# DinDrikkeLek

(YourDrinkingGame)

MOB3000R

Group 8.

Leonard Solvik Lisøy 141

Hasan Hasanovic 143

## Table of contents

## 1. Introduction

We have developed a mobile application which we decided to call DinDrikkeLek (YourDrinkingGame). The purpose of the app is to give people a drinking game where the user can add its own games and rules. The goal is to eliminate the need of several drinking game applications, and to be able to add a personal touch to the drinking game.

## 2. Problem

The problem with drinking games is that you usually need a deck of cards to play. Although there are many drinking games through apps today, they are usually limited to specific types of games. Furthermore, you can't change the games within the applications, nor add rules. DinDrikkeLek tries to solve this issue by introducing an application in which you can customize.

### 2.1 Competitors and innovation

DinDrikkeLek is a unique application in this form of drinking game. However, there are similar applications. We choose to compare us to Børst(Google Play, 2018), because our application is greatly inspired by it. Børst is similar to our app because it also gives the user random "rules" or "twists" as you may call it, in which the people participating in the game have to follow. Børst also offers categories of drinking games, depending on the mood or participants. Børst also has paywalls in which you must pay for additional games or rules. This is where our application excels.

Our compeditor Børst doesn't let you add own rules to the games, nor add any new games. Our application lets you do exactly that. If you've used Børst for a long time, you may start to notice the same rules to the games and the game may become dull. With our app you can keep adding games and rules, and therefore the user shouldn't need to ever use another drinking-game application.

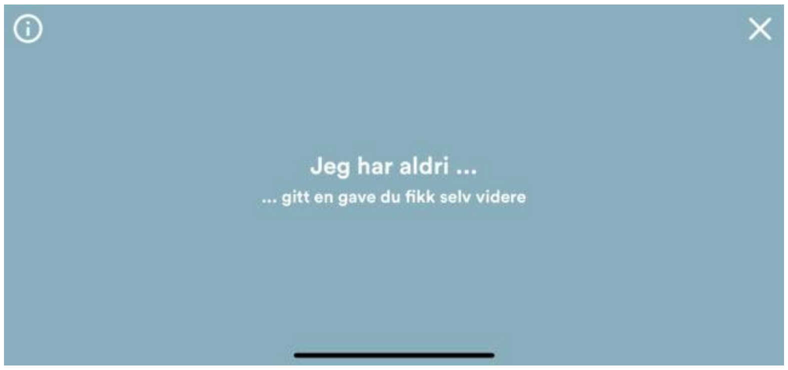

Jeg har aldri ...
... gitt en gave du fikk selv videre

*Figure 2.1.1 An instance of a Børst game*

*Figure 2.1.2 Børst logo*

## 3. Description

We chose to design an overview of our application to illustrate the structure.

