Design Documentation – Free choice group

HomeDork – Interactive Smart House

Revision History

Name	Associated Letter
Fanny Söderlund	A
Malek Alabed	B
Nishat Jahan	C
Suzanne Zomer	D
Ismail Eyamba	E

Date	Version	Description	Author
22/09/2021	1.0.0	Initial discussion of design	A, B, C, D
06/10/2021	1.1.0	Design of mock lab	A, B, C, D
20/10/2021	1.2.0	Additions of figures 2 and 3	B
21/10/2021	1.2.1	Addition of D4 and complementing figure 4	A
23/10/2021	1.2.2	Rearranging text, check spelling and details	A, B, C, D, E
27/10/2021	1.3.0	Grammar revised	E
14/11/2021	1.3.1	Changes in document formatting such as versioning, tables, and titles according to group standards.	A
14/11/2021	1.3.2	Addition of D5, table of figures, and designs related to R1 and R8	A
15/11/2021	1.3.3	Addition of designs related to R2 and R3	B, C, D
4/12/2021	1.4.0	Addition of designs related to R6, R8 and R9	A, B, C, D, E
6/12/2021	1.4.1	Addition of diagrams and design to R4	D
21/12/2021	1.5.0	Addition to D4 and D5: personalised status commands and high contrast	D
21/12/2021	1.5.1	Addition to text in D4, and D5, and addition of Personalised status command use case, and addition of High contrast ratio figure	B
5/1/2022	1.5.2	Format changes, addition of appendix	A, B, C, D, E

Design item List

Design name	Requirements related	Priority
D1. Client android app environment	R1, R2, R3	Essential
D2. Client web app environment	R1, R2, R3	Essential
D3. Server/API connection	R2, R4, R5, R6, R8	Essential
D4. Home page designs	R4, R5, R7, R8, R9	Desirable
D5. Settings designs	R1, R2, R3, R6, R10	Desirable

Design Item Descriptions

D1.

A collection of the designs for the requirements relating to the android client side of the system.

Mocked environment
The mocked environment allows the features to be developed encapsulated without outside interaction. The mocked client app is a simple android app that contains the necessary components for the related requirements. The mocked environment is created using Android Studio using mainly Java to integrate easier with the other subgroups. As seen in *figure 1,* the mocked environment aims to be simple with a few components to make it easy for us to develop our features in it.

D2.

A collection of the designs for the requirements relating to the web client side of the system.

Mocked environment
The mocked environment allows the features to be developed encapsulated without outside interaction. The mocked client app is a simple web app that contains the necessary components for the related requirements. *Figure 2* shows the shell of the mock environment of the web client. *Figure 3* shows the connection of the mocked web client to the sever, whether a mocked API call or the actual system server.

D3.

A collection of the designs for the requirements relating to the server of the system. Initially, for testing and developing, the mocked environment will be used, but eventually the implementations will migrate to the actual server/API.

Mocked environment
The mocked environment allows the features to be developed and encapsulated without outside interaction. The mocked server allows for testing and implementing of features not yet handled by the current system.

D4.

A collection of the designs for the requirements related to the home page, meaning all requirements that are features that the user does not toggle on and off. These features are constant to the system much like the feature to turn on and off different devices.

R2 – Voice commands

The voice command feature is in the navigation bar on both clients and allows the user to either navigate through the system or control the devices. *Figures 11-14* shows the diagrams for this feature.

R4 – Personalized status commands

This feature is displayed on the home page where the user can set different command buttons that perform different actions set by the user. More is discussed in our requirements document under [R4].
For designing and implementing this feature, *figure 5* have been made. This is not the definite look of the system, but it gives an understanding of the requirement itself and allows us as a group to explain it better to other subgroups and be clear within the group how the feature should function.

Web implementation:
The status command feature consists of two JSP pages. The first page displays the buttons each with their custom title and connected devices. The user can click on each of these buttons to activate or deactivate them, controlling devices in their home.
When a new button shall be created, the user shall be redirected to another JSP page where all devices connected to the system, information retrieved from the server, shall be displayed. Here the user can select devices that will be added to this custom status command button, once created the button will be available to see in the first JSP page. *Figure 6* displays the design and *figure 7* the implementation for the features on the web.

Android implementation:
The Personalized status command works similarly on the android as it does on the web. What was done inside the Android project to make this possible was add 2 new adapter classes, “Device Adapter” and “Mood Adapter”, and implementing a new model called “Mood”, and we added 4 new activities inside the signup package, and these 4 activities allow a user to add a mood, edit a mood, delete a mood, and keep up with the current moods. and the design aspect of the android was implements very similarly to the design of the web.

R5 – Scheduled commands

The scheduled commands feature can be accessed through anywhere in the system. The user can add personal small notes or control the devices on a schedule through here.

D5.

A collection of the designs for the requirements related to the settings panel, meaning all requirements that are features that the user can toggle on and off.

R1 - Haptic vibration

Figure 8 shows the class diagram of how the settings pane is related to its xml files for the android client. By using the special “root_preferences.xml” in the xml folder with a PreferencesPane, the changes made in that view will be global to the whole application.
Figure 9 shows a use case diagram of a user turning on and off the setting for haptic vibration.
Figure 10 shows a state diagram of the states the application takes when the haptic vibration setting is turned on or off.

R3 – High contrast

High contrast is one of a few features that are accessible through the settings pane, both in android and web.

Android Implementation:
The implementation for android was made using themes. A theme has been created with custom colors suitable for people with impaired vision, making sure that the contrast ratio maintains a high value throughout the application.
When a new activity is being loaded the program checks for the variable set by the settings pane and applies the correct theme. This is either the standard white theme or high contrast theme. Below see the figure 15 and 16 corresponding to this feature.

