Design Documentation – Unit group

Homedork – Interactive Smart House

Revision History

Name	Associated Letter
Hani Alzir	A
Ali Habesh	B
Stiv Abdulwahed	C
Amr Al-shaaba	D

Date	Version	Description	Author
2021-09-15	1.0	Modifying Design documentation	A, B, C, D
2021-10-04	1.1	Modified the images, changed POST to PUT, also images were not properly placed.	A, B, C, D
2021-10-22	1.2	UML classes - API UML classes - Model Use case diagram - Login Use case diagram - general use features	A, B, C, D
2021-11-15	1.2.1	Sequence diagram for lamps (under D6) Changed D6 “general” to “essential”	A, B, C, D
2021-12-6	2.0	Added Calendar Use case & Calendar UML class diagrams.	A, B, C, D

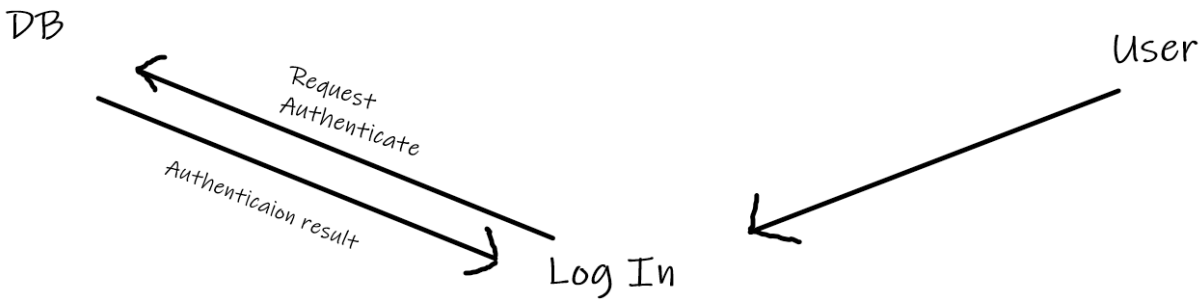
Design item List

Requirement Name	Priority
D1. Log in system (API get/post)	Essential
D2. Scrolls & On/Off button (API get/post)	Essential
D3. UML classes - API	Essential
D4. UML classes - Model	Essential
D5. Use case diagram - Login	Essential
D6. Use case diagram - essential features	Essential
D7. Use case and Class diagram - Calendar	Essential

Design Item Descriptions

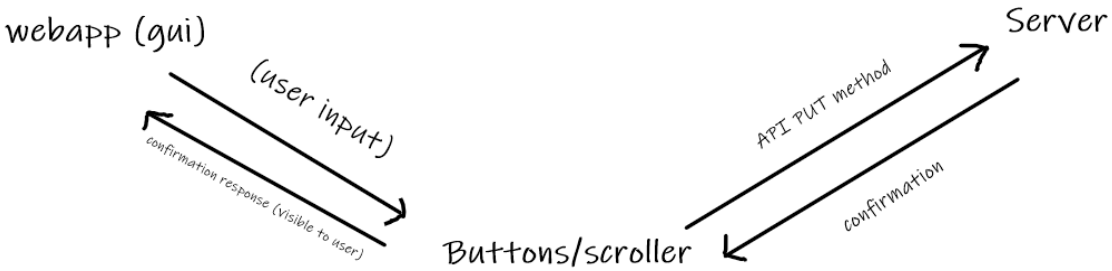
D1 – Log in

Login system is used for user authentication to be able to access control of their home (smart house). The basic idea is to use the database with some form of security to identity the user before giving him access.