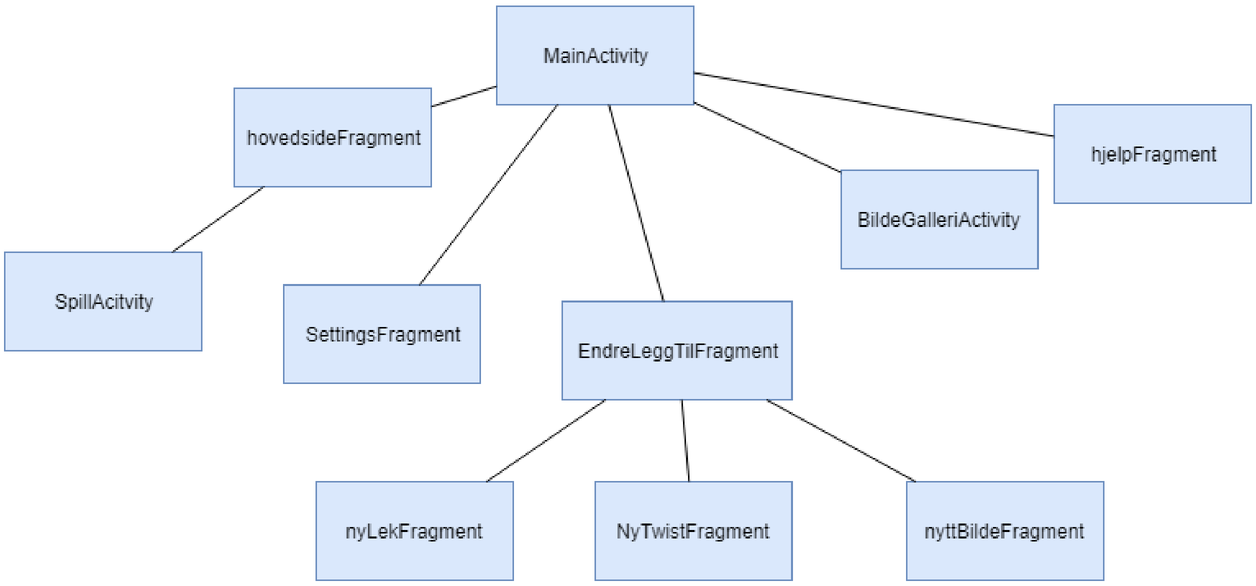


*Figure 3.1 Application diagram*

Other javaclasses:
Bilde
bildeAdapter
BildeBaseColumns
dbHandler

## “Play”

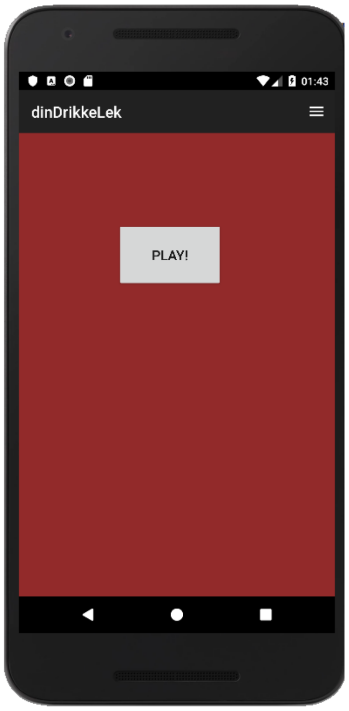


*Figure 3.2 fragment_hovedside.xml*

When the user first starts the application the play button will be the first thing the users sees. Here you can start playing the game by pushing the play button. This is the “main page” of the application. Here the user also has the option to use the navigationmenu from the top right button in the toolbar.

## “The Game”

This is the main function of our application, the game itself. Here the user is prompted to touch anywhere on the screen to start playing. When the user touches the screen, a random game will be chosen. In addition to choosing a game, the app also chooses a random rule linked to that specific game. The games and rules are loaded from a database within the app, with rules and games the user may have entered himself.

Every time the user clicks anywhere on the screen, the application fetches a new random game with a random twist. If the user has uploaded an image to the rule (or twist), the game will also display the image connected to the rule.

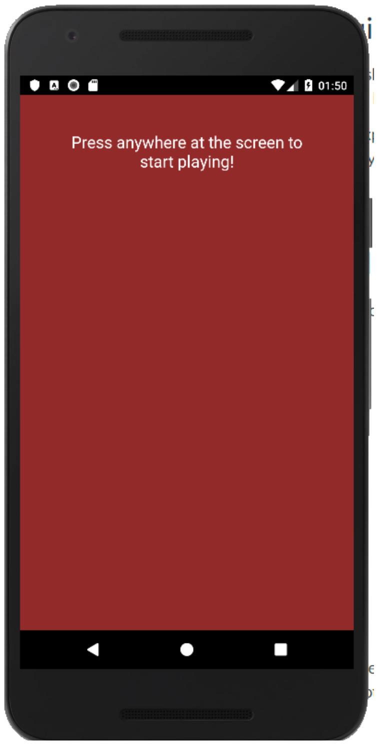
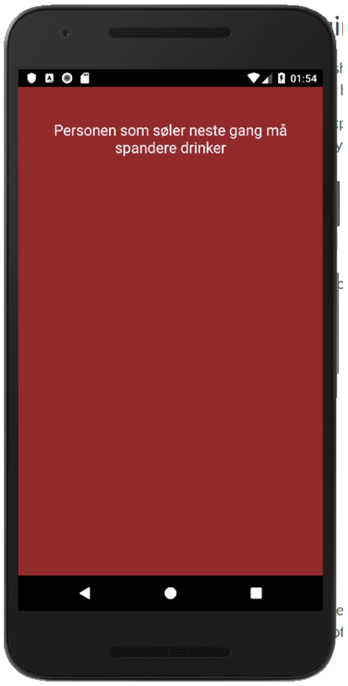


*Figure 3.3 Activity_spill.xml*



In this case a random game has been chosen ("The person who"). Alongside the game a rule has also been fetched ("The next person who spills their drink must order drinks for the rest"). When the user touches the screen again, the rule (or twist) disappear and a new twist occurs. If the twist had an image, the image will also disappear.

*Figure 3.4 «The next person who spills their drink must order drinks for the rest"*

## "NavigationMenu"

By clicking the hamburger button in the toolbar, the user can access the navigation drawer. From here the user can navigate to certain aspects of the application. The user may also swipe from left to right to access the menu. Let's say the user chooses to click "Add/delete"…

