - 30 hours
 - 2.4 shuffle image location - 30 hours
3. Game mechanics - 50 hours
 - 3.1 slide pieces of the images between positions - 25 hours
 - 3.2 restrict player movements to legal moves (a player can only swap places with pieces that are directly above, below, left, right of the intended swapping piece, this means the player cannot swipe diagonally, or off the puzzle board) - 25 hours
4. Win logic 48 hours
 - 4.1 determine if puzzle pieces are in correct order - 40 hours

- 4.2 lock puzzle if it is in correct order - 15 hours
- 4.3 display screen to play again button - 3 hours
- 5. Image distribution - 80 hours
 - 5.1 make connection between email and image - 40 hours
 - 5.1.2 specify type of image file (must be .jpg, png, jpeg)
 - 5.2 down load image from email - 40 hours
- 6. Appearance/features - 24 hours
 - 6.1 determine button style - 8 hours
 - 6.2 determine button layout - 8 hours
 - 6.3 general color scheme of app - 8 hours

Dev 3 - Development Iterations

- Planning: In this step, we will get together and discuss what the project is supposed to do. For us, this includes the fact that the app is supposed to be a puzzle game in which users can send each other pictures and solve them. We get all this information from the client.
- Requirements: Requirements for this app have to do with hardware, software, and time constraints. We have planned, for instance, that Navigation elements will take 2 hours, Image Processing will take 100 hours, Game mechanics will take 50 hours, Win logic will be 48 hours, Image distribution will be 80 hours, etc. We also keep asking the client if these are what he wants requirement-wise.
- Analysis & Design: This is where we start making UML diagrams with our java classes and what they do to help the app function. We will analyze logic and how the puzzle, logically, can be solved and incorporate that into our design. For example, we will focus on the Win logic to determine if puzzle pieces are in correct order, and to lock the puzzle if it is in correct order.
- Implementation: Implementation, for us, is going to mean coding the app. After all the designing is done, we should have a good idea on how to code our app. As well as coding, unit testing will be applied as we code. Our implementation will have a modular and object orientated design.
- Testing: While the code is written, numerous tests will be applied and recorded on an excel sheet for whether the test passed or failed. If the test failed, it will be corrected so that it passes the next time.

- Maintenance: After the app is deployed, bugs still may happen. If they happen, our team will be working on fixing and maintaining the app as deemed fit by the client.

(due date, person on a specific task (how they will accomplish it), tied to a calendar, testing for each iteration)

- Sprint 1 (framework) / 5 hours
 - Ned
 - Help button notification
 - Sunaina
 - Implementing home screen transition
 - Christian
 - Text formatting
 - Shayla
 - Button formatting
 - Test transition between views
- Sprint 2 (image handling) + testing 7
 - Ned
 - Importing images to a gallery
 - Sunaina
 - Resizing images
 - Christian
 - Maintaining image quality/ Accessing images in gallery
 - Shayla
 - Splitting images
- Sprint 3 (message alerts) + testing
 - Ned
 - Transition from board state give up (back button)
 - Sunaina
 - Transition from board state win (play again / exit)
 - Christian
 - Welcome screen high score notification
 - Shayla
 - Error handling non valid email
- Sprint 4 (tile sliding) + testing
 - Ned
 - Working on scrambling the image
 - Sunaina
 - Working on sliding over a tile not under it
 - Christian
 - The sliding motion
 - Shayla
 - Allowing you to only slide images one cell at a time

- You cannot slide a cell from (1,1) to (1,3), only from (1,2) to (1,3)
- Sprint 5 (win condition) + testing
 - Ned
 - Replay notification
 - Sunaina
 - Whether the board is in a winning state
 - Christian
 - Alerting the user to a winning board
 - Shayla
 - Freezing the board upon a win
- Sprint 6 (image sending) + testing
 - Ned
 - Accessing the camera
 - Sunaina
 - Sending a picture to another device
 - Christian
 - Importing the photo to a gallery in the application
 - Shayla
 - Notification of a new image in gallery

DEV 4 - Effort Required

- a. Breaking up the project into parts:
 - i. Shayla will spearhead the design portion of the project - designing the application logo, the color scheme and font layout of each screen, and the layout of the game board.
 - ii. Sunaina will focus on the testing and documentation as well as manipulating the photo within the game - in other words, given a scrambled photo, how swapping the tiles would look
 - iii. Ned will focus on coding the interaction between one phone and the other- emailing a photo taken with a camera to a given email address, retrieving the picture from an email in the application, and making that photo into a game ready to be played.
 - iv. Christian will be focusing on the win logic - how the game recognizes a win, how to make the screen lock after a win is recognized, and the screen transition back to the welcome screen.

b. Table of tasks with their time requirement:

Name of Task	Related Tasks	Required time to complete
Screen transitions (from welcome screen to game screen etc.)	Assigning onClick methods to every button and starting activities as necessary	5 hours with testing
Photo manipulation for the game (mixing up the photo selected into an appropriate number of pieces given the board size)	Parsing the image into pieces programmatically, storing the original order of those pieces, and then mixing them up	75+ hours with testing
Moving the tiles on the board	Making sure the tile that is being moved slides in front of the tile it is replacing and not behind it	25 hours with testing
Recognizing a win	Checking the board after every move against the stored original order of the pieces and then displaying a message if the orders match	40 hours with testing
Interaction between phones	Taking a picture with a camera, emailing it to another phone through the application, retrieving the photo through the application, and then performing the regular scrambling of the photo and playing the game	75+ hours with testing
Design	Finding the best combination of color, font and overall look to fit the personality of the game	10-15 hours (no testing required)