|  |  |
| --- | --- |
|  |  |
| Mobile Information Systems |  |
|  |  |
| Truth Estimation from Mobile Interaction |  |
|  |  |
|  | Written by  Jyotsna Gorle | Matr.Nr.: 120290  Christopher Heiden | Matr.Nr.: 120249 |
|  |  |
|  | Submitted to Prof. Echtler |
|  |  |
|  | Submission date: |

Contents

[1 Theoretical Part 1](#_Toc11231469)

[1.1 A Summary of the paper ” Veritaps: Truth Estimation from Mobile Interaction” 1](#_Toc11231470)

[1.2 Focused part of the paper 1](#_Toc11231471)

[2 Practical Part 1](#_Toc11231472)

[2.1 Differences between paper code 1](#_Toc11231473)

[2.2 Important code snippets 1](#_Toc11231474)

[2.2.1 Transitions 1](#_Toc11231475)

[2.2.2 AI programming 2](#_Toc11231476)

[2.3 Code Structure 2](#_Toc11231477)

[3 Literature 4](#_Toc11231478)

[4 Annex 5](#_Toc11231479)

# Theoretical Part

## A Summary of the paper ”Veritaps: Truth Estimation from Mobile Interaction”

In the paper “Veritaps: Truth Estimation from Mobile Interaction” the authors Aske Mottelson, Jarrod Knibbe, and Kasper Hornbæk thought about ways to find out if someone is lying or not. They argue that it is very difficult to find out by digital communication; however, they programmed three applications that can be used by a phone to find out if possible.

They argue that people use the phone very often, so it is to find out differences in their habits if they don’t act like normal. Therefore, time differences for answering questions and gestures can be used to detect a specific behavior. Because the frequently used of a phone, an approximation can be made if someone is lying or not.

The first study, they made is called ”simple lies”. The app had the goal to find out if someone is lying through touch interaction and if it is possible to separate the honesty and dishonesty. Therefore, participants have been asked to lie or to tell the truth about the background color. Every participant had to do this 192 times, 64 times with truth conditions, 64 times with directly lying and the other 64 times the participant could make their own choice. They either had to respond by tapping or dragging. The order was random.

In the other study participants had to do 10 trials. Still, that was only the proposal. In this study, an AI (artificial intelligence) has to communicate with the user. This happens with text input fields that can be used by the participant. This AI either accept the answer of the user or not. This choice is based on a too low payment respond or too unfair or passed process. In every trial, the amount of US cents varies by 25 and 99. With that, the user can choose if he/she wants to lie about the money and also how much he/she wants to share. With a huge dataset, it was possible to calculate if someone is lying or not.

The last app has been named as “Yatzy Game”. In this study, the authors tried to find out if it is possible to notice spontaneous lying by reference dishonesty. Therefore, the participants play a dice game on their phone. The user was rewarded based on their reported score, thereby making lying profitable. Still, users didn’t have to lie. However, to encourage lying, participants are rewarded relative to the reported outcomes. 12 rounds had to be played by the user. of rolls with five dice. In a list of possible combinations, the user has to choose a possible combination and its score. In total the participant has to have 12 combinations at the end; one for each round. The clue is that every combination could only be used once. If the final dice of around did not equate to a combination, then any combination could be selected and a score of 0 should be entered. The game recorded both the participants’ actual score and their reported score.

In all of these games, it was possible to find out if a user has been lied or not after a certian time period. This means it is possible to find out if a user is lying or not.

## Focused part of the paper

The group of J.Gorle and C.Heiden has been focusing on the third study. The first and second study isn’t the focus of this project. The dice game (see A Summary of the paper ”Veritaps: Truth Estimation from Mobile Interaction”) has been implemented with some slight changes in the design.

At first, the app has been structured, so now the application has an opening screen, an information screen, and the game itself with its two screens. On these screens, buttons aren’t as often included as in the screens of the app that is presented in the paper. Instead of swipe gestures have been used to navigate though the application. So, in navigation gestures have been used and for acceptance, buttons has been used.

The rest has not been changed and is part of the application.

# Practical Part

## Differences between paper code

In the project, we are just working on an implementation of the third task (see “Focused part of the paper”). The group decided to change the design to make it more appealing (see Figure 2, 3, 4, 5, 6). Furthermore, they added some screens to the original game app, so the user can understand the app without any explanation. Therefore, an information screen has been added that explains the game itself and how it works. Sure, it does not tell the user that the backend stores the answering time.

## Important code snippets

### Transitions

The most difficult part in programming the interface was to find out how to animate the transitions also and change the screens. Therefore, the group had to create an ”anim folder”. There, XML files had to be created with a set-tag and a translate-tag. In these tags programmers can implement transition aspects like the duration of the transition and the x- and y-coordinates where the transition starts and where it ends.