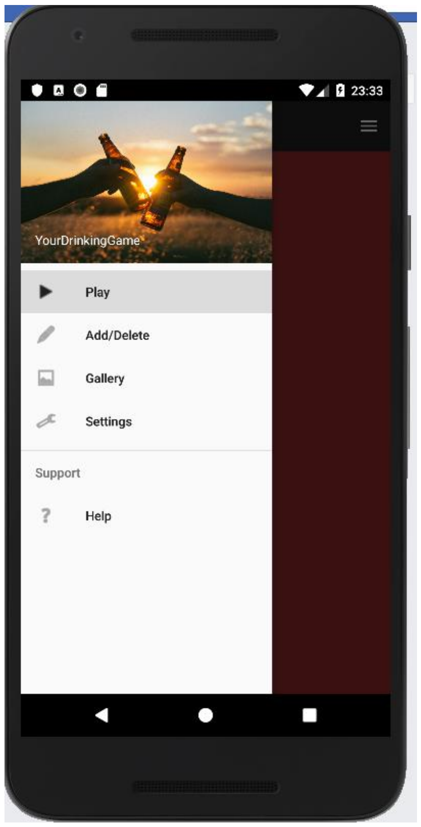
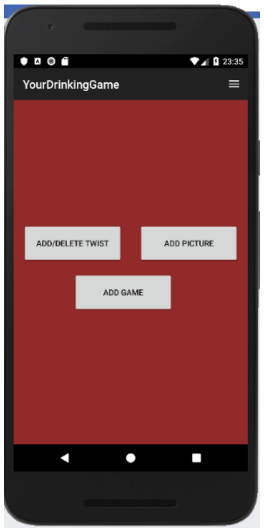


*Figure 3.5 layout/menu/menu_nav.xml*

## “Add/delete”



Here the user can choose what option he may prefer. If he wants to add a new game, twist(rule) or add a picture to an existing twist.

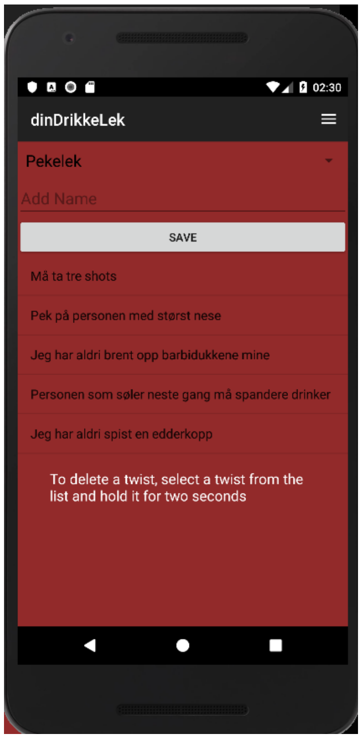
Let’s say the user selects “Add/delete twist”…

*Figure 3.6 fragment_endre_legg_til.xml*

## “Add/delete Twists”

In this “window” the user may choose a game from the spinner and add a new rule (or twist) to it. When the user clicks on the spinner, the spinner drops down a list of all the games that is stored in the database. The user may then input the new twist, click save, and the twist is now added to the game. Now it may appear when you play the game.

In this part of the app the application also provides you a list of existing twists stored in the app(database). If the user wants to delete a rule(twist), he may press a twist from the ListView and hold it for a few seconds. He is then prompted if he wants to delete this twist.



Må ta tre shots

Pek på personen med størst nese

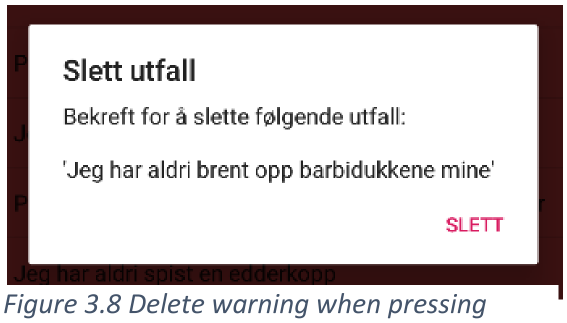
Jeg har aldri brent opp barbidukkene mine

Personen som søler neste gang må spandere drinker

Jeg har aldri spist en edderkopp

To delete a twist, select a twist from the list and hold it for two seconds

*Figure 3.7 fragment_ny_twist.xml*



Slett utfall

Bekreft for å slette følgende utfall:

'Jeg har aldri brent opp barbidukkene mine'

SLETT

Jeg har aldri spist en edderkopp

*Figure 3.8 Delete warning when pressing down listitem*

## "Add Picture"

By clicking "Add Picture" the user may add a picture to his/hers twists.

1. The user chooses a picture from either their phone gallery or camera.
2. The user inputs a name for the picture
3. The user selects a twist to connect the image to from the spinner.
4. The user clicks save
5. The picture is now saved.

