**Grand Psychedelic Space Voyage Experiment**

**Grupp 23**

**Designdokument**

**V. 1.5**

**17-08-12**

**Handledare Gruppmedlemmar**

Farid Naisan Miran Amin

Nils Lindkvist

Oskar Lloyd

Petter Månsson

# Dokumenthistorik

|  |  |  |  |
| --- | --- | --- | --- |
| Datum | Version | Beskrivning | Författare |
| 17-04-06 | 0.1 | Uppdaterat sektion om Användningsfall | Petter Månsson |
| 17-04-12 | 1.0 | Skrivit de delar som saknats (syfte, systemdiagram, diagram samt användargränssnitt), rensat onödig text. | Miran Amin |
| 17-04-21 | 1.1 | Ändrat usecase diagram | Petter Månsson |
| 17-05-22 | 1.2 | Klassdiagram och punkter, fetstil, Uppdaterat klass och komponentdiagram | Oskar Lloyd  Miran Amin |
| 17-05-25 | 1.3 | Uppdaterat beskrivningen till komponentdiagrammet, lagt till medlemmar och handledare | Miran Amin |
| 17-05-25 | 1.4 | Uppdatera innehållsförteckning | Oskar Lloyd |
| 17-08-12 | 1.5 | Kompletterad Klassdiagram samt beskrivning för klassdiagrammet | Miran Amin |

**Innehåll**

[Designdokument 1](#_Toc483476185)

[Syfte 1](#_Toc483476186)

[Systemdiagram 2](#_Toc483476187)

[Användningsfallsdiagram 3](#_Toc483476188)

[Scenarion/Användningsfallsbeskrivningar 3](#_Toc483476189)

[Användargränssnitt 4](#_Toc483476190)

[Klassdiagram 5](#_Toc483476191)

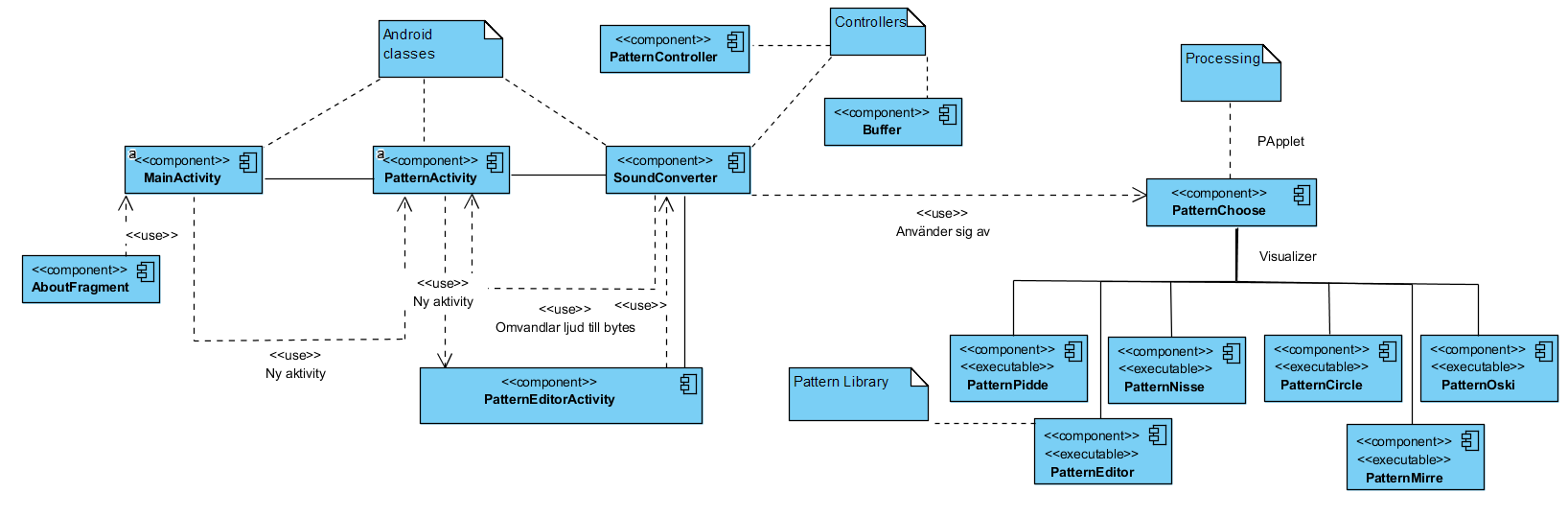
# Designdokument

# Syfte

Designdokumentet är avsedd att beskriva projektets system, hur komponenter förhåller sig till varandra, hur dessa används och vad resultatet av dessa kan innebära. Dokumentet innehåller ett komponentdiagram som visar, samt beskriver vilka klasser som används, klassernas syfte, vilka bibliotek som används samt hur systemet är uppbyggd i helhet. Klassdiagrammet visar klasserna mer detaljerat genom attribut och operationer där man kan få en bättre översikt om hur komponenterna hänger ihop samt klassernas relation till varandra genom associationer

Ett usecase-diagram används för att visa vilka funktioner användaren kan använda sig av när samtliga av kraven för applikationen uppfyllts. Användargränssnittet visar hur applikationen kan komma sig att se ut i slutändan.