Web Implementation:
The implementation for the web side of the feature was done very similarly to the android side, a theme is implemented and on that new theme we insert very specific colors that were picked by a website that had the highest high contrast rating(*Figure 18*) and a button was introduced to the top right of the page that when pressed, the theme with the high contrast color is picked.

R6 – Magnifying zoom

The zoom feature allows for users with poor eyesight to use the system easier.

R7 – Bliss expressions

The bliss expressions are integrated into the feature ***R4 – Personalized status commands*** in every mood button.

R8 – Game

The game is a simple JavaScript game that uses mainly the browser's engine to play. The game saves your local high score locally on the device and your favorite character. *Figures 20* shows the class diagram of this feature and *figure 21* shows the use case diagram.

R9 – Disco mode

Disco mode simulates blinking-colored lights inside the client application that can be further developed into actual lightbulbs. *Figures 22-25* shows the diagram related to this feature.

Appendix

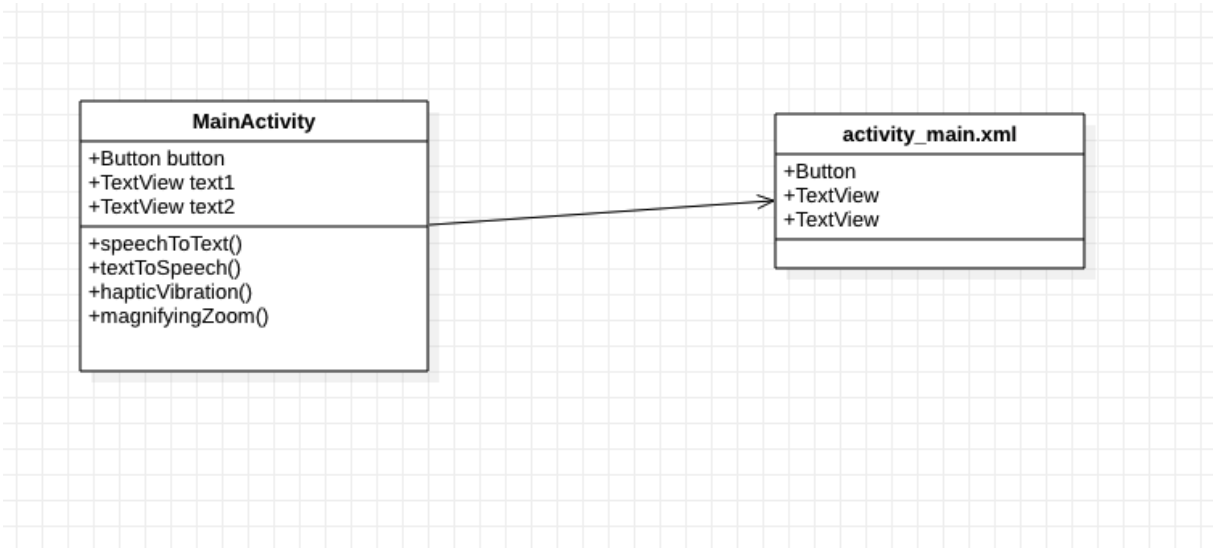

Figure 1 - Class diagram of mock client android app

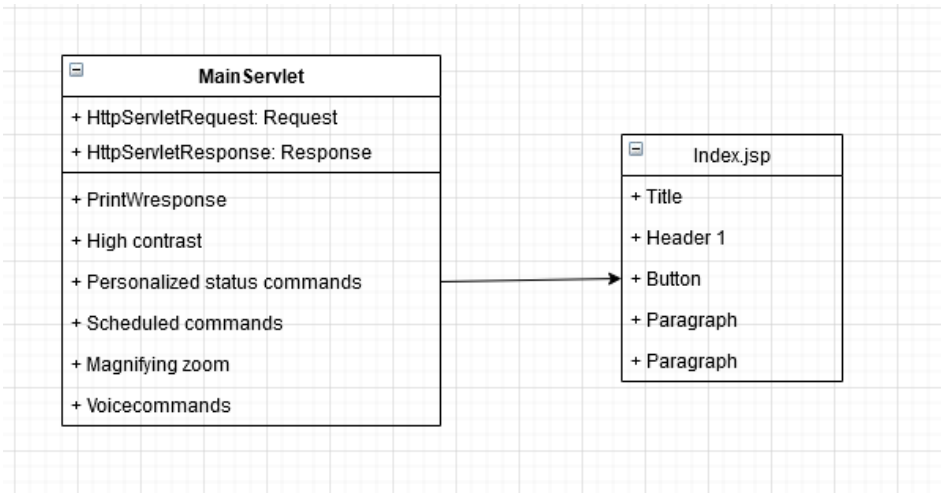

Figure 2 - Class diagram of mock client android app

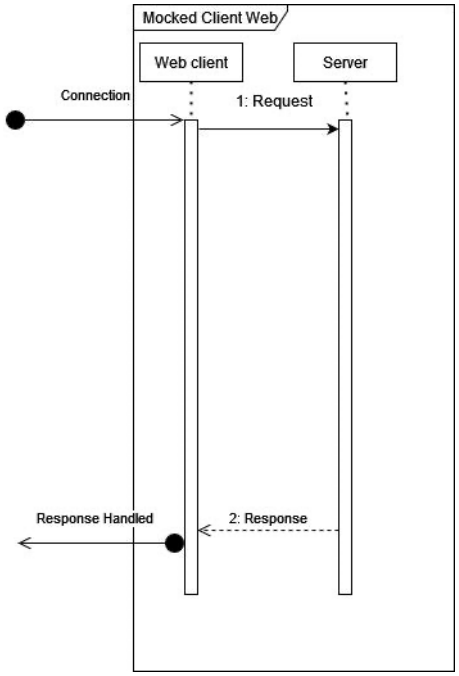