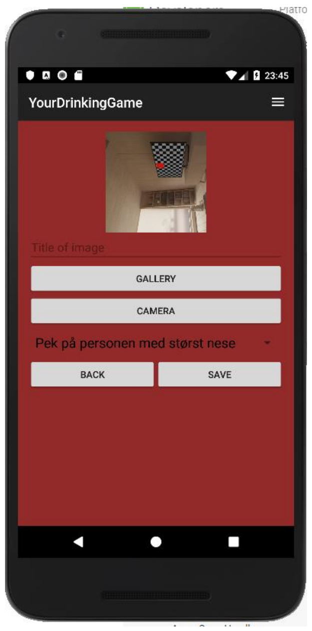
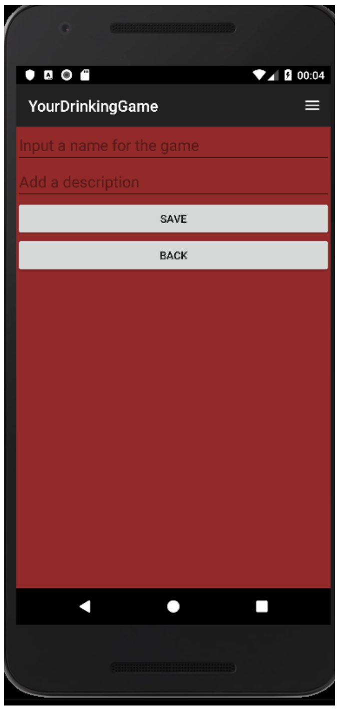


*Figure 3.9 fragment_nytt_bilde.xml*

## "Add Game"



Here the user may add a new game if he/she would like to. The "game" will however not appear in the drinking game until the user adds a twist to it.

1. The user enters a name of the game
2. Enters a description
3. Clicks save
4. The game is now saved

*Figure 3.10 fragment_ny_lek.xml*

## “Settings”

When the user navigates from the navigation menu he may choose “settings”. Here the user can delete all pictures, twists, and games if he wants to. The user simply clicks on the option he wants to delete. By deleting all games, all twists connected to that game will also be deleted. If the user clicks “back”, he will be sent back to the “main page”.

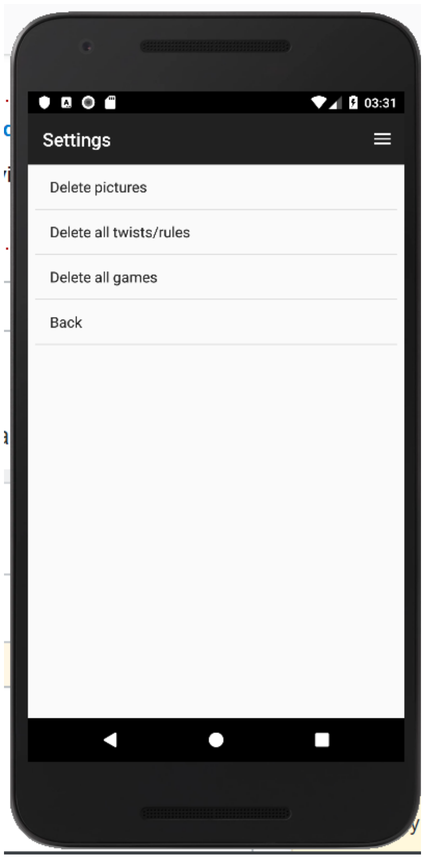


*Figure 3.11 fragment_settings.xml*

## “Gallery”

When the user selects “Gallery” from the navigation menu, he will be redirected to all the picture he has added. Here he can scroll through all images stored in the app. All pictures are shown in a GridView and utilizes an adapter class called bildeAdapter.java.

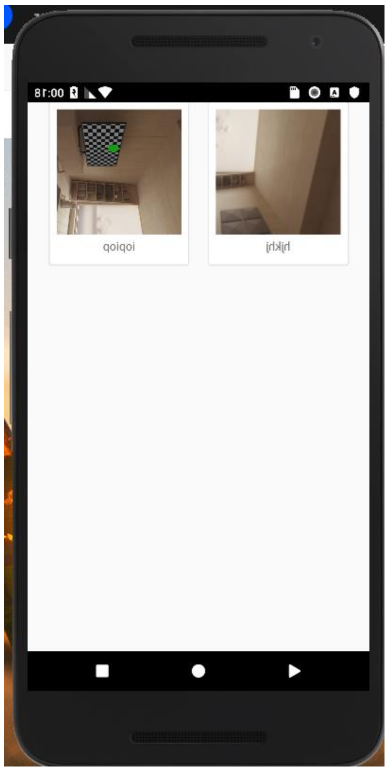
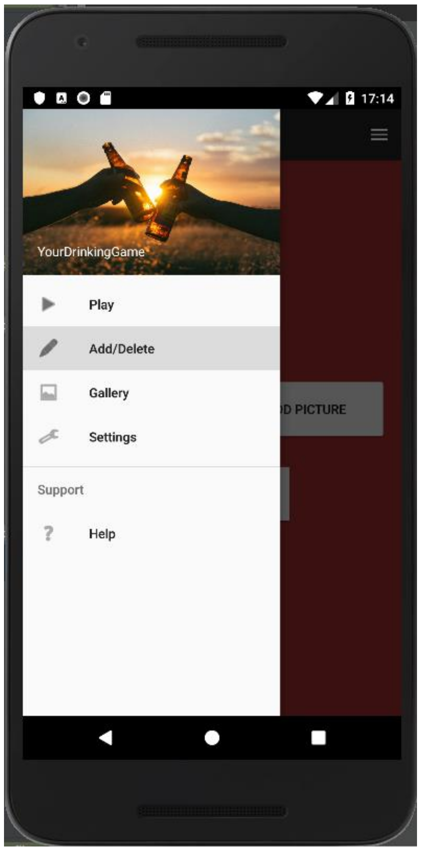