# Systemdiagram



**Android Klasser:**

MainActivity: Representerar Applikationens första aktivitet. När programmet startas så är det MainActivity klassen som visas och där väljer användaren vilken funktion som skall köras. Klassens syfte är applikationens startsida.

PatternActivity: Representerar klassens andra aktivitet. Här visas den funktion som väljs i MainActivity. Är alltså klassen som visar applikationens mönster i fönster.

SoundConverter: Omvandlar ljud till bytes för det ljud som spelas upp i mobilen. Syftet med klassen är satt sedan använda sig av den omvandlade data till att rita upp mönstren.

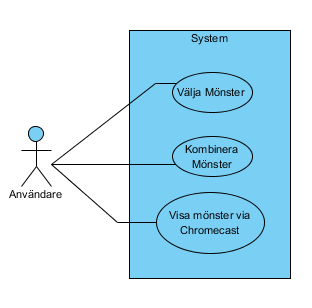
PatternChoose: Är klassen som ritar mönstren som vissas i PatternActivity. Valet av mönster/pattern görs alltså i MainActivity.

Pattern1-6: Är mönstrena som avändaren väljer i MainActivity. Syftet med Pattern1-4 är att användaren kan välja mellan 1-6 olika mönster som ska visas i PatternActivity.

PatternEditorActivity: En aktivitet som tillåter olika kombinationer av mönster.

AboutFragment: Visar en backstory från MainActivity.

# Användningsfallsdiagram



## Scenarion/Användningsfallsbeskrivningar

Var användningsfall beskrivs nedan med en text som förklarar vad en avändare kan förväntas göra i varje fall.

### Välja Mönster

Användare kan via produkten välja ett mönster som visas i takt till musik i ett fönster. Musik startas utan för produkten. Användaren kan sen fritt byta mönster som då börjar reagera på musiken och visas i fönstret.

