ulm university universität
uulm

Faculty of Engineering, Computer Science and Psychology
Databases and Information Systems Department

Conception and Realization of a Brain and Memory Training Application

Bachelor's Thesis at Ulm University

Submitted by:
Sina Vähäkangas
sina.vaehaekangas@uni-ulm.de
832026

Reviewer:
Prof. Dr. Manfred Reichert

Supervisor:
Michael Winter

2020

Version November 26, 2020

Satz: PDF-LaTeX 2$_\varepsilon$

Abstract

In this bachelor's thesis, an Android brain and memory training application BrainBoozled for devices with an Android operating system was developed. This application is focused on different aspects of training the human brain in regard to memory, speed and logical thinking. Nowadays, there are already many brain and memory training applications for Android operated devices. The application BrainBoozled consists of three different brain and memory training games. The game Process Models makes this application stand out with making the user recreate different kinds of process models from memory. First, a process model is displayed. The user then rebuilds the entire process model by dragging and dropping its elements together under a specific time limit. By playing this game, it is easy to learn different structures of process models in an easygoing and fun way. Additionally, this game contains two other games focusing on memory training and logic under a time limit. The Squares game creates different patterns using black and white squares. The user tries to remember the displayed pattern and rebuilds it by tapping on the different squares of the pattern to change their colour between black and white. The Numbers game has a logical aspect to it. Randomly generated numbers are shown on the screen with a specific order. The user has to quickly sort the displayed numbers either in an ascending or descending order by selecting all numbers on the screen in a specific time limit. This game makes the user think logically under a time limit and trains the brain in a different way.

Acknowledgement

First of all, I want to thank Michael Winter for his helpful advice and feedback throughout the creation of this thesis. Additionally, I would like to thank him and Prof. Dr. Manfred Reichert for making this thesis possible.

Furthermore, I would like to thank my loved ones for their constant support as well as motivating and believing in me during this thesis as well as during my entire studies at the University of Ulm.

Contents

1 Introduction

1.1 Motivation

The brain is a complex part of the human body. By frequently training and exercising the brain's functionalities and memory, everyday skills are being improved, which can lead to better performances in working and social life. This is why the brain should often be trained and challenged. [6]

Considering there is a lack of a brain and memory training application for Android operated devices that uses process models for brain and memory training, the application BrainBoozled was developed and implemented. This game uses process models of different structures and elements, unique patterns and sorting numbers to challenge the user's logical thinking and memory training. It makes the user use logic and memory to solve different kinds of games in a challenging way. The games of the application BrainBoozled are focused on different aspects of brain and memory training to achieve a variety to inspire and challenge the user.

1.2 Objective of thesis

The goal of this thesis is to develop and design a memory training application for devices with an Android operating system with the help of process models, different patterns and logic to increase the efficiency and memory of the human brain. This application should provide a fun environment, which does not only challenge the user's brain and memory but also provides the user with amusing and educating games. The three games included in this application should make the games interesting due to their differences and variations with a smooth game

flow. Additionally, the games should have a clean and simple design, which invites the user to play the games and train his brain and memory.

1.3 Structure of thesis

This thesis consists of 8 chapters. Chapter 2 introduces the context of use in detail and establishes relevant technologies that were used to develop this application and describes their working structures. Hereinafter, in chapter 3 the functional 3.1 as well as the non-functional requirements 3.2 are stated and described.

Chapter 4 analyzes the technologies from chapter 2 and explains in a more detailed way their functionality including their usage for the development of this application. Additionally, this thesis presents in chapter 5 the concept and design of this game. This thesis shows first drafts in section 5.1 of the application, that were created before the app development started. Subsequently, chapter 5.2 displays the design and appearance of the application and explains in detail the rules and structure of the three implemented games.

Afterwards, chapter 6 compares the requirements that were created in chapter 3 before this app was developed and analyzes which requirements were necessary, implemented differently or skipped completely during the development process.

Chapter 7 leads a discussion about similar theses and their differences to the application BrainBoozled developed for this thesis. Finally, chapter 8 shows an outlook on future prospects. This chapter discusses the possibilities of expanding this application with new features for future implementations.

2 Overall description

2.1 Context of use

An important part of the human body is the brain. Strengthening your memory will improve everyday performance as well as efficiency at work and boost confidence. The application BrainBoozled is developed for devices with an Android operating system to train and maintain the brain's functionality. It is suitable and developed for people of all ages. To use the brain's full potential, it has to be trained and challenged regularly and consistently. [6]

BrainBoozled consists of three different games. All of these games have their focus points on different areas to train the brain and memory efficiently. The game Process Models trains the brains visualisation skills, speed and memory in terms of remembering and recreating different process models before the given time runs out. The game Squares is fully focused on training the memory part of the brain. It makes the user use his memory to remember a pattern with black and white squares, after which the user has to rebuild the same pattern. The game Numbers focuses on logic and speed by making the user select a variety of different numbers in a specific order during a time limit.

2.2 Business Process Model and Notation 2.0

Business Process Model and Notation, or BPMN for short, is used to describe business processes and models. It is visualized with predefined objects. It persists of different flow objects connected to each other with sequential flow objects (Figure 2.1). Business Process Models that use BPMN 2.0 are highly adjustable and can be modified easily in order to change a process model or to add more process steps

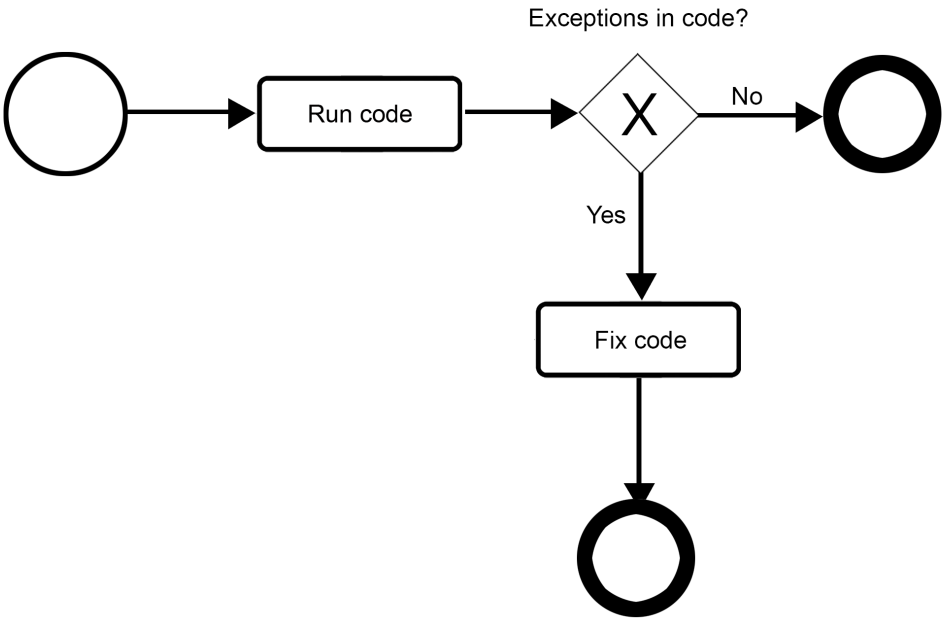

Figure 2.1: Process model for testing an application for bugs

to it. With process models, different work flows and operations can be described clearly and well structured. [4]

2.3 Android Studio

The Android Studio development environment was used to develop this application. It is the most used platform for native Android development. Android Studio has a lot of advantages to help develop native Android applications efficiently. Android Studio is easily operated as well as clearly structured and is split into different sections. [3]

The activities contain all the logic and implementation of all screens displayed in the application. They are combined with an XML layout file to store information about the different objects of the layout and their design attributes. The XML layout files can be viewed in Code and Design mode. This makes it simpler and more efficient to better adjust the view and design of the app without having to restart the application to apply design changes. [3]