*Figure 3.12 activity_bilde_galleri.xml*

## 4. Component Features

DinDrikkeLek has four different features in terms of programming. SQLLite, good navigation, use of fragments and multimedia components (camera, image upload).

The database is used on several places in the app. Including “The Game”, “Settings”, “Add/Delete Twists”, “Add Picture”, “Add Game” and in the “The Game” itself. Furthermore, the application has an app bar in the top. Here the user can navigate through the application. When you click on the hamburger button the menu appears on the left.



The toolbar is available throughout the whole app except in “The Game” and “Gallery”. This is because we didn’t want a toolbar in them. But it’s no problem. The user only must click the back-button to get back to the ‘main page’.

*Figure 4.2 Layout/toolbar.xml*

*Figure 4.1 layout/menu/menu_nav.xml (navigation drawer)*

Our application uses The Design Support Library to enable material designs. This includes Material color themes, app bar support and fragments. We also implemented the use of Constraintlayouts on several places for better organizing of views. The Constraintlayout is much more flexible than other layouts. We also found it often easier to design our layouts visually.

We used a lot of fragments in the application because it made the navigation drawer feel better when clicking a menu item. This way the navigation drawer doesn't suddenly disappear instantly when the user clicks on an item, as it would have with activities.

The application also provides feedback on invalid actions. For example, if no twists are selected when the user tries to upload a picture, the app will show an error message. We used fragments in the "Play" window, "Add/delete", "Add/Delete twists", "Add games", "Add Picture" and "Help". Therefore, we also had to implement a method in the main activity to make the back-button functional between fragments.

## 5. Process and Contribution

Our group decided to use the waterfall model as a development method. This is because we we were only two members, and communication, organizing and collaboration was therefore easier to handle.

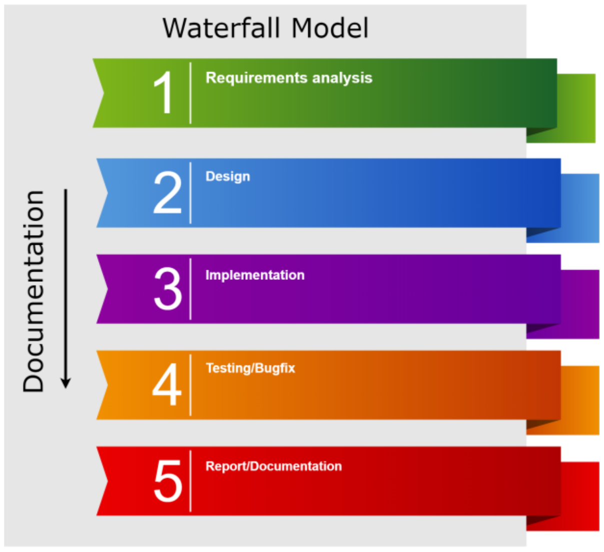
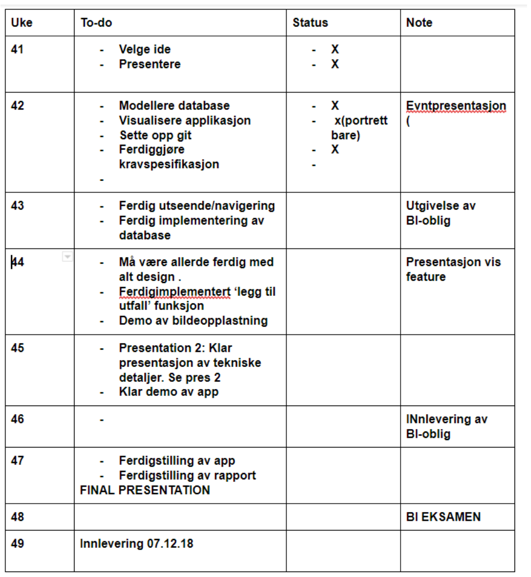