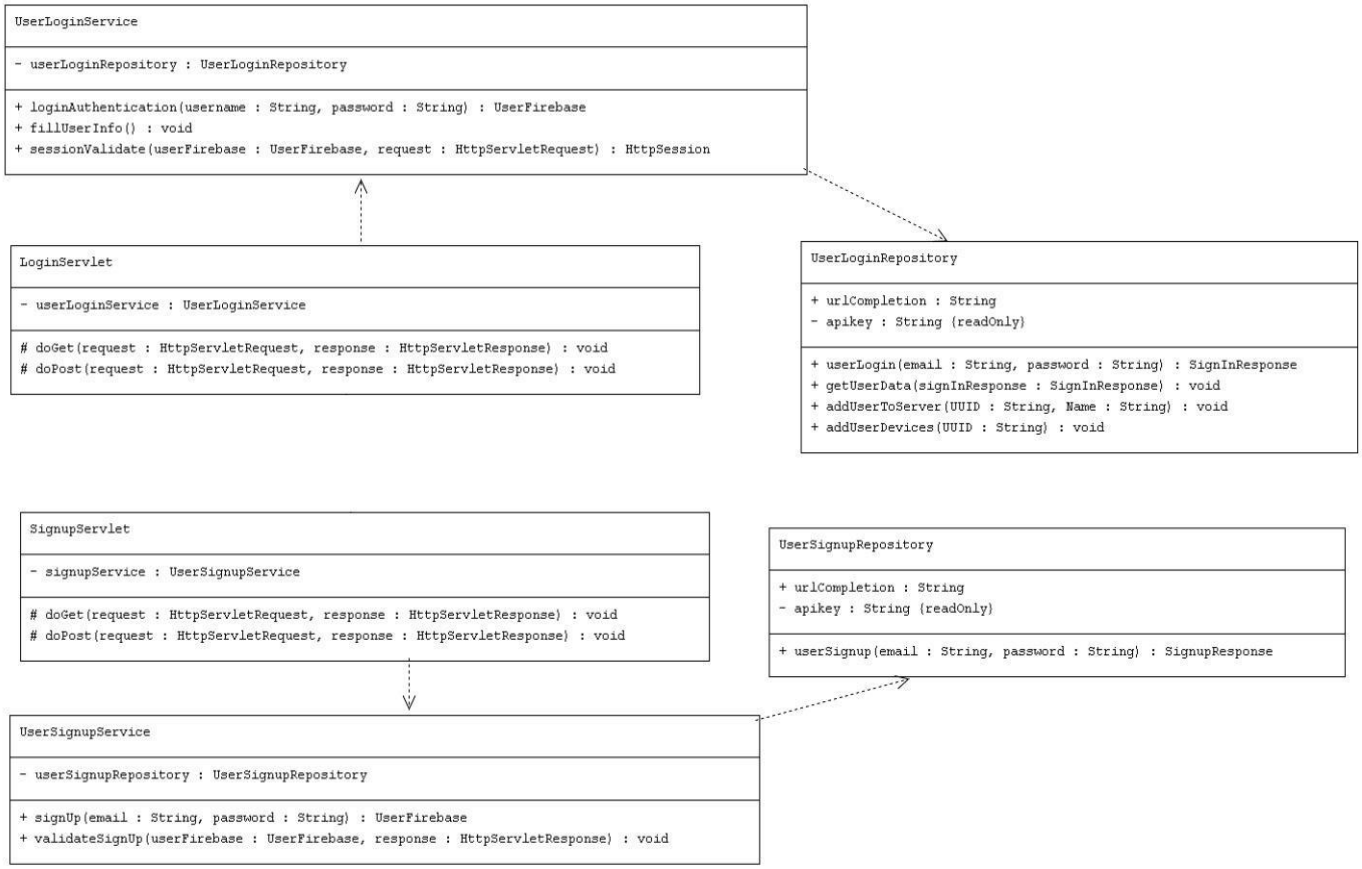


D2 - Scrolls & On/Off button

The user should be able to communicate to his smart house through his private profile using features such as scrollers and buttons. The scroller could be used for a more specific input such as light sensitivity, while the button can be used as more off a binary input (on/off). The buttons will give information and receive information using API's get/post methods, which essentially means the web/phone application will communicate with the server.