Figure 3 - Sequence diagram of connection of the mocked web client

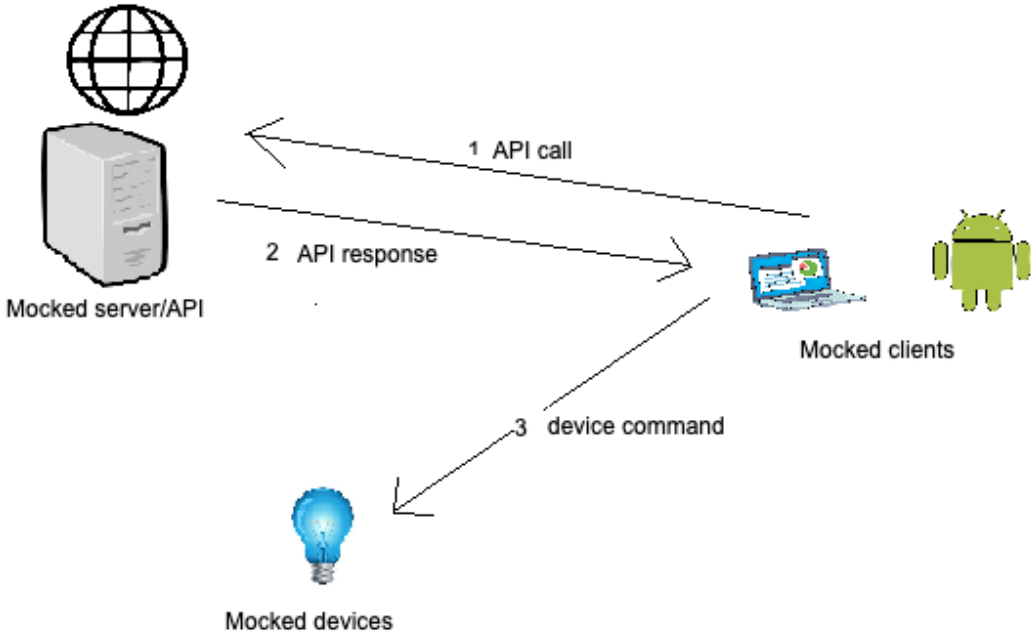

Figure 4 - General layout of mocked sever/client interaction

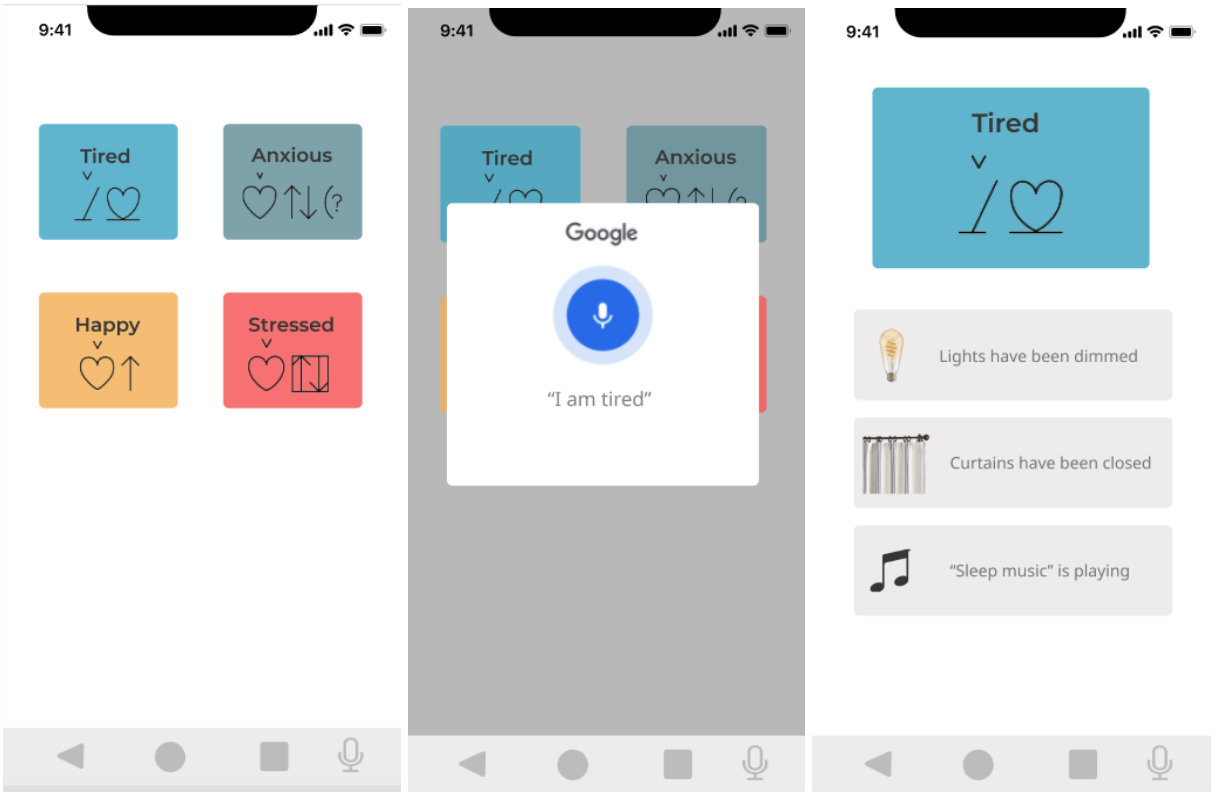

Figure 5 – Explanatory figures of R4

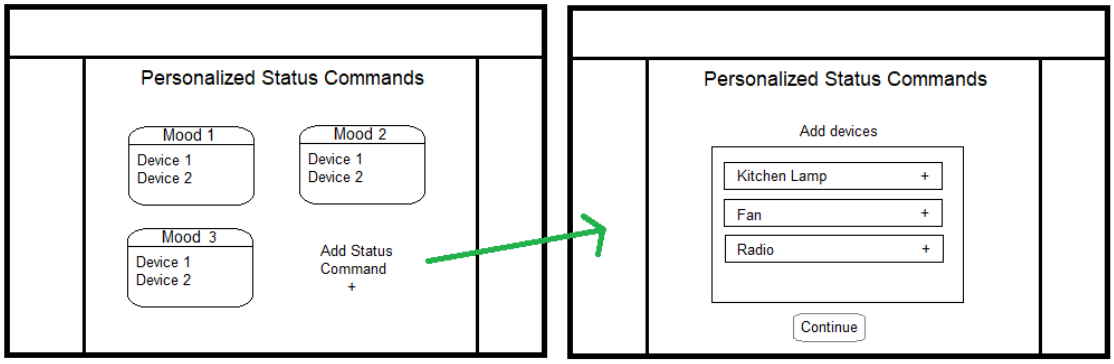

Figure 6 -– Design of status commands on the web

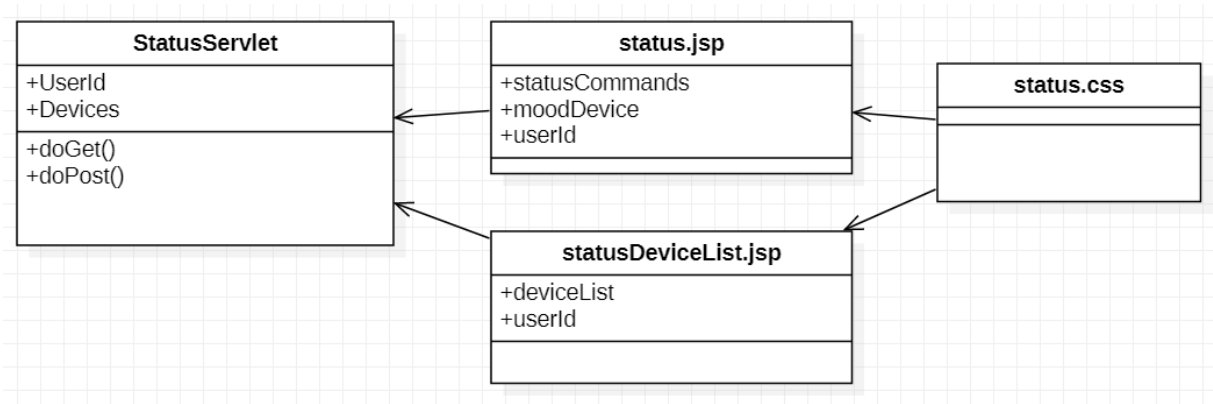

Figure 7 – Class diagram of status commands on the web

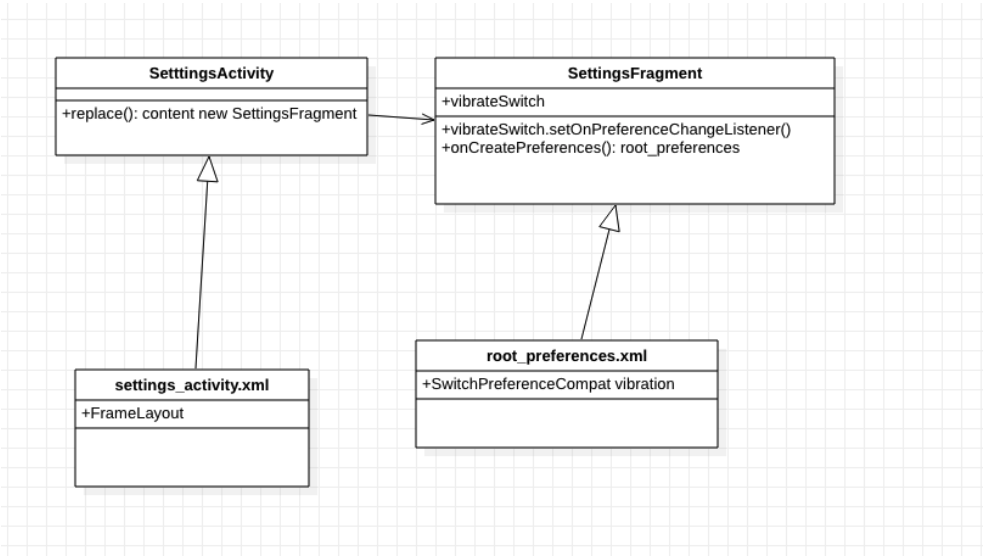

Figure 8- Class diagram of settings pane with preferences

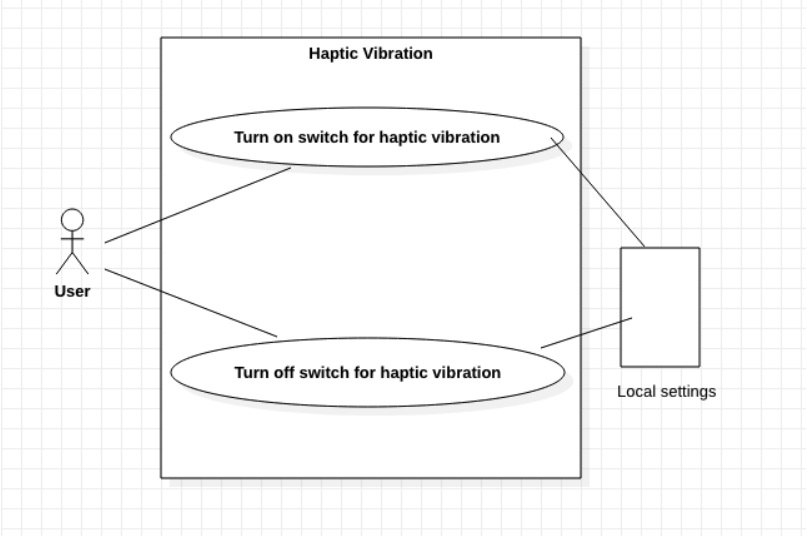

Figure 9 - Use case diagram of haptic vibration

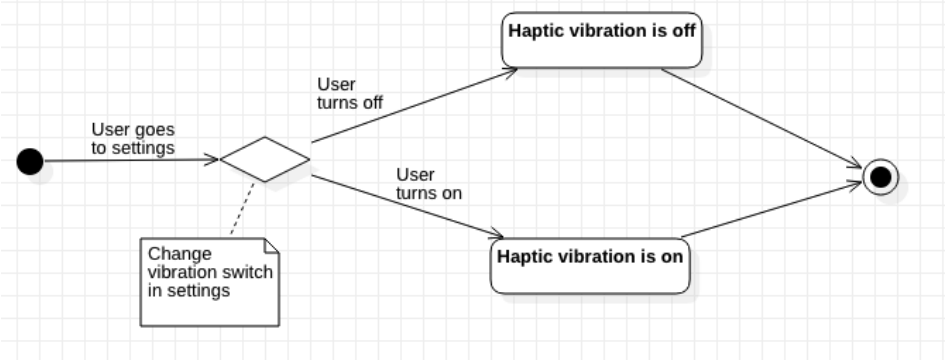

Figure 10- State diagram of haptic vibration