*Figure 5.1 Traditional Waterfall Model*



| Uke | To-do | Status | Note |
|---|---|---|---|
| 41 | - Velge ide<br>- Presentere | - X<br>- X | |
| 42 | - Modellere database<br>- Visualisere applikasjon<br>- Sette opp git<br>- Ferdiggjøre kravspesifikasjon<br>- | - X<br>- x(portrett bare)<br>- X<br>- | Evntpresentasjon ( |
| 43 | - Ferdig utseende/navigering<br>- Ferdig implementering av database | | Utgivelse av BI-oblig |
| 44 | - Må være allerde ferdig med alt design .<br>- Ferdigimplementert 'legg til utfall' funksjon<br>- Demo av bildeopplastning | | Presentasjon vis feature |
| 45 | - Presentation 2: Klar presentasjon av tekniske detaljer. Se pres 2<br>- Klar demo av app | | |
| 46 | - | | INnlevering av BI-oblig |
| 47 | - Ferdigstilling av app<br>- Ferdigstilling av rapport<br>FINAL PRESENTATION | | |
| 48 | | | BI EKSAMEN |
| 49 | Innlevering 07.12.18 | | |

*Figure 5.2 Week plan for the project*

To organize our tasks, we simply made a table and documented the deadlines of certain tasks. Since we we're only two people we mainly worked together. Therefore, we didn't need to assign many different tasks to different members.

To collaborate on the application, we used GitLab in Android Studio to efficiently share changes and control new versions of the app.

Below is an illustration over our schedule. We didn't follow is as strictly as we had hoped for. Some tasks needed to be redone and group members didn't always have the time to stay on top of the schedule due to personal reasons and other school matters such as exams.

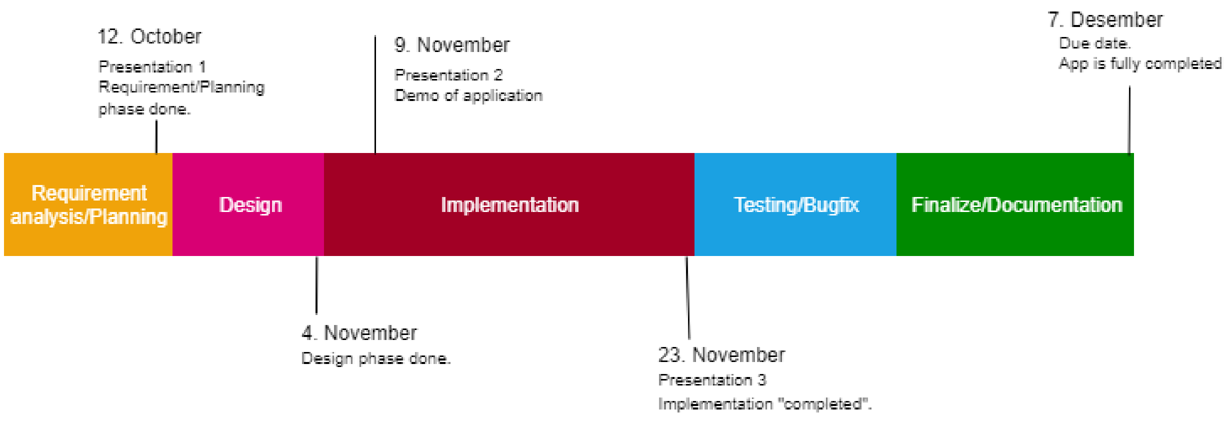


*Figure 5.3 Schedule*

We had weekly collaborations on Fridays and generally any day we both had time to do some work. The group discussed what had been done, how the progress was from the last time we spoke and what needed to be done until the next deadline, such as preparing for presentations or completing some piece of functionality. These meetings happened either in person or over Discord. The group also worked and discussed the project irregularly several times during the weekdays.

## 5.1 Contribution

Every member of the group contributed to all aspects of the application and project. Since the group only consisted of two members it was easy to include both members to the project tasks. All members also contributed on writing the report.