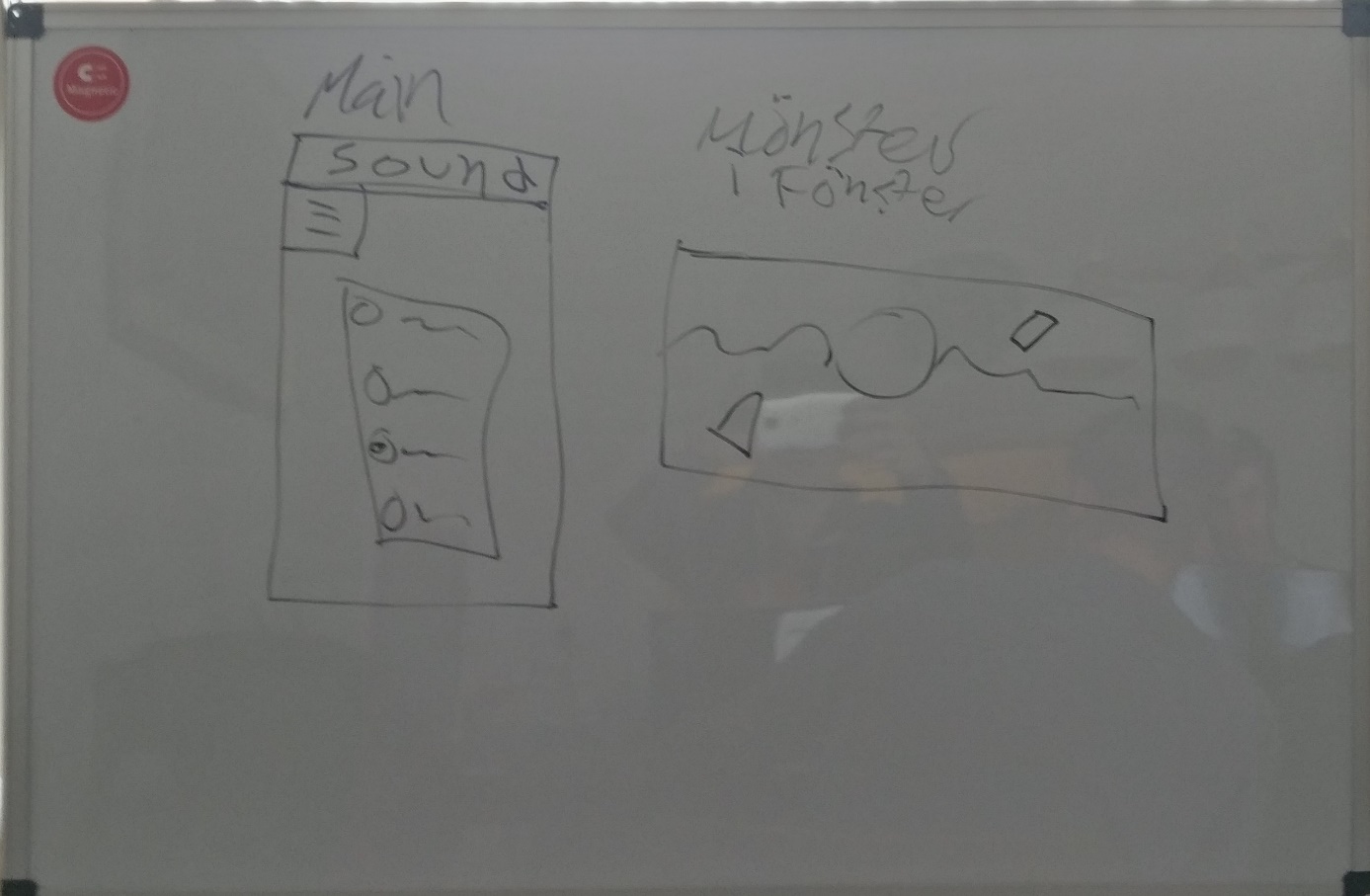
### Kombinera Mönster

Användaren kan välja att visa ett eget komponerat mönster. Detta mönster skapas genom att användaren via ett mönsterbibliotek kan kombinera mönster som reagerar på musiken. Ifall ett för inställt mönster redan visas kommer detta ersättas med det egen komponerade mönstret.

### Visa mönster via Chromecast

Användaren kan via produkten skicka bilden till en tv som har en chromecast inkopplad. Detta möjligör så att mönstret visas på en större skärm.

# Användargränssnitt



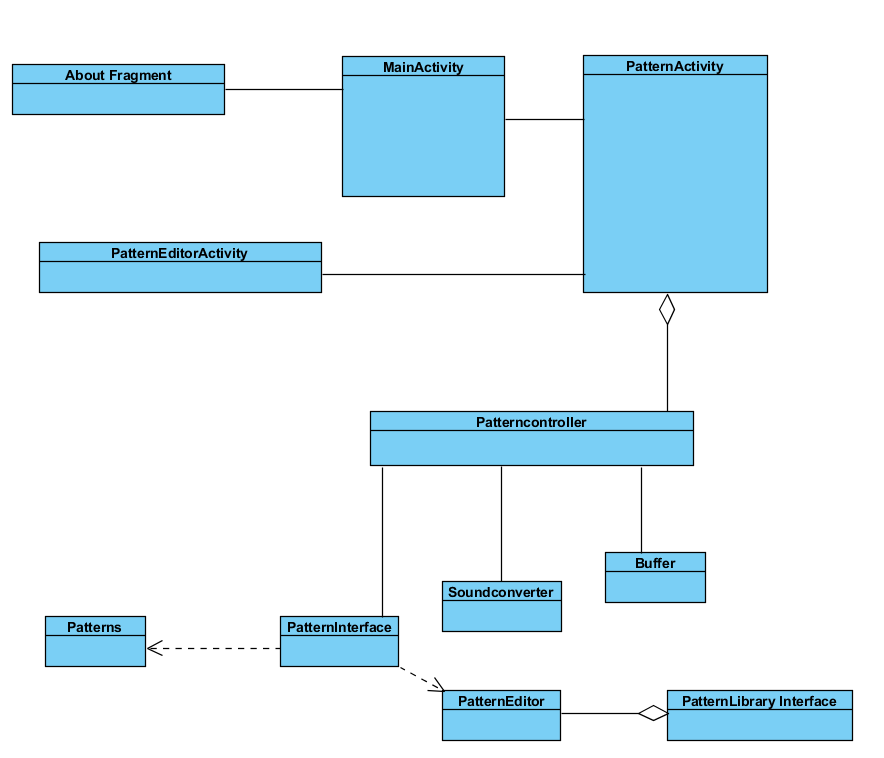
*Första skissen på UI. Första skissen visar hur UI ser ut vid val av mönster. Andra skissen visar hur UI ser ut när ett mönster visas.*

**UI:**

Visar en startsida med en dropdown meny där man kan välja olika funktioner. Detta kan vara exempelvis olika mönster, kasta till tv:n genom chromecast eller kombinera olika mönster tillsammans.

Den andra liggande sidan visar mönstret som skapas när musik spelas. Detta mönster varierar beroende på vad användarens inställningar.

# Klassdiagram



**Beskrivning av klassdiagram:**

Klassdiagrammet visar vilka attribut och operationer klasserna innehåller, hur klasserna relaterar och förhåller sig till varandra samt vilka bibliotek som ärvs.

AboutFragment klassen är en informationsruta om projektet som man endast kan komma åt genom MainActivity klassen. Tar man bort AboutFragment klassen så påverkas inget annat i programmet därav en enkel association.

MainActivity, PatternActivity och PatterneditorActivity är programmets 3 gui klasser som ritar upp det som finns och det som sker på skärmen. Man kommer åt PatternActivity genom Mainactivity och PatternEditorActivity genom PatternEditor, men dessa har fortfarande en enkel association mellan varandra eftersom att de annars inte påverkar varandra systemmässigt. De resterande klasserna. PatternController ritar upp alla patterns/mönster i PatternActivity och därav en komposition då även om man tar bort PatternActivity så körs mönstren i bakgrunden. Resterande klasser bygger upp logiken genom PatternLibrartInterface klassen medans PatternInterface generealiserar Patterns. I Patterncontroller klassen med en buffer och soundconverter.