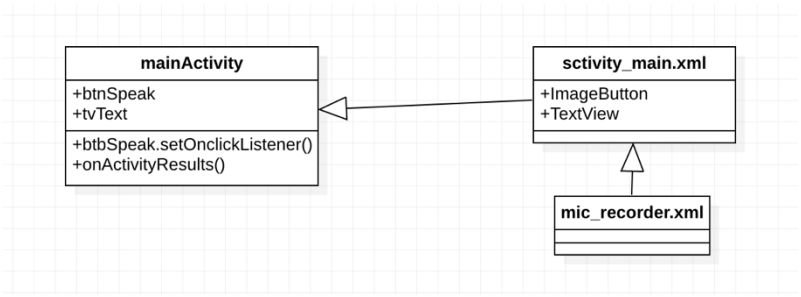

Figure 11- Class diagram of voice command on Android

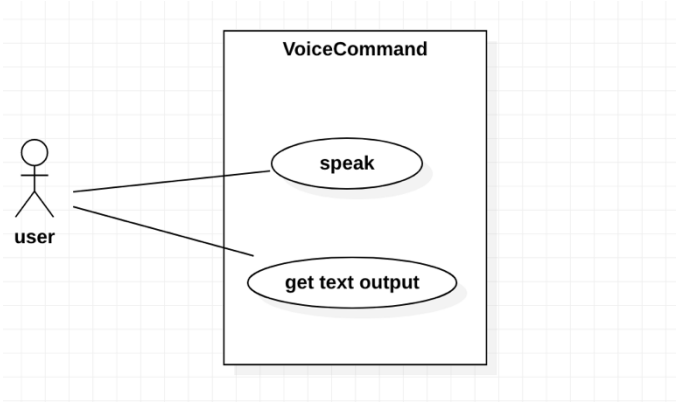

Figure 12- Use case diagram of voice command on Android

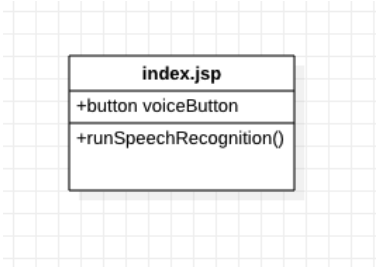

Figure 13- Class diagram of voice to text on web

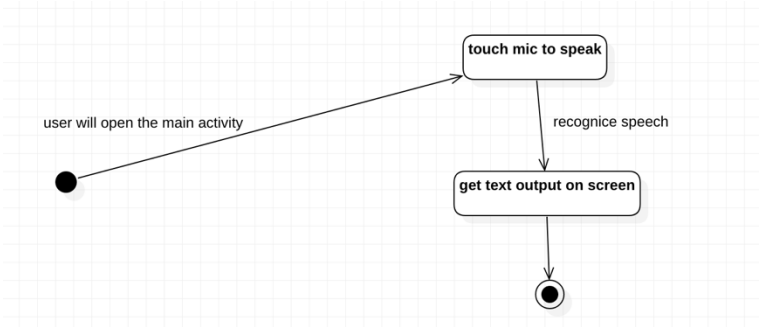

Figure 14- State diagram of voice command on Android

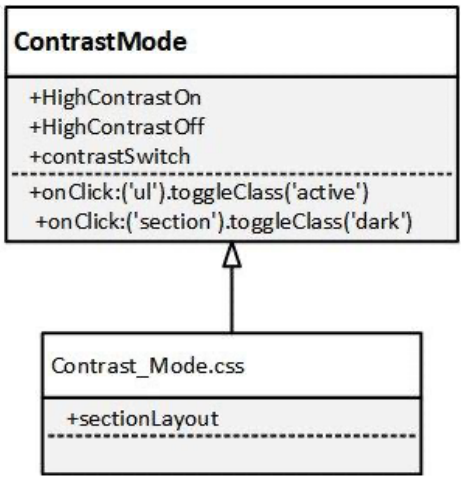

Figure 15- Class diagram of high contrast on web

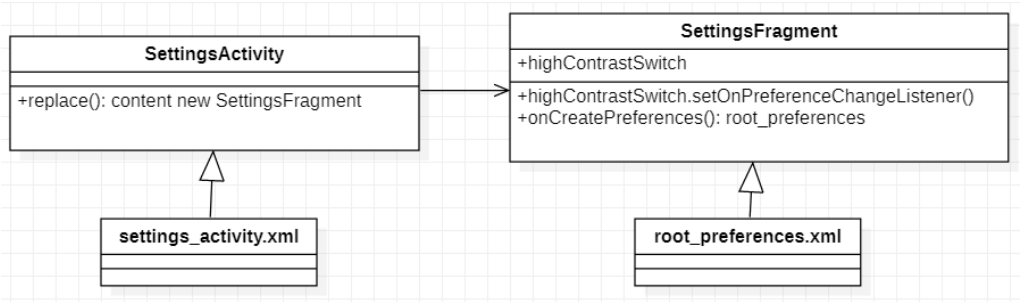

Figure 16- Class diagram of high contrast on Android

Use case diagram:

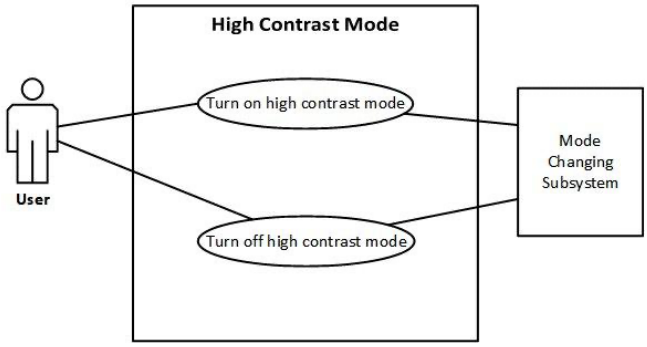

Figure 17- Use case diagram of high contrast

State diagram:

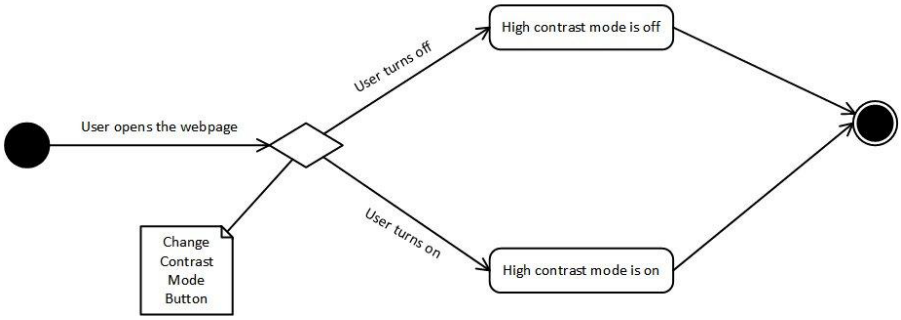

Figure 18- State diagram of high contrast on web

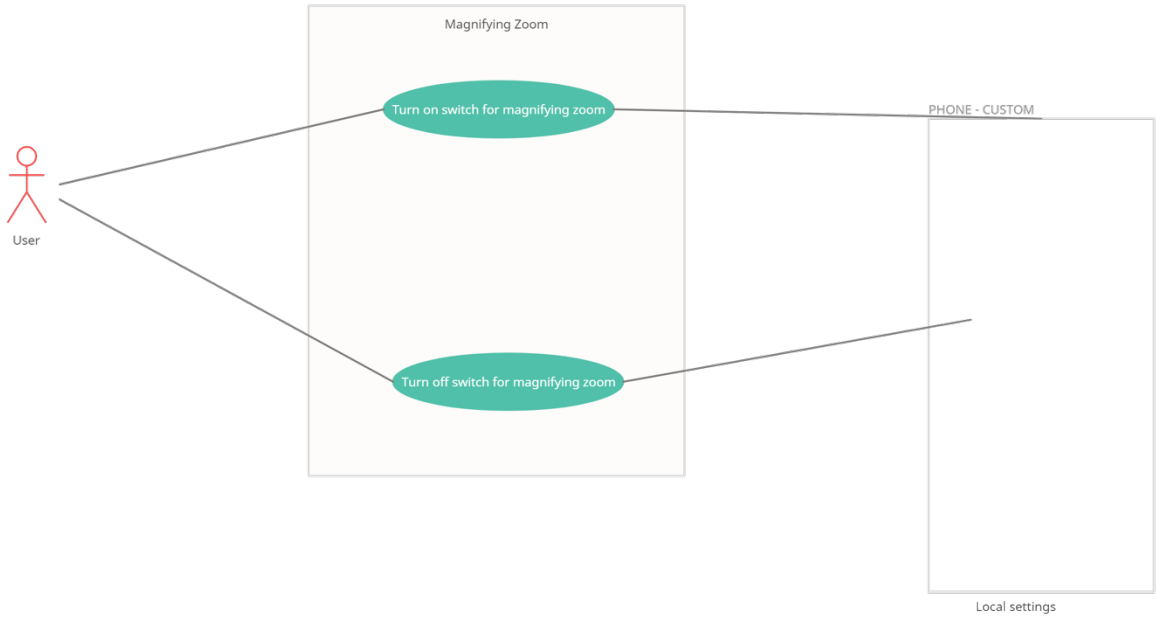

Figure 19- Use case diagram of magnifying zoom on Android

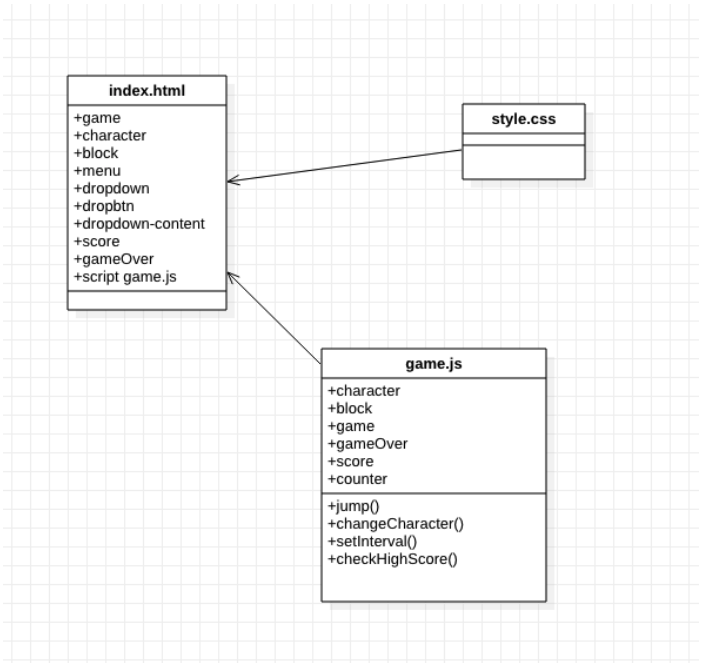

Figure 20- Class diagram of game

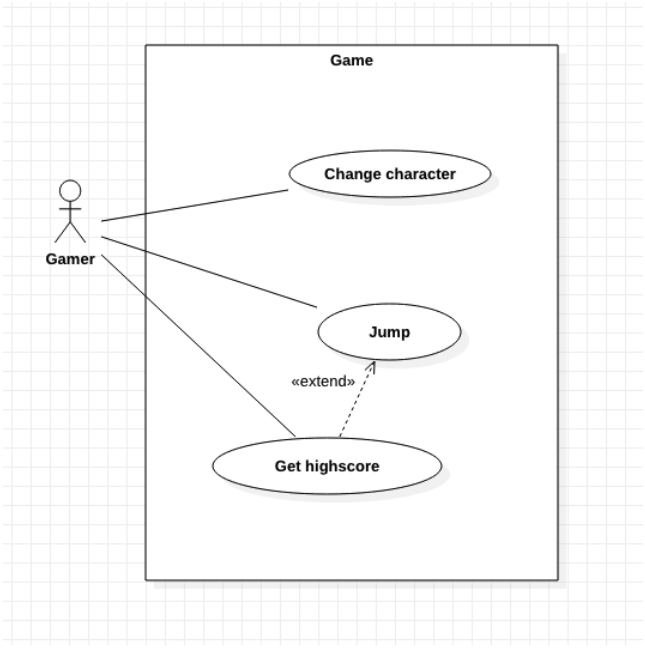