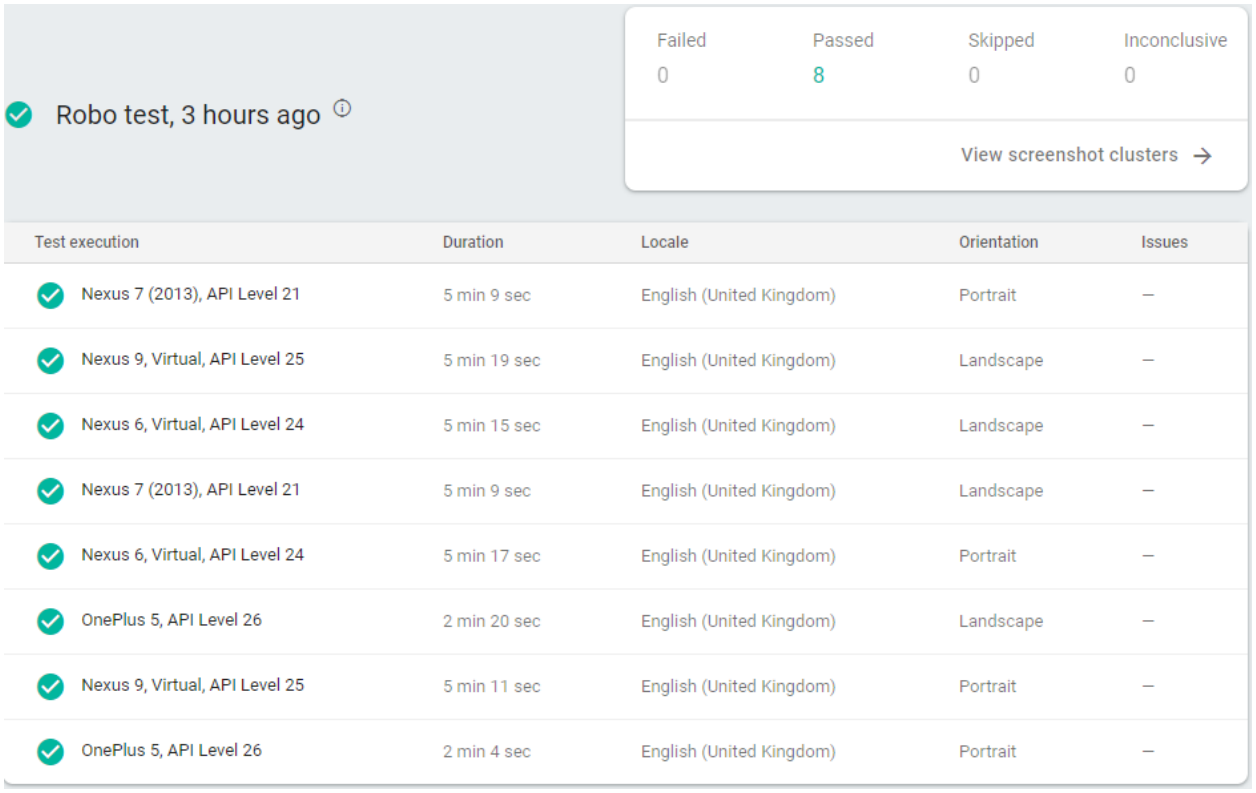
## 5.2 Testing

Testing is something we have done throughout the whole project, in varying degrees. In the implementation phase we tested continuously as small pieces of code were completed to make sure some part of a functionality works. We would for example work on one function, let's say adding a picture to the database, and see if the written code actually does what it is supposed to do and without any failure or crash.

As more and more functionality started falling into place, we downloaded some virtual devices with both newer and older API levels (mainly Lollipop 5.0 API level 21) and made sure everything worked as good as possible.

With emulators we have also done some testing with different screen sizes, tablets (such as the Nexus 9) and orientations. Downloading and selecting a bunch different deployment targets for the emulator in Android Studio isn't that efficient so we used the Test Lab made by Google Firebase(Firebase, 2018).

We had some trouble with Firebase to begin with. We would for example need to pay a subscription fee to do more extensive testing with a wider range of phones, tablets, API's and locales/languages. This is because the Test Lab in Firebase has a daily quota of 5 physical and 10 virtual device test executions per day. Never the less we were at least able to run some tests, as shown in *figure 5.2 Firebase Test Lab*.



✓ Robo test, 3 hours ago

| Failed | Passed | Skipped | Inconclusive |
|---|---|---|---|
| 0 | 8 | 0 | 0 |

View screenshot clusters →

| Test execution | Duration | Locale | Orientation | Issues |
|---|---|---|---|---|
| ✓ Nexus 7 (2013), API Level 21 | 5 min 9 sec | English (United Kingdom) | Portrait | – |
| ✓ Nexus 9, Virtual, API Level 25 | 5 min 19 sec | English (United Kingdom) | Landscape | – |
| ✓ Nexus 6, Virtual, API Level 24 | 5 min 15 sec | English (United Kingdom) | Landscape | – |
| ✓ Nexus 7 (2013), API Level 21 | 5 min 9 sec | English (United Kingdom) | Landscape | – |
| ✓ Nexus 6, Virtual, API Level 24 | 5 min 17 sec | English (United Kingdom) | Portrait | – |
| ✓ OnePlus 5, API Level 26 | 2 min 20 sec | English (United Kingdom) | Landscape | – |
| ✓ Nexus 9, Virtual, API Level 25 | 5 min 11 sec | English (United Kingdom) | Portrait | – |
| ✓ OnePlus 5, API Level 26 | 2 min 4 sec | English (United Kingdom) | Portrait | – |

*Figure 5.2.1 Firebase Test Lab, Robo Test 06.12.2018*

# 6. Technical Description

## 6.1 Screen sizes.

DinDrikkeLek supports normal screens as well as large screens. This includes phones and tablets. The application also supports landscape and portrait-mode for both small and large screens. When the user rotates their screen, an alternative layout for this orientation will be loaded. Alternative landscape layouts have been implemented in places we felt needed them. Additional alternative layout recourses have also been implemented for tablet devices.