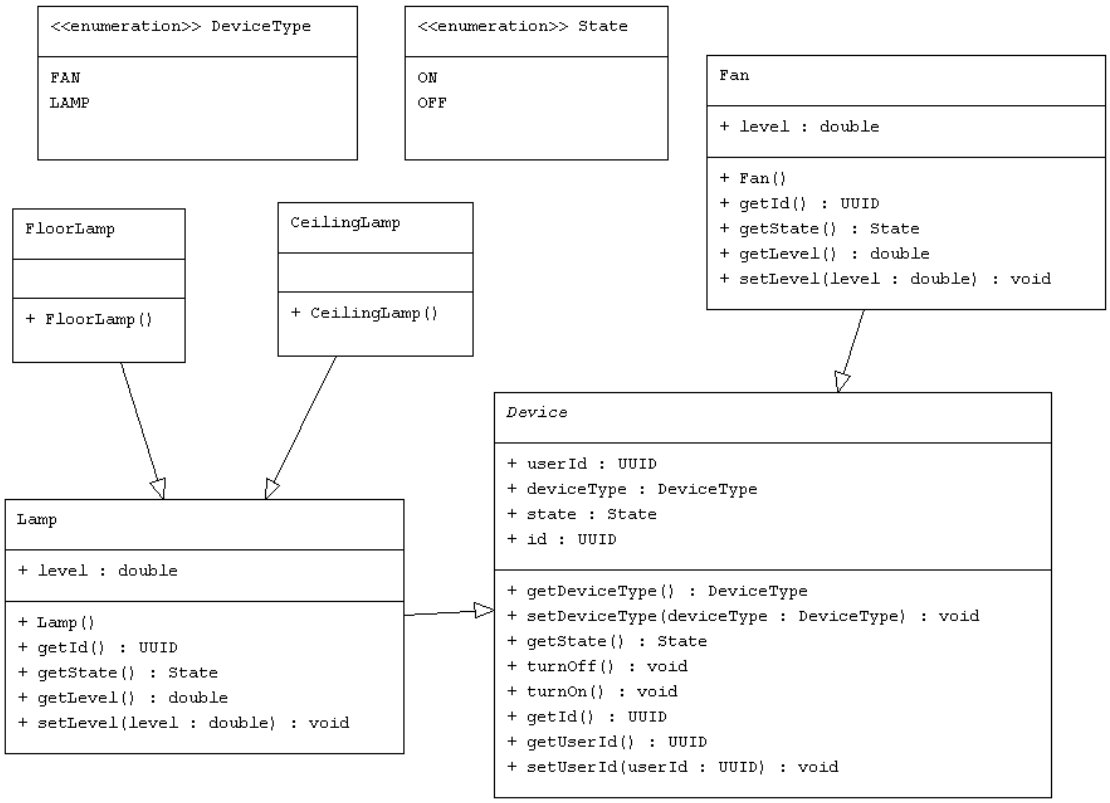


D3. UML class API



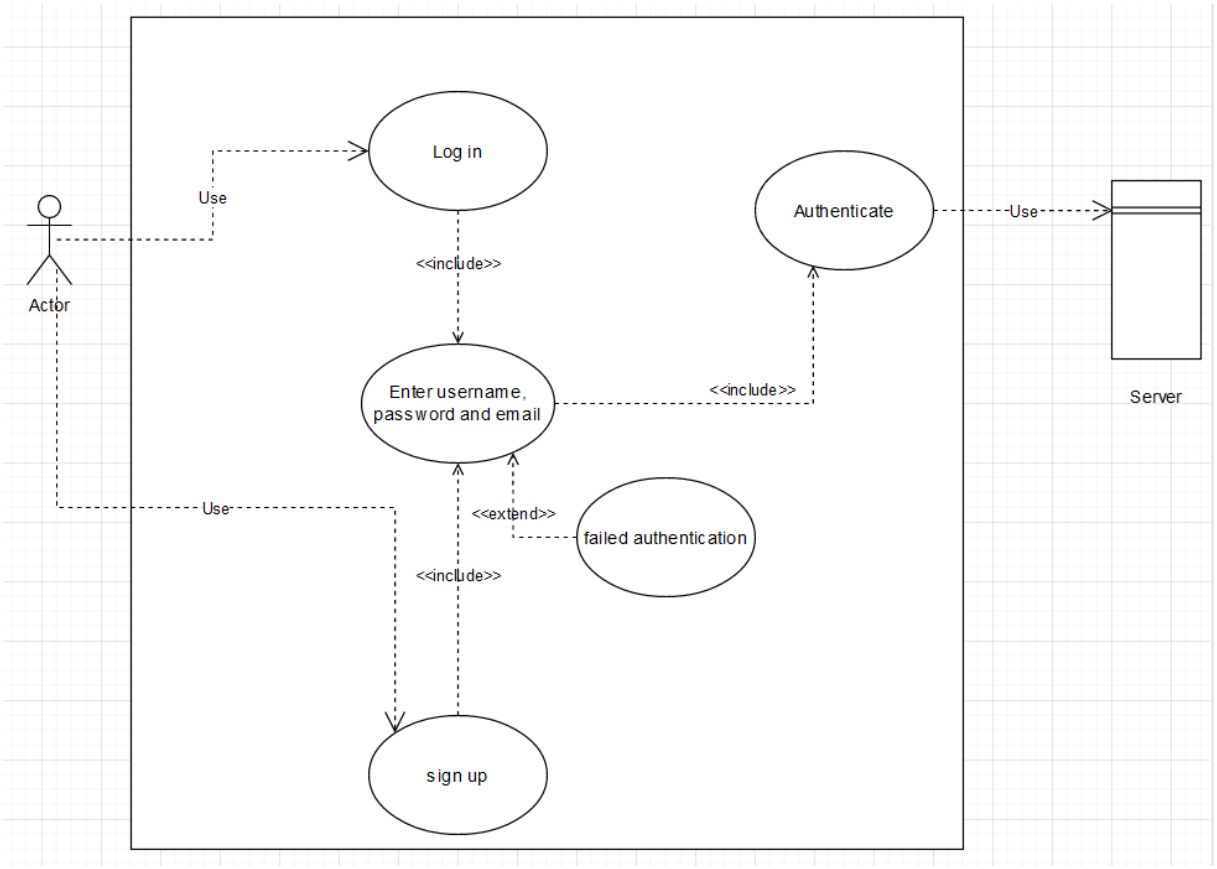
- The API UML class diagram shows the underlying structure of the api implementation, these classes are mandatory to create a working environment and to be able to communicate without fail using Homedorks API server.