Figure 21- Use case diagram of game

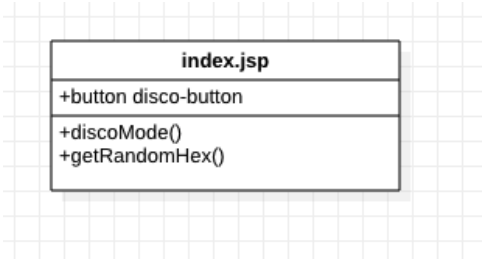

Figure 22- Class diagram of disco mode on web

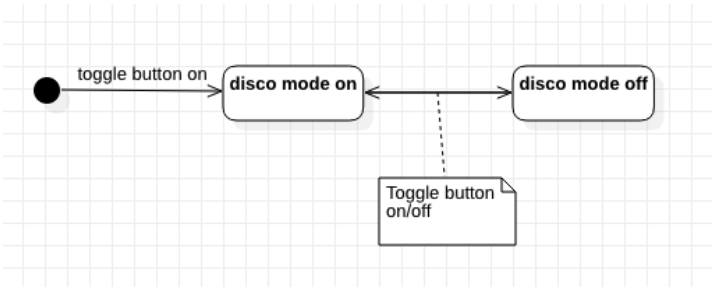

Figure 23- State diagram of disco mode on web

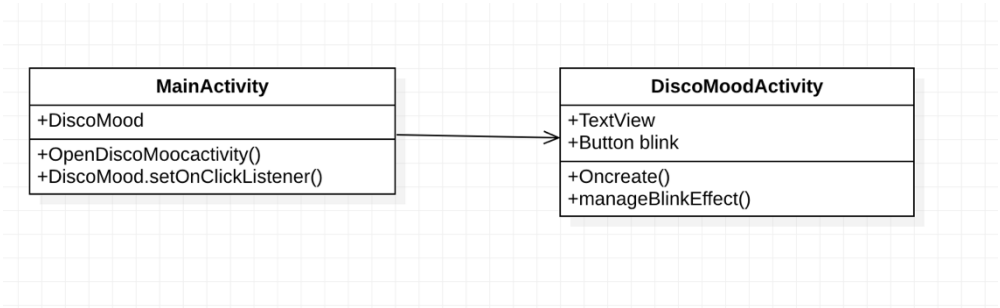

Figure 24- Class diagram of disco mode on Android

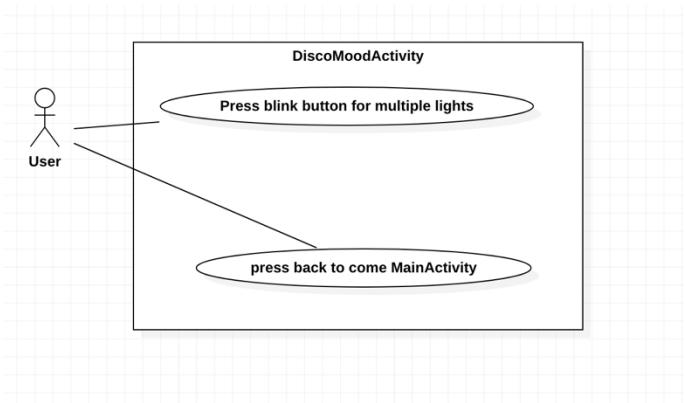

Figure 25- Use case diagram of disco mode on Andorid

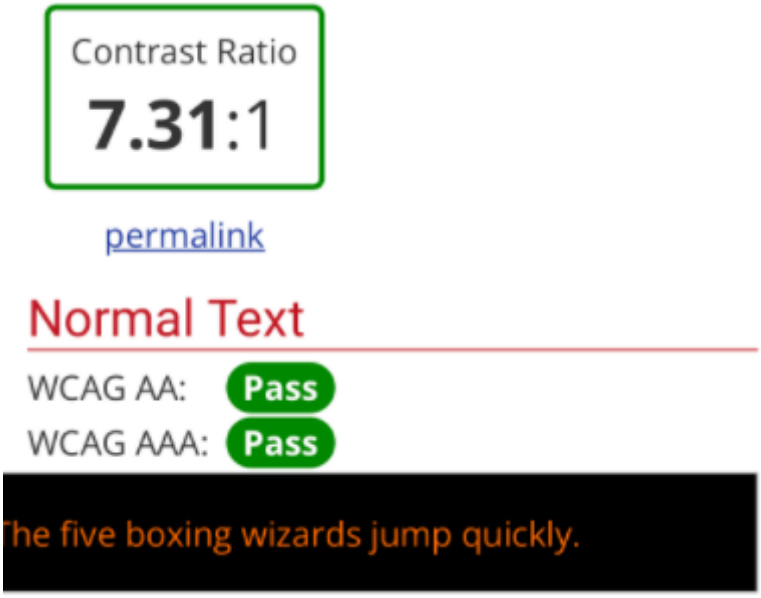

Figure 26- High contrast ratio

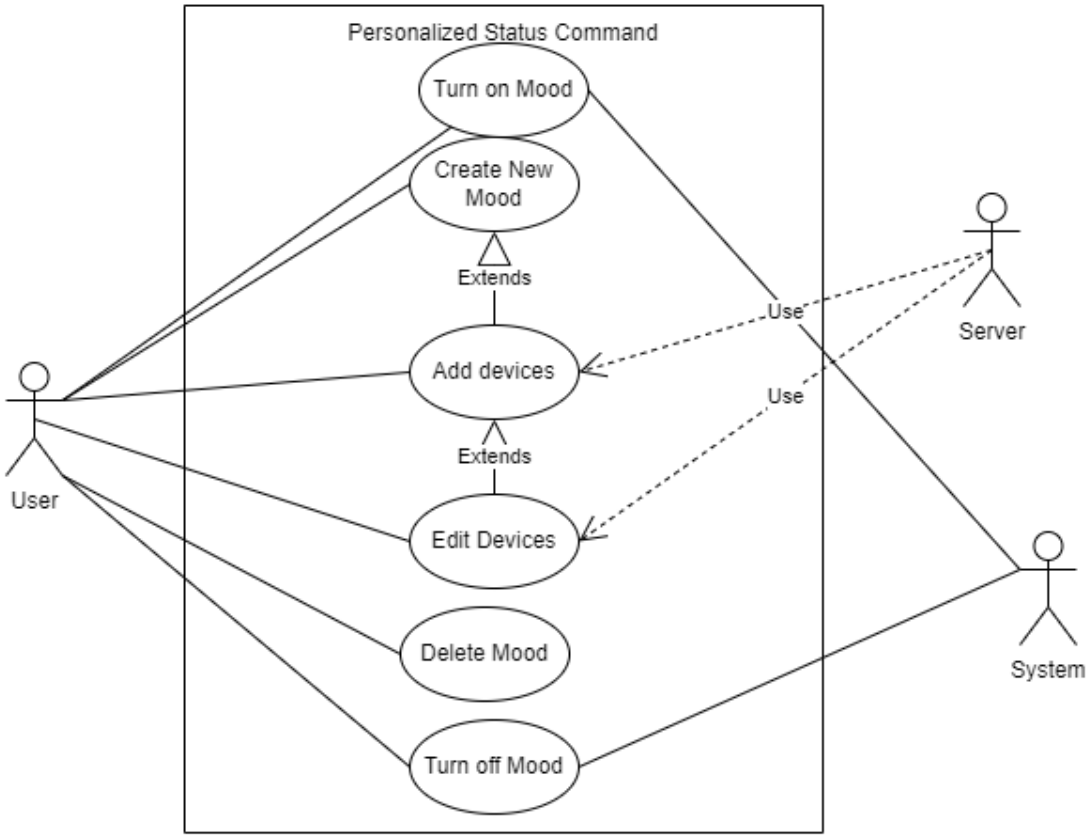

Figure 27- Personalized Status command Use case