## 6.2 SDK Tools

We used the SDK-versions 21-28 and chose to use the build tool version 28.0.3. We specified build tool version as 28.0.3 even though it is no longer necessary (Developers, 2018) because our tutor allowed us to use this version. Since the assignment specified that we should use a specific version, we chose to implement the property in the gradle build.

We didn’t use any other external SDK’s. We only used the internal SDK tools. These include “Android SDK Tools”, “Android SDK Build-Tools” and “Android SDK Platform-Tools”, as well as the “Support Repository”.

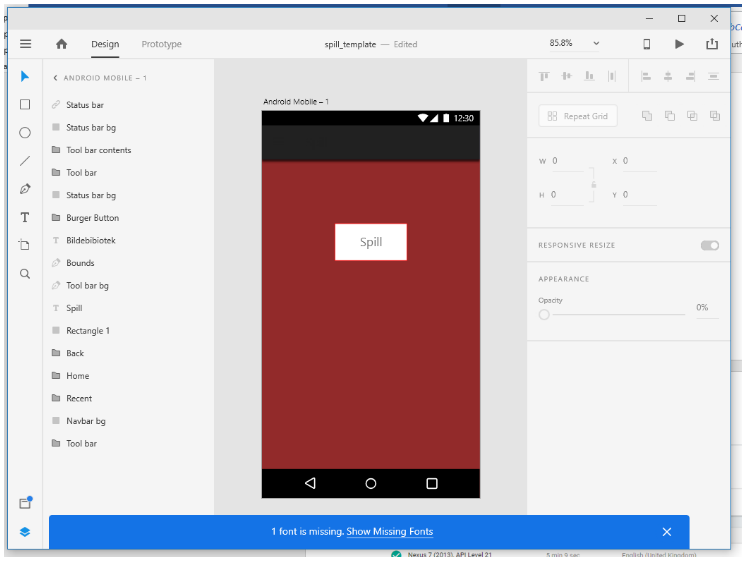
SDK Dependencies we used:

```
    implementation fileTree(dir: 'libs', include: ['*.jar'])

    implementation 'com.android.support:appcompat-v7:28.0.0'

    implementation 'com.android.support.constraint:constraint-layout:1.1.3'

    implementation 'com.android.support:support-v4:28.0.0'

    implementation 'com.android.support:design:28.0.0'

    testImplementation 'junit:junit:4.12'

    androidTestImplementation 'com.android.support.test:runner:1.0.2'

    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
```

These dependencies should be compatible with Android Studio version 3.2.1. After some testing they should also be compatible with SDK’s in range of 21-28.

## 6.3 Design Tools

In our planning phase we illustrated the application for ourselves before we started implementing. We used the tool Adobe XD to make illustrations of how we wanted the activities and fragment to look like.



## 6.4 Documentation

### Database

The database is not tied to several users. Since the application and database only runs on the user's phone, we saw no need to implement user registration to the database.

When you play the game, the database fetches a random game from table "Lek", and then finds a random twist from table "Utfall" that has a reference to that specific Lek. If a post in table "Utfall" has a foreign key to table "Bilde", the game will also load the picture tied to the twist. The foreign key reference to "Lek" in table "Utfall" is implemented with an "ON DELETE CASCADE". This way, when the user deletes a game it also deletes the twist linked to the game.

### Camera/Gallery Upload

When a user uploads an image through camera or phone gallery, the fragment will fetch the image and upload it to the database when the users clicks "Save", alongside with picture title.

## 7. Conclusion

DinDrikkeLek (YourDrinkingGame) is an excellent application for people who enjoy drinking games and wants to add a personal touch to them. With all members working hard and using their knowledge we've achieved an app we are proud of. Collaboration tools has made a serious positive impact on our development progress. By deciding early what to make and by making illustrations of the application in early stages we achieved a great workflow.  This is because we had already seen illustrations of what we we're trying to make.

For future implementation our project group would like to implement a feature to delete specific games and specific pictures. We would also like to implement a feature to share twists and games from the app to other people who use the application. This way, people can participate in expanding the twists and games within the app to make an even more diverse drinking game.

Even though there are similar apps out there that provides subgames with twists, no app lets you add your own personal touch to them. And that's why DinDrikkeLek stands out between them all.

## 8. References

Developers Android. (2018). Android Gradle plugin release notes. Retrieved from: https://developer.android.com/studio/releases/gradle-plugin#behavior_changes

Firebase (2018) Google Firebase . Retrieved from: https://firebase.google.com

Google Play(2018). *Børst.* Retrieved from: https://play.google.com/store/apps/details?id=com.borst&hl=en

Image from Navigation Drawer. Retrieved from: https://www.behance.net/gallery/63242635/graduation