To use the transition, programmers have to define an FragmentTransaction. With that, you can replace the older screen and also set up a costume animation (see *Code 1*).

|  |  |  |
| --- | --- | --- |
|  | SecondScreen fragment = SecondScreen.newInstance(); FragmentManager fragmentManager = getSupportFragmentManager(); FragmentTransaction transaction = fragmentManager.beginTransaction(); transaction.setCustomAnimations(R.anim.enter\_from\_bottom, R.anim.exit\_from\_bottom); transaction.addToBackStack(null); transaction.add(R.id.fragment\_container, fragment, "SECOND\_SCREEN").commit(); |  |

Code 1 shows how to change screens in an app and also how to call the self-generated transitions.

### AI programming

## Code Structure

The code structure for the whole application can be found in Figure 1.

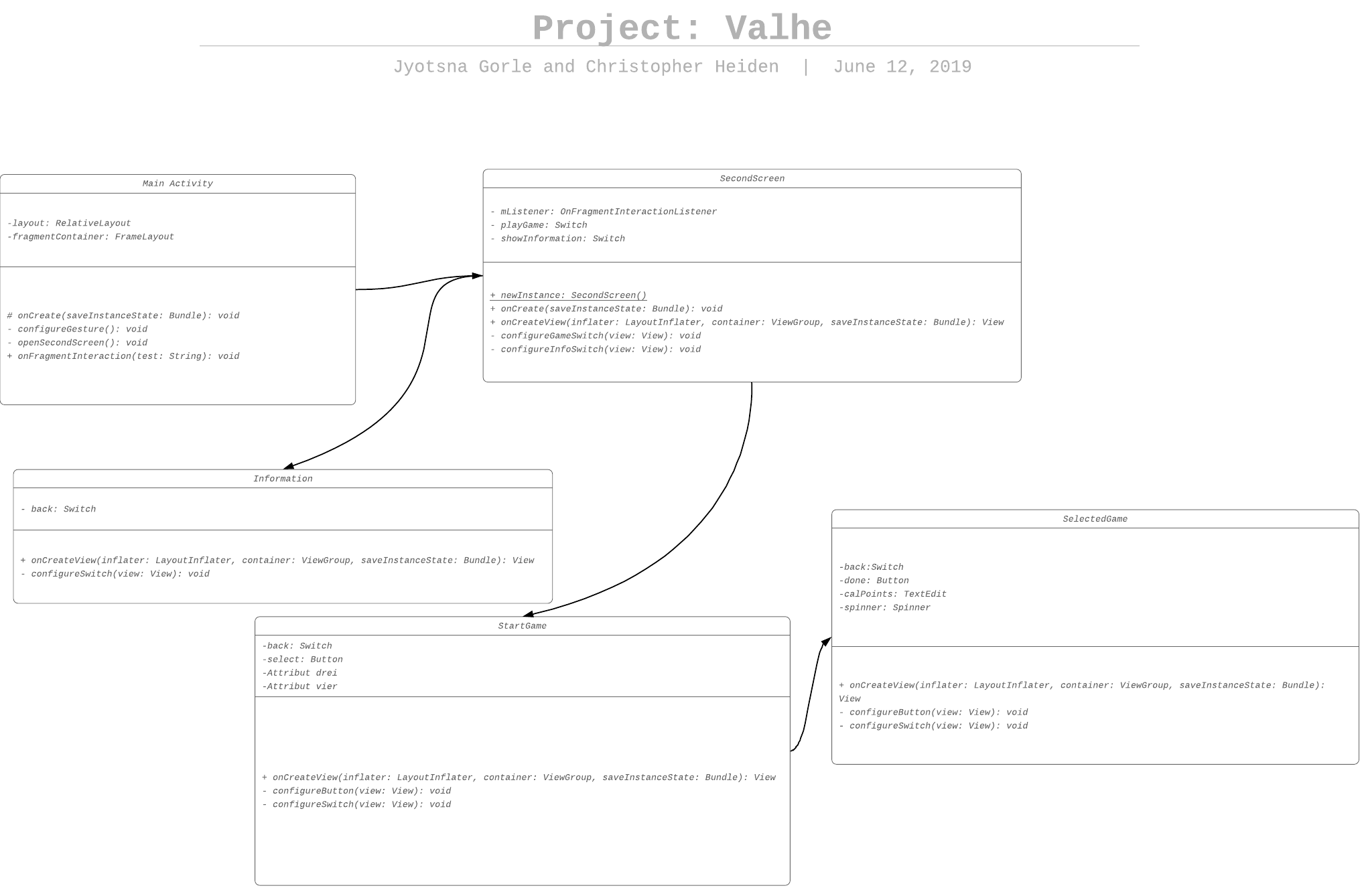


Figure 1 shows the code structure of the project.

# Literature

# Annex

|  |  |
| --- | --- |
|  |  |
| Figure 2 shows the first screen of the app which acts as an opening screen. | Figure 3 shows the second screen in which the user can either start the game or find out how the game works. |
|  |  |
| Figure 4 shows the information screen that the user can open to learning how the game works. | Figure 5 shows the rolling dice screen and the selecting dices part of the game. |
|  |  |
| Figure 6 shows the last screen of the app in which the user can choose the combination and the points he/she gets. |  |