One major advantage of using Android Studio as a development platform for a native Android application is the Debugger. The Debugger lets the developer set breakpoints all around the code. Subsequently, the application runs through the set breakpoints. Debugging and issue-solving can be more efficient and straightforward with the use of the Android Studio Debugger. [3]

Since Android devices differ in different aspects such as screen size or operating system the AVD Manager is a useful tool when developing an application. The AVD Manager lets the developer run the application on one or multiple different virtual devices. For these virtual devices the Android operating system, screen size and model can be chosen freely. [3]

Considering, Android Studio is one of the leading development platforms for native Android applications, there is a lot of assistance and support in the online community for developing Android applications. [3]

2.4 JavaScript Object Notation

A JSON object, or also known as JavaScript Object Notation, is a data-interchange format which is used to store data. It is supported and used by a variety of platforms such as the Android Studio development environment. A structure of a JSON object can be expanded and modified very freely. [1]

The JSON object is surrounded by two curly brackets. An array in a JSON object is located between two square brackets. The elements of an array are separated by a comma. [1]

A name-value pair stores information for a specific key. The name-value is followed by a colon, after which the value for this name-value is stored. The name-value pairs are also separated by a comma. Since a JSON object can be structured very flexible, it can also be nested and expanded as requested. [1]

Considering a JSON object is very flexible and data can easily be added, it was convenient to use for storing level information as well as the information of the different process models for this application. Since more levels might be added in the future, it was important to store data in a format, which can be modified easily. [1]

3 Requirements

This chapter describes and explains the functional and non-functional requirements that this application should require. These requirements were created before the development phase to give a desired direction of the different functions and features of this application.

3.1 Functional requirements

Functional requirements are used to describe desired functions and features, that should be implemented. These requirements guide the developer through the entire implementation process with specific desired functions and features the application should contain.

3.1.1 General

FR 1: (Android Application) The user is able to use the application on any Android operated device.

FR 2: (Different games) The application offers three different games with different focuses. The three games in question are Process Models, Squares and Numbers. This way the user trains his brain and memory with a focus on memory training, speed and logic.

FR 3: (Instructions) The user can view instructions of each game before starting to play any of the games. These instructions should be brief and clear to the user.

FR 3: (Portrait mode) This application should be viewed in portrait mode.

3.1.2 Levels

FR 4: (Level view) The level view of all available levels should be consistently visible to the user in the application. This view shows the user the amount of completed levels, the current level as well as not competed levels.

FR 5: (Reset game scores) The user has the ability to reset his game progress with all completed levels of the different games through the settings menu at any time. After the game scores are reset, the user can start playing the game from the beginning.

FR 6: (Saving game score) The score of each game will be automatically stored in the storage on the application device and loaded as soon as the user restarts the application.

3.1.3 Process Models game

FR 7: (Time limit) The given process model is shown for a specific amount of time only. Afterwards the user needs to remember and rebuild the shown process model from memory.

FR 8: (Correct input) After placing an incorrect element to the process model, the user should have the ability to correct his input.

FR 9: (Difficulty of levels) The difficulty of the levels increases with the concurred levels to keep the games challenging to the user.

FR 10: (Drag and drop) The user uses drag and drop to move the elements in the correct position to the middle of the screen to rebuild the process model.

3.1.4 Squares game

FR 11: (Time limit) The user has a specific amount of time to remember and recreate the given pattern. If the user does not recreate the correct pattern during this time, the game is ends.

FR 12: (Difficulty of levels) The difficulty of each level increases with the concurred levels. The higher the level, the more squares the user has to memorize.

3.1.5 Numbers game

FR 13: (Time limit) The user has a specific amount of time to select all shown numbers in the right order. If the user is not capable of selecting all numbers in the correct order the game ends.

FR 14: (Difficulty) The user has the ability to choose between 4 different levels of difficulty: Easy, Medium, Hard, Expert. The level of difficulty varies by the time for each round. For each level of difficulty new level information is saved in the phone's storage.

FR 15: (Difficulty of levels) The difficulty of the different levels increases with every concurred level.

FR 16: (End game after wrong input) When the user chooses the wrong number, the game ends immediately, so that the user does not have to wait until the time is up or after the user has selected all other numbers.

3.2 Non-functional requirements

The non-functional requirements describe characteristics of the app in order to provide a high quality with a pleasant and clear user experience.

QA 1: (Operating system) The application should work on every device with an Android operated system.

QA 2: (Robustness) The application should work robustly and load all information the correct way. The application should display the same information after exiting and reopening the application.

QA 3: (Reliability) The game scores and level information should be saved and reloaded automatically so that the application works reliably. The user can continue playing the game at the same level after reopening the application.

QA 4: (Learnability) The rules of each game are brief, structured clearly and understandable so that the user does not need to spend much time with learning and understanding each game.

QA 5: (Usability) The application should be intuitive to the user and at the same time behave the way the user expects the application to react to specific functions.

QA 6: (Design) The application should have a clear and structured user-friendly design. It should make a pleasant impression on the user without distracting the user from the game activities.

4 Technologies

4.1 Game Process Models

For the game Process Models, JSON objects were used to store all information regarding the different levels of all process models and their objects (Listing 4.1). The application loads and reads the JSON object. Subsequently, the process models are drawn according to the level information from that JSON object. The information about the different coordinates of the rectangles and spaces is stored in the JSON object. The rectangles and spaces are used as target positions for the process model elements in the game. The application differs between rectangles and spaces, which are also displayed in different colours. Rectangles are used for the different elements of the process models. Spaces are used for the different connecting objects between these elements. The JSON object also stores the process model element and connecting object.

The level information for the game Process Models is saved in a JSON file in the internal storage of the phone. When the game is started, the level information is read and managed automatically.

4.2 Game Squares

To retrieve the level information for the game Squares a JSON object was used. The Level class contains different functions, which help with the management of the different level information for all three games. To retrieve the level information, the Level class receives the filename and the context of the current activity and reads the JSON data as a String (Listing 4.2).

```
    "level1": [{
        "image": "level_1.png",
        "rectAmount": "4",
        "spaceAmount": "3",
        "rectCoordinates": [{
            "rect": "1",
            "x": 50,
            "y": 100,
            "processmodel": "CIRCLE_START"
          },{
            "rect": "2",
            "x": 450,
            "y": 100,
            "processmodel": "TASK_A"
          },{
            "rect": "3",
            "x": 850,
            "y": 100,
            "processmodel": "TASK_B"
          },{
            "rect": "4",
            "x": 1250,
            "y": 100,
            "processmodel": "CIRCLE_END"
          }],
       "spaceCoordinates": [{
            "space": "1",
            "x": 250,
            "y": 115,
            "processmodel": "ARROW"
          },{
            "space": "2",
            "x": 650,
            "y": 115,
            "processmodel": "ARROW"
          },{
            "space": "3",
            "x": 1050,
            "y": 115,
            "processmodel": "ARROW"
          }]
        }]
```

Listing 4.1: JSON data for the first level of Process Models

```
public static String loadJSONFromStorage
        (String filename, Context context) {
    String json = null;
    try {
        File file = new File(context.getFilesDir()
            + "/" + filename + ".json");
        FileInputStream is = new FileInputStream(file);
        int size = is.available();
        byte[] buffer = new byte[size];
        is.read(buffer);
        is.close();
        json = new String(buffer, "UTF-8");
    } catch (IOException ex) {
        ex.printStackTrace();
        return null;
    }
    return json;
}
```

Listing 4.2: Load JSON object from phone storage

```
// Get current level list
levelList = Level.getLevelList
    (this, "numbers_" + difficulty);
```

Listing 4.3: Retrieve level information for the Numbers game

4.3 Game Numbers

The level list for the game Numbers was saved and loaded from a JSON object. Since there are four different difficulties, a JSON file with the level information was added to each of the different difficulties. The filename consists of the game name and the difficulty. Each difficulty has its own level list, so that the user can play each of the difficulties with all levels (Listing 4.3).

5 Concept and Design

5.1 Mock-ups

A mock-up is a design prototype of a software or application, which is created before the actual implementation and development of a software or application is started. A mock-up has the purpose of guiding the developers and relevant people of the project of the future application design and logic. This way the involved individuals can view a possible design for the application when programmed is and can easily decide on modifications by thus avoiding a longer process for the implementation. The mock-up is part of the planning part of a software project development. By the help of a mock-up, some thought process issues can be caught and fixed accordingly before investing too many resources in development. For the Android application BrainBoozled, the mock-ups were created with the graphics editor Adobe Photoshop CS5. [2]

5.1.1 Main

Through the main view (Figure 5.1) the user has the ability to read brief instructions of all three games. The main view also guides the user to the settings menu, where the user can reset all game scores if wanted. The user is also able to start playing any of the three games through the game activity (Figure 5.2) which can be accessed through the main view.

5.1.2 Level view

The level view (Figure 5.3) displays all available levels for the different games. From here the user can start playing each game.

Figure 5.1: Main Activity

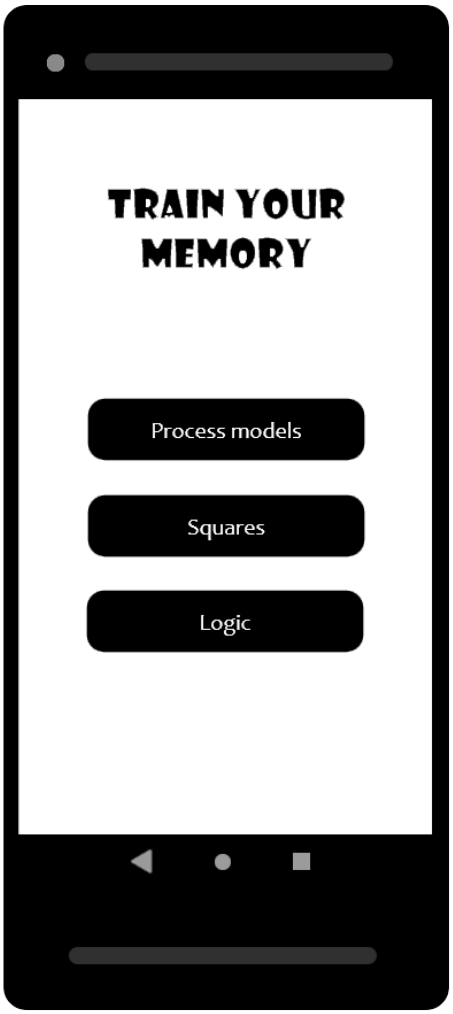

Figure 5.2: Game Activity

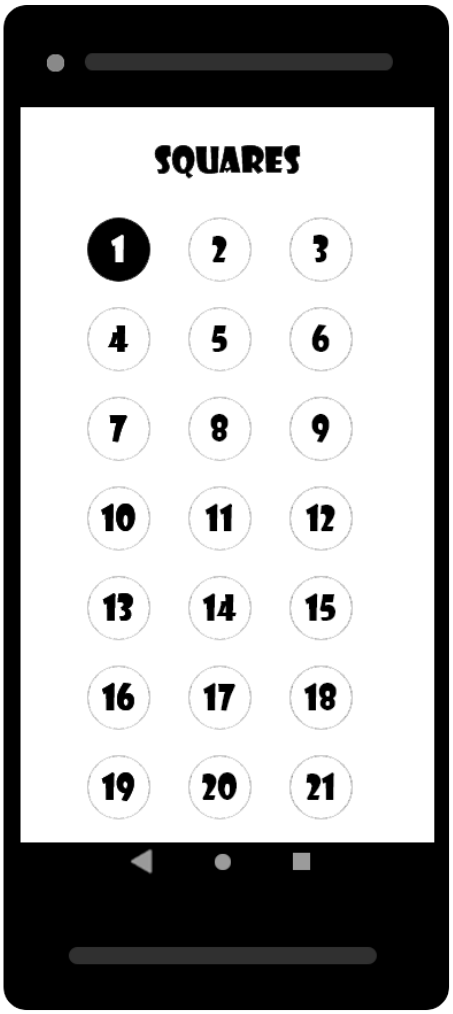

Figure 5.3: Level view

5.1.3 Process Models game

The user views a process model in the view (Figure 5.4). During a specific amount of time, which is displayed on the bottom of the screen, the user has to remember the displayed process model. After the time runs out, the user is automatically forwarded to the next screen (Figure 5.5) to rebuild the just viewed process model from his memory.

From the different process model elements, the user has to choose the correct elements and place them in the correct order in the middle of the screen by dragging and dropping the object (Figure 5.5).

5.1.4 Squares game

In the game Squares, a specific pattern of white and black squares is displayed. Due to the fact that there is no time limit in this game, the user can take as much time as wanted for remembering the patterns. Afterwards, the user clicks on a button and is then forwarded to the next screen (Figure 5.7) to recreate the pattern.

In the view the same pattern with only white squares is displayed. The user has to tap on the different white squares to turn them black to recreate the pattern from the previous screen (Figure 5.6).

5.1.5 Numbers game

In the Numbers game, the user is first shown an order and a variety of randomly generated numbers (Figure 5.8). The user starts selecting the numbers in the correct order before the time runs out. Subsequently, the user selects a number on the screen. When a number is selected, it disappears from the screen. The game ends as soon as the given time runs out, the user selects the wrong number or all numbers are selected in the correct order.

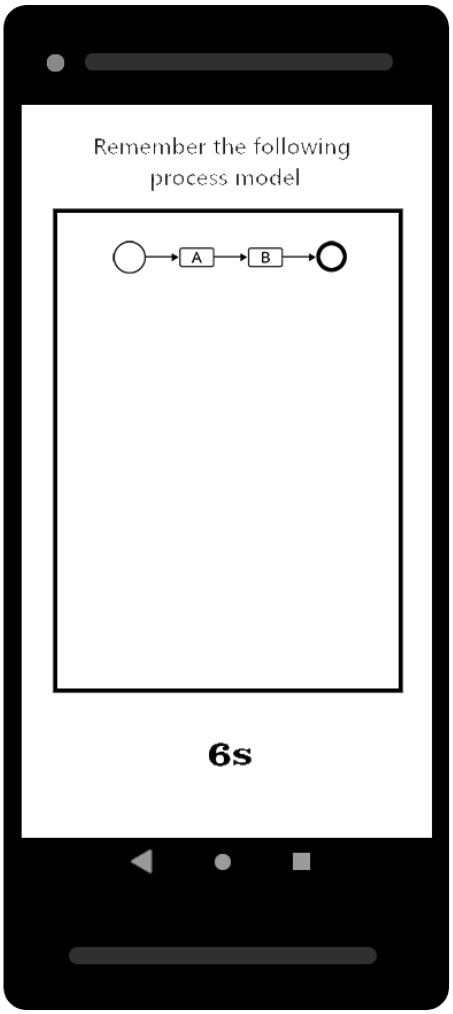

Figure 5.4: Process Model Game 1

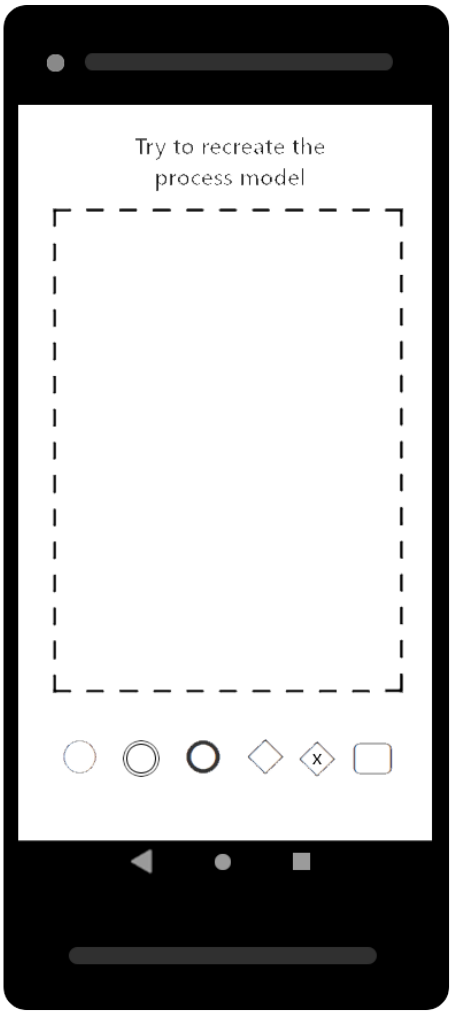

Figure 5.5: Process Model Game 1

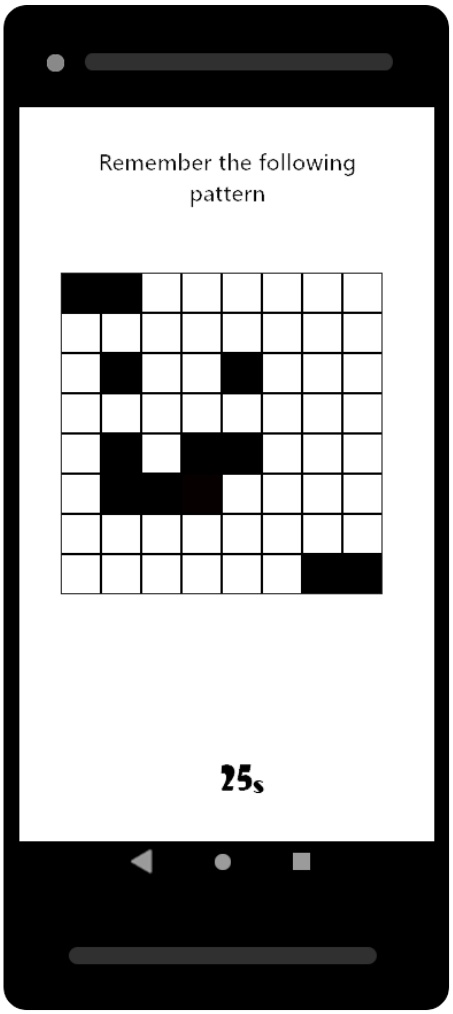

Figure 5.6: Squares Game 1

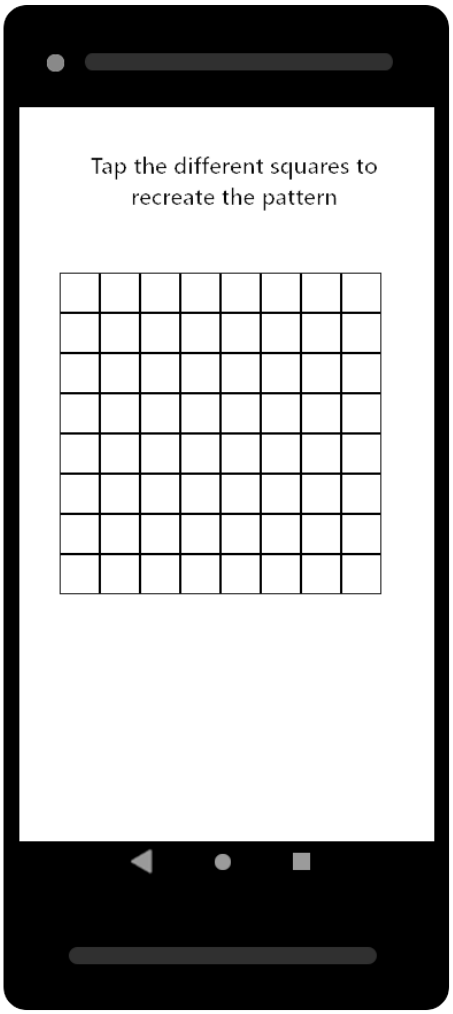

Figure 5.7: Squares Game 2

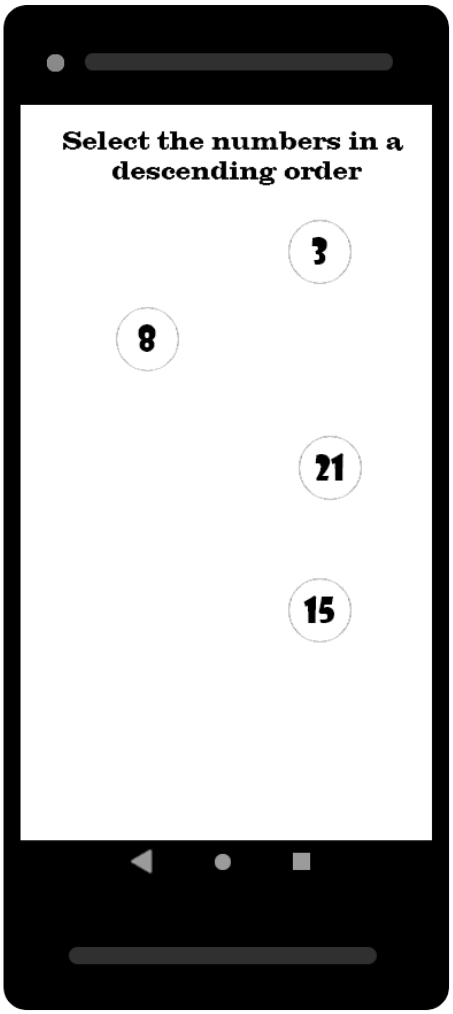

Figure 5.8: Numbers Game

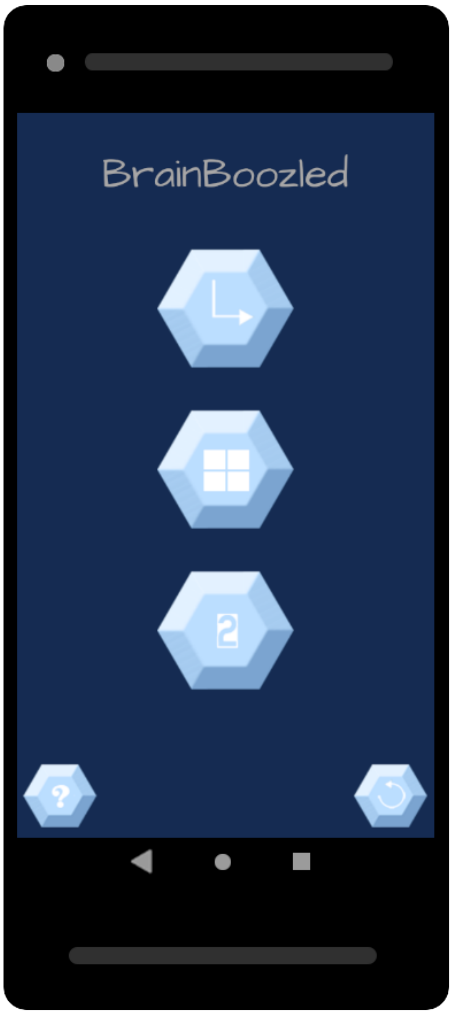

Figure 5.9: Main Activity

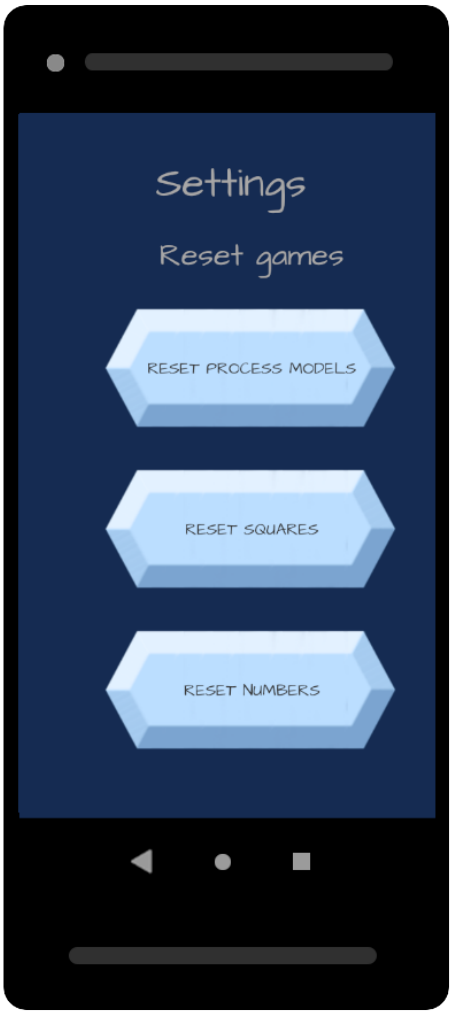

Figure 5.10: Settings Activity

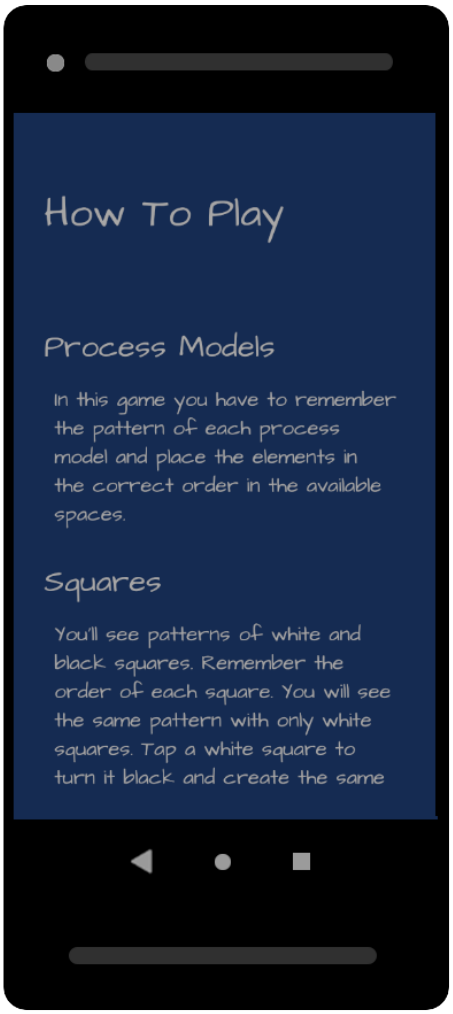

Figure 5.11: How To Activity

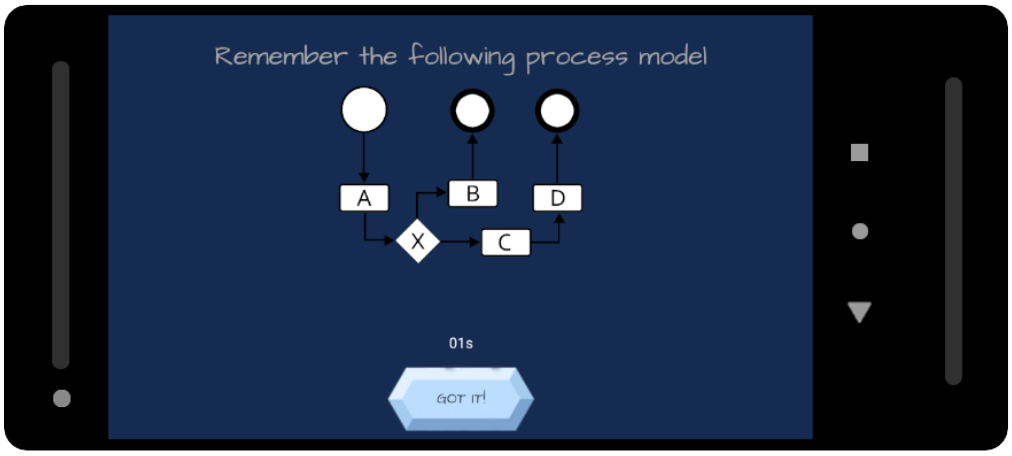

Figure 5.12: Process Models Game Activity 1

5.2 Presentation of the App

The brain and memory training application BrainBoozled consists of three different games training memory, speed and logic with different techniques (Figure 5.9). The user has the possibility to restart each game at any point. All game scores can individually be reset from the settings view (Figure 5.10). From the main view (Figure 5.9) the user has the possibility to read brief instructions for each individual game (Figure 5.11) and start playing each of these games.

The user views a different process model each round in the game Process Models (Figure 5.12). Then, the user has to remember the order of all flow and connecting objects in a certain time span. Afterwards, the user assembles the process model by using drag and drop to move the objects from the bottom of the screen to the correct rectangles in the middle of the screen. (Figure 5.13) The process models become more challenging the higher the level (Figure 5.14).

The second game Squares displays a pattern with both white and black squares (Figure 5.15). The user has to remember the position of all the white and black squares without a time limit. After this, the user can view the same pattern with only white squares and has to turn the white squares into black squares by tapping on the specific white square to complete the pattern (Figure 5.16). The difficulty gets

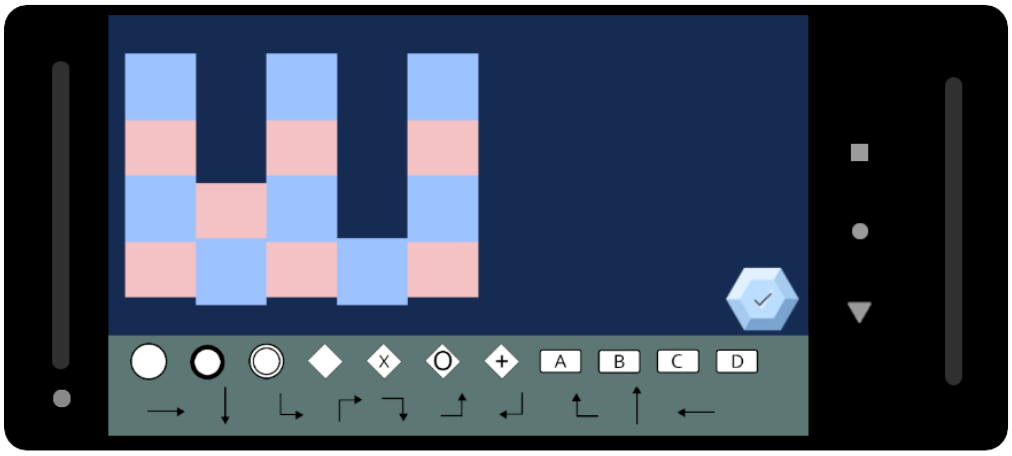

Figure 5.13: Process Models Game Activity 2

higher with each level by incrementing the number of squares (Figure 5.17) the user has to memorize.

For the Numbers game, the user first chooses a specific difficulty. There are four different difficulties available: Easy, Medium, Hard and Expert (Figure 5.18). The different levels of difficulty vary by the time limit being available for each round (Figure 5.19). When the game starts, the user sees numbers on the screen and an order he has to select the numbers in. The order, either ascending or descending, is being randomly generated for each round as well as the numbers (Figure 5.20). In the next step, the user has to select the numbers in the correct order within the time limit. The levels get more challenging by the amount and range of the randomly generated numbers.

A level class was implemented, since all games control their levels the same way. This level class handles all the level coordination for each of the games. For every function, the game name has to be provided as a parameter. The level class takes care of saving, loading and resetting levels as well as getting detailed information about the different levels (Listing 5.1). This way the app can be extended more easily in case new games are being implemented.

During the process of implementation, there were some aspects requiring more attention due to their complexity. Since there are a lot of different objects and

Figure 5.14: Process Models Level Activity

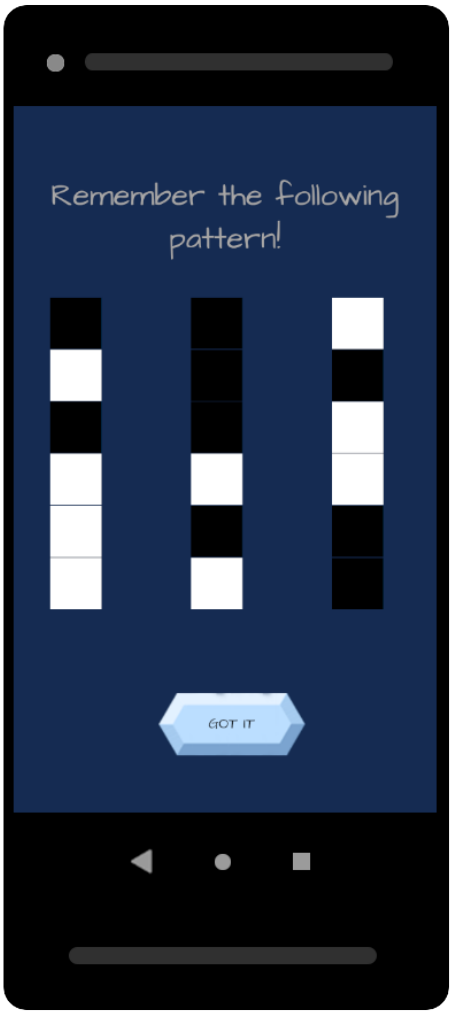

Figure 5.15: Squares Game Activity 1

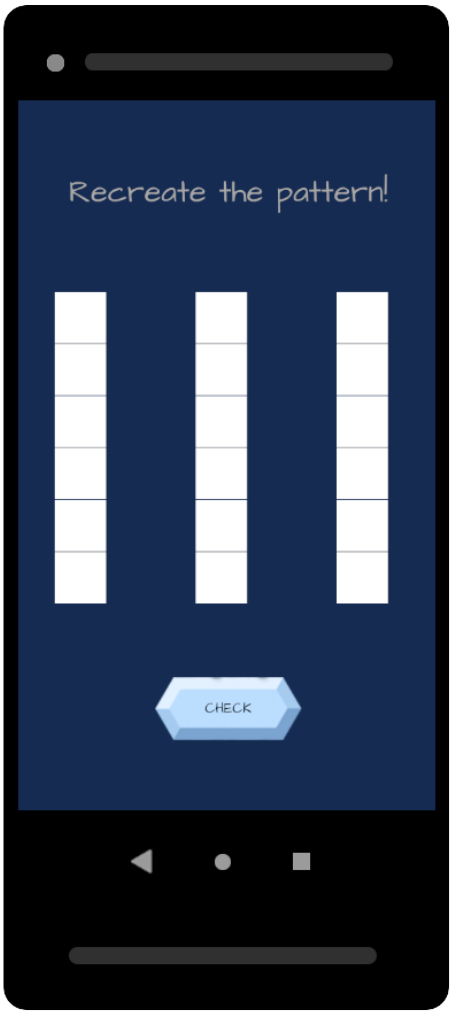

Figure 5.16: Squares Game Activity 2

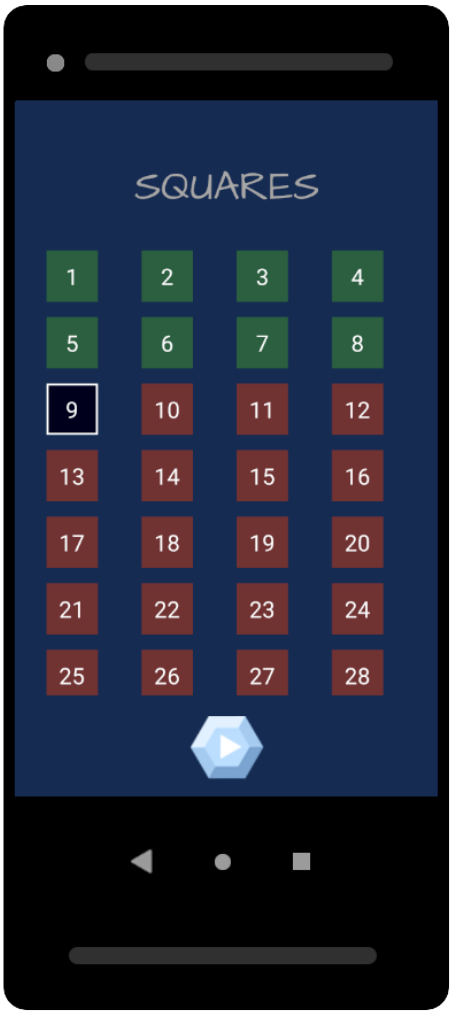

Figure 5.17: Squares Game Levels Activity

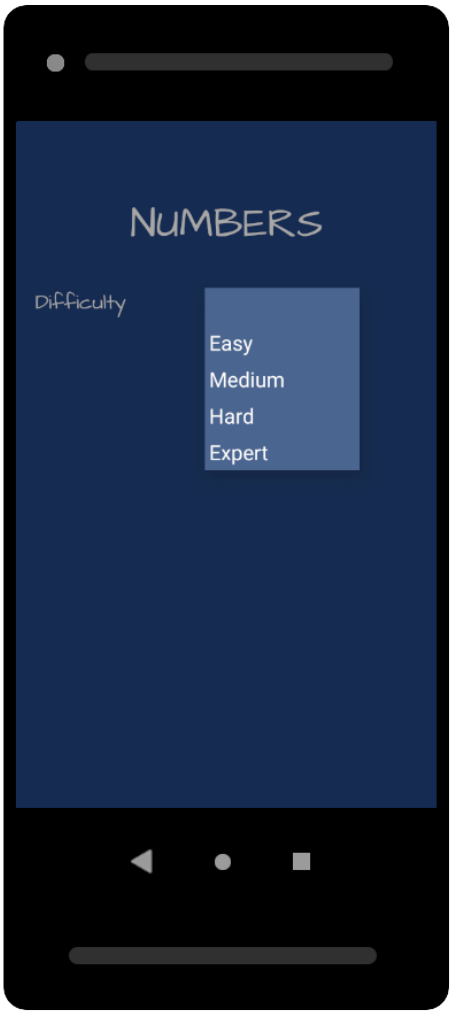

Figure 5.18: Numbers Difficulty Activity

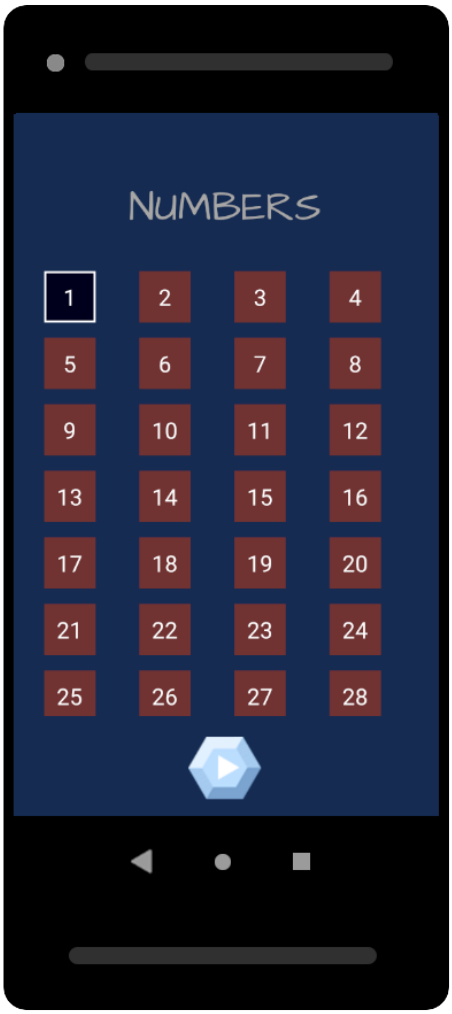

Figure 5.19: Numbers Levels Activity

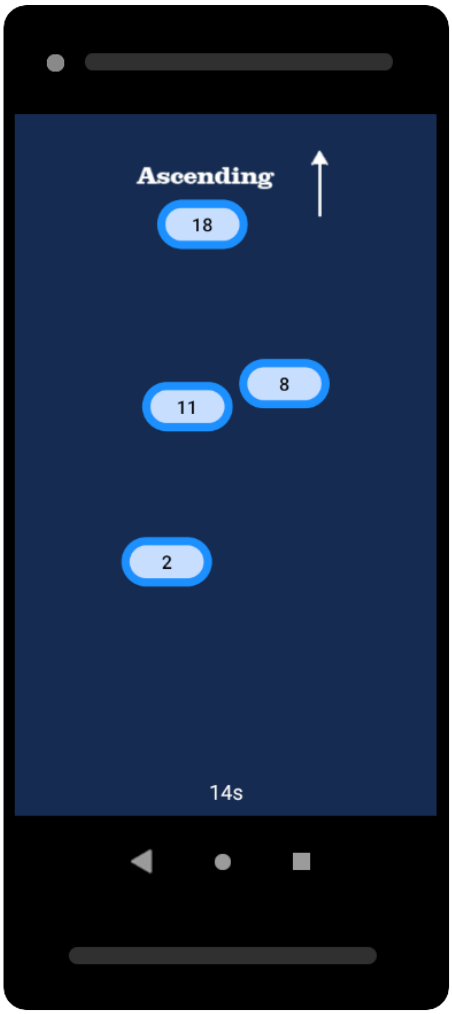

Figure 5.20: Numbers Game Activity

```
1  public static List<Level> getLevelList
2      (Context context, String game) {
3          List<Level> levelList = new ArrayList<>();
4          File file = new File(context.getFilesDir()
5          + "/" + game + ".json");
6
7          if(file.exists()) {
8              String gameStr = loadJSONFromStorage
9                  (game, context);
10             Gson gson = new Gson();
11             Type listType = new TypeToken
12                 <List<Level>>(){}.getType();
13             levelList = gson.fromJson
14                 (gameStr, listType);
15         }
16
17         return levelList;
18     }
19 }
```

Listing 5.1: Retrieving levellist from a JSON object

```
View view = (View) dragEvent.getLocalState();
ViewGroup owner = (ViewGroup) view.getParent();
owner.removeView(view);
LinearLayout container = (LinearLayout) layoutView;
container.removeAllViews();
container.addView(view);
view.setVisibility(View.VISIBLE);
```

Listing 5.2: Removing all items from a rectangle

rectangles the implementation of the drag and drop offered a challenge. To receive a fluent and functioning drag and drop feature, it was important to make sure all the correct objects were dragged in the middle of each rectangle with only one object visible at a time. This issue was solved by removing all views from the rectangle before adding a new dragged item into it (Listing 5.2).

In the game Process Models every rectangle is being drawn with coordinates from a JSON object. These rectangles are being saved in an array list. If the user drags an object into a rectangle, a function was created receiving the coordinates from the dragged object, looping through the list of rectangles and spaces to check which rectangle the object was drawn to. After finding out, which rectangle the object was drawn to, the object was saved to an array list containing all the dragged objects. This made it more straightforward to check whether the user dragged every element correctly or incorrectly when getting the results (Listing 5.3).

For each round, the Squares game generates randomly a pattern, which the user needs to remember. The amount of squares is generated by the current level the user is currently at (Listing 5.4). Each of these squares is either filled with white or black colour. The colour of the squares is being set by a random variable.

The Numbers game contains an algorithm which creates random numbers and random orders. For every round, the sorting order of the numbers is created randomly with the help of a random boolean (Listing 5.5).

The numbers are generated randomly. During the creation of a new number, the application checks, whether the same number has already been generated (Listing 5.6) or not. In case it has been generated and is already a chosen number, a new generated number is created. This procedure ensures that the user does not

```
public boolean onDrag(View layoutView,
    DragEvent dragEvent) {
        int action = dragEvent.getAction();
        switch (action) {
            case DragEvent.ACTION_DROP:
                View view = (View) dragEvent
                        .getLocalState();
                ViewGroup owner = view.getParent();
                owner.removeView(view);
                LinearLayout container = layoutView;
                container.removeAllViews();
                container.addView(view);
                view.setVisibility(View.VISIBLE);
                int[] location = new int[2];
                layoutView
                    .getLocationOnScreen(location);

                int spacesPosition = -1;
                int rectPosition = checkRectPos
                    (location[0], location[1]);
                if(rectPosition == -1) {
                    spacesPosition = checkSpacePos
                            (location[0], location[1]);
                }
                if(rectPosition > -1) {
                    rectangleList.get(rectPosition)
                            .setProcessModelElement
                                (getLastProcessModel());
                } else if(spacesPosition > -1) {
                    spacesList.get(spacesPosition)
                            .setProcessModelElement
                                (getLastProcessModel());
                }
                layoutView.setVisibility(View.VISIBLE);
            default: break;
        }
        return true;
    }
}
```

Listing 5.3: Drag and drop

```
private List<Integer> createPattern() {
    List<Integer> pattern = new ArrayList<>();
    for(int i=0; i<amountSquares(); i++) {
        pattern.add(randomInt());
    }
    return pattern;
}
```

Listing 5.4: Create pattern algorithm

```
Random ascDesc = new Random();
    if(ascDesc.nextBoolean()) {
        sortDirection = "ascending";
    } else {
        sortDirection = "descending";
    }
```

Listing 5.5: Create random boolean

become confused during the game due to the uncertainty which number he should choose first. The purpose of this game is to sort the numbers in an ascending or descending order, so this issue was solved by adding a simple function of checking if the number already has been generated.

The numbers in the game Numbers also have randomly generated coordinates for the screen position. In order to avoid the numbers being too close to each other and the user not being able to select the correct number, a function was created. This function checks if the given coordinates have enough space in each direction so that there is no issue of the numbers overlapping (Listing 5.7).

```
private boolean checkIfNumberAlreadyExist(int number) {
    boolean check = false;
    for(int i=0; i<randomNumbers.length; i++) {
        if(randomNumbers[i] == number) {
            check = true;
        }
    }

    return check;
}
```

Listing 5.6: Function to check double numbers

```
private boolean checkPositionOverlap(float x, float y) {
    boolean check = false;
    int bubbleRadius = 150;

    for(int i=0; i<randomNumbers.length; i++) {
        if(positionXs[i]-x < bubbleRadius
            && positionXs[i]-x > -bubbleRadius) {
            if(positionYs[i]-y < bubbleRadius
                && positionYs[i]-y > -bubbleRadius) {
                return true;
            }
        }
    }

    return check;
}
```

Listing 5.7: Algorithm to check number overlapping

6 Comparison of requirements

This chapter describes the implemented features of the mentioned requirements from chapter 3.

During the implementation, the general requirements were not all implemented in the specified way 3.1.1. There were some differences in the implementation of the general requirements (Table 6.1). The changes were necessary in order to offer the user a more clear design.

The levels and level view are a crucial part of this application. Consequently, the level and level view were implemented regarding to the level requirements from subsection 3.1.2. The stated requirements for the level view do not differ from the implemented features for the level view (Table 6.2).

The requirements for the game Process Models were not implemented as described in subsection 3.1.3. During the implementation, the features needed to differ from the mentioned requirements, in order to make the game both challenging and accomplishable (Table 6.3).

Some modifications were made to the requirements in subsection 3.1.4 for the Squares game (Table 6.4).

In subsection 3.1.5 the requirements for the Numbers game were implemented accordingly (Table 6.5).

No	Requirement	Implementation
FR 1	Android application	This application was developed for all Android operated devices.
FR 2	Different games	This application consists of the three different games Process Models, Squares and Numbers. This combination gave the application a good mixture for training the human brain and memory.
FR 3	Instructions	An activity called "How To" was implemented, which contains brief instructions of all the games. The user can read the instructions before playing each game and does not need to invest a large amount of time on learning the rules.
FR 4	Portrait mode	The game Process Models was developed in landscape mode instead of portrait mode due to the fact that the process models can be viewed and structured more clearly and the process models have a more user-friendly design.

Table 6.1: Comparison of general requirements

No	Requirement	Implementation
FR 5	Level view	For a user-friendly design, the level view was a crucial feature for the implementation. The user should have a clear view of all concurred and not concurred levels.
FR 6	Reset game scores	It was important that the user can restart each game and reset all game scores, which is why this feature was implemented.
FR 7	Saving game score	The level information is saved automatically to a JSON object. This way the user can concentrate on playing the games without having to remember his current level in each game.

Table 6.2: Comparison of level requirements

No	Requirement	Implementation
FR 8	Time limit	The time limit for the game Process Models had an important effect on the game flow due to the fact that the user gets more challenged when playing this game. The time limit makes sure that the user has to remember the process model faster and more efficiently to concur the different levels.
FR 9	Correct input	The user might accidentally drag and drop a wrong element to a rectangle or space. This is why it is important for the user to be able to correct his input easily by removing the dragged item out of the rectangle or by moving the correct element to the rectangle.
FR10	Difficulty of levels	To give the user a feeling of success and being challenged it was important to implement different levels, which get harder the more levels the user concurs.
FR11	Drag and drop	The drag and drop process of the game Process Models was implemented differently from the planned requirements. The objects are being dragged and dropped into separate rectangles instead of the middle of the screen. Every object has an own rectangle with only one object at a time.

Table 6.3: Comparison of requirements for the game Process Models

No	Requirement	Implementation
FR12	Time limit	During the implementation and testing the Squares game, it did not seem to be necessary to add a time limit to this game. Since the levels get more challenging the more levels the user has concurred, the game becomes difficult without having to add a time limit. Due to the fact that the two other games already have time limits, this feature was not implemented for this game.
FR13	Difficulty of levels	To give the user a feeling of success and being challenged it was important to implement different levels, which get harder the more levels the user concurs. The difficulty of the different levels is measured by the amount of squares in each pattern. The user has more squares to remember.

Table 6.4: Comparison of requirements for the game Squares

No	Requirement	Implementation
FR14	Time limit	The time limit in the numbers game was crucial, so that the game challenges the user. This was an important feature for this game. Without the time limit, the game would be too easy, because the user can take his time to select the correct numbers in the correct order, without it being challenging.
FR15	Difficulty	The difficulties for the Numbers game were implemented, so that the user can decide by himself which time limit challenges him the most, but is not too difficult to concur. During testing, the amount of difficulty levels seemed to be appropriate, which is why four different difficulties were added.
FR16	Difficulty of levels	To give the user a feeling of success and being challenged it was important to implement different levels, which get harder the more levels the user concurs. The difficulty varies by the amount of numbers in one round as well as expanding the range of each number.
FR17	Wrong input	To keep the game interesting to the user, the game ends immediately after a wrong input is made. When the user selects a wrong number, the game is immediately stopped and the user can restart the level.

Table 6.5: Comparison of requirements for the game Numbers

7 Similar theses

In this chapter, 2 theses with a focus on similar topics are explained and their differences to BrainBoozled are discussed. The first thesis is about a memory training application which has its main focus on preventing the brain to reduce its functionality in retirement age [7]. The second thesis focuses on an Android application to learn the MVVM Architecture patterns in terms of a puzzle game [5].

7.1 Brain Workout, An Application for Enhancing Memory and Problem Solving Skills

As people get older and reach retirement age, the brain can more easily diminish its full capacity and functionality as people choose to focus less on work and start retirement sooner. As this thesis states, this can be a cause of dementia if not prevented. Nowadays, modern medicine is always improving, which has the cause of people reaching an older age. As people are getting older and starting retirement sooner, it is important to train the brain's functionality. For this purpose, an application was developed, which focuses mainly on training the brain and enhancing memory capacity of people in retirement. [7]

This application consists of different games training the brain, which enhance the problem-solving skills of retirees as well as improve their memory. The game "Guess the word" chooses a category and shows the user all the letters of one word the user has to guess only in the wrong order. The user has to rearrange the letters so that the word matches the given category. Another game of this application is the game "Play with numbers", which has a focus on solving mathematical equations. The user sees an equation and has to solve it during a specific time limit. [7]

The purpose of this application is similar to the Android application BrainBoozled developed for this thesis. BrainBoozled also has its main focus on training the brain with logic and memory training games. The difference between these two applications is the contents of the different games. While BrainBoozled uses the game Process Models to not only train the brain and memory but also helps the user to learn the structures of process models, the application of this mentioned thesis focuses on training the brain capacity in terms of enhancing the memory and problem-solving skills. [7]

7.2 Puzzle game using Android Model View ViewModel Architecture

This thesis introduces the Model View ViewModel pattern by Microsoft. MVVM is an architecture pattern used to define the state and behaviour of a specific view and is used in XAML platforms. This pattern separates the interface development from the model. [5]

The Android application developed for this mentioned thesis a simple puzzles game, which provides a basis for learning MVVM patterns and their architecture. With this app, the user can learn MVVM Architecture patterns in a playful way. [5]

BrainBoozled differs itself from this application due to its training techniques. The game Process Models teaches the user not only the structure of process models but also trains the brain and memory. The purpose of the application from the mentioned thesis is to learn the MVVM Architecture patterns with a puzzle game. [5]

8 Summary

This thesis is about a brain and memory training application for Android operated devices which was developed with a focus on different techniques to train the human brain in terms of memory training, speed and logical thinking. This application consists of the three different brain training games: Process Models, Squares and Numbers. To date, there is no brain and memory training application for Android operated devices with a focus on remembering and recreating process models with their structures from memory.

More features can be implemented to extend this application and its games. For the game Process Models pools can be added to the different process models. This would make the process models more complex as well as add more difficulty to the game. The game Process Model can also be extended with an automated algorithm creating the used process models by an implemented logic. The application could randomly generate new process models in order to offer more variety. Consequently, the levels vary after each reset due to the fact that the player is able to play the game as often as he wants with different process models.

This game can be extended with more games, which train the brain and memory with different techniques. Another strategy to extend this application is to add more colours and variety to the game Squares. Currently, this game contains both white and black squares for the user for remembering and rebuilding. By adding more colours to the given patterns, the difficulty of this application would increase and make this game more challenging for the user.

Additionally, this application can be extended with a twist to the Numbers game to challenge the logical thinking of the user. For this purpose, the game Numbers can be extended with a multiplayer system. Users can play against each other to select all numbers faster than their opponent. This changes the outlook on the game due to the fact that the game becomes a competition between different users.

Bibliography

[1] Lindsay Bassett. *Introduction to JavaScript Object Notation*. First. Gravenstein Highway North, Sebastopol, CA, USA: O'Reilly Media, Inc., 2015.

[2] Alan Cohen. *Prototype to Product*. First. Sebastopol, CA, USA: O'Reilly Media, Inc., 2015.

[3] Ted Hagos. *Learn Android Studio 4: Efficient Java-Based Android Apps Development*. Second. Manila, National Capital Region, Philippines: Apress, 2020.

[4] Manas Deb Prasen Palvankar Heidi Buelow Manoj Das and Meera Srinivasan. *Getting Started with Oracle BPM Suite 11gR1*. First. Birmingham, B27 6PA, UK: Packt Publishing, 2010.

[5] Bikesh Maharjan. *Puzzle game using Android MVVM Architecture*. Bachelor's thesis. Metropolia University of Applied Sciences, 2018.

[6] David Thomas. *DK Essential Managers: Improving Your Memory*. First. DK Publishing, 2007.

[7] Daphnee Lo Kah Yii. *Brain Workout, An Application for Enhancing Memory and Problem Solving Skills*. Bachelor's thesis. Universiti Teknologi PETRONAS, 2012.

Name: Sina Vähäkangas Matriculation number: 832026

Honesty disclaimer

I hereby affirm that I wrote this bachelor's thesis independently and that I did not use any other sources or tools than the ones specified.

Ulm, ...

Sina Vähäkangas