D4. UML classes - Model



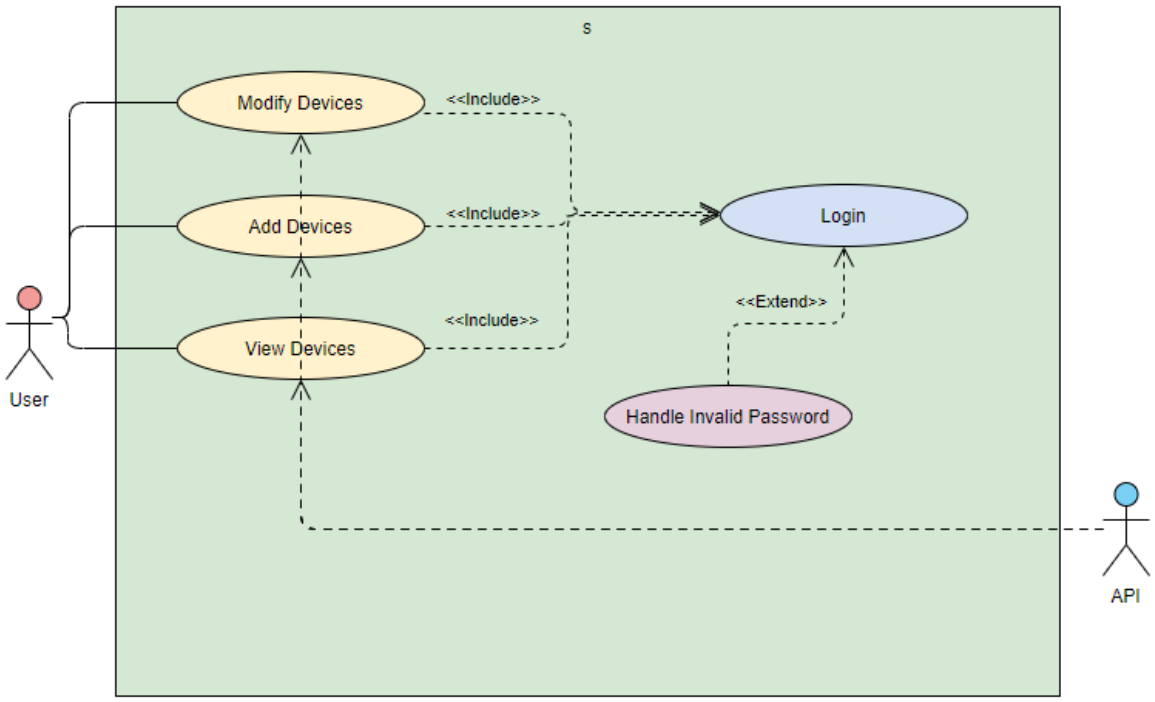
- The Model UML class diagrams depict the required classes in order to consume the Homedork API. Homedork API responds with a set of structured POJO Objects and/or List with POJO objects. In order to receive the response these classes will have to correspond to the server objects.

D5. Use case diagram - Login/Signup

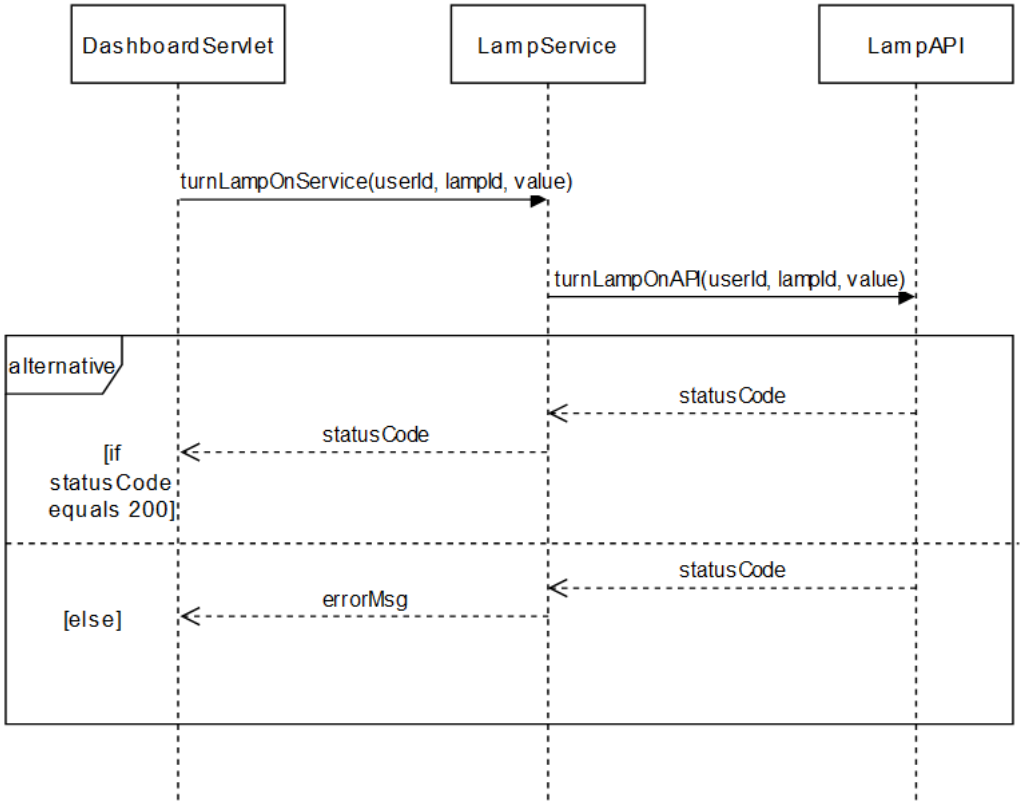


- The login use cases system. The system simply requests the user to input email, password and/or username depending on what you choose, either signup or login. The system then tries to authenticate/check for valid email and either a failed or successful response is expected.

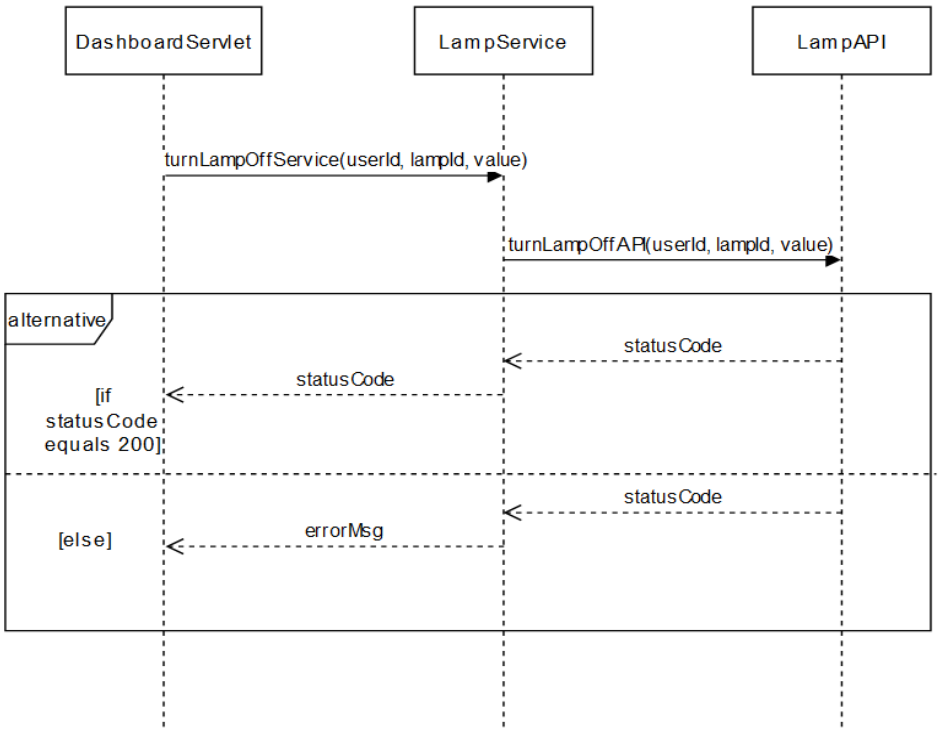
D6. Use case diagram - general use features



- The general use features correspond to the normal API consumption that will take place once the user either turns off/on TV, increases light brightness and much more. This is a general scenario of a functioning Homedork feature system.



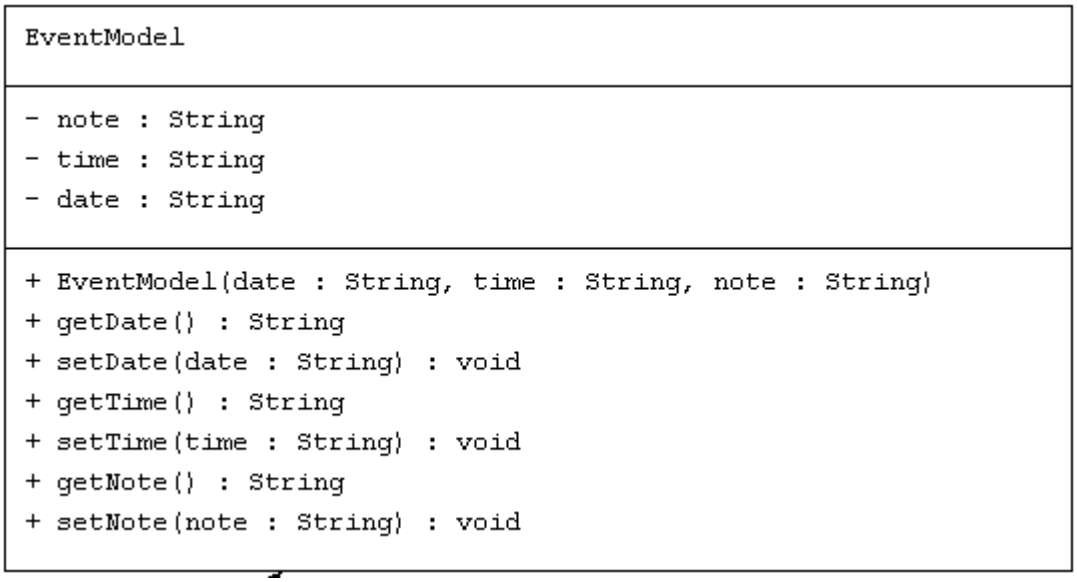
- The sequence diagram above depicts the scenario for lamp feature. This feature takes an input through two different ways, one is a button (on/off) and the other is a range slide (which changes light sensitivity). When the feature has sent a request to the server, it expects a string as return type, this string holds the status code which under the frame “alternative” is based on either 200 (which means success) or anything else which generally means there was no success.



- Similarly, to the above scenario, in this case the lamp simply turns on or off. The user has two ways of turning it off, either through a button or a range slide. If the status code is 200, nothing happens because it indicates everything went smoothly. If the status code is anything else, we get a simple error message.

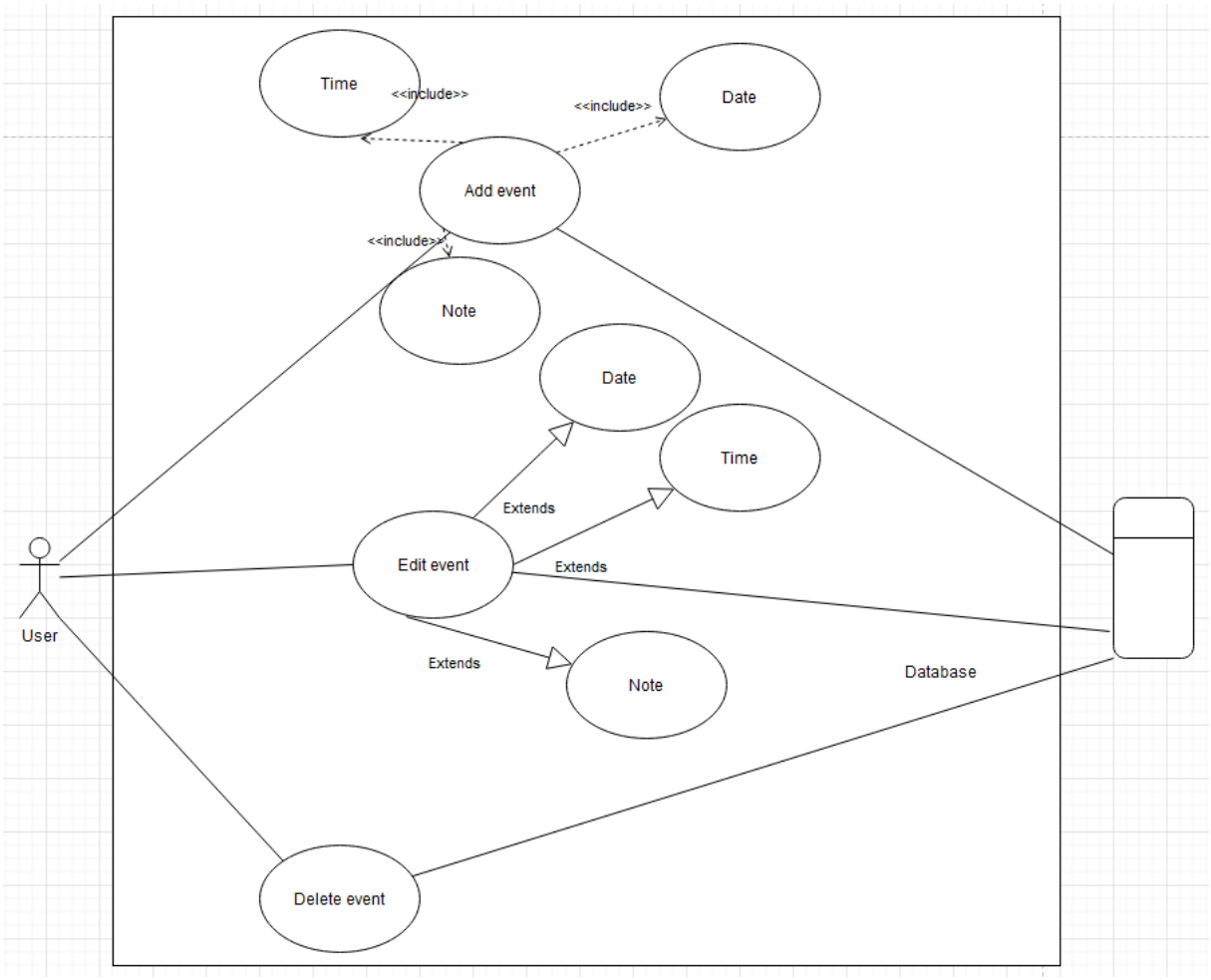
D7. Use case and Class diagram – Calendar

UML class diagram:



- EventModel represents the class which corresponds to the Json object that will be retrieved from firebase database. Within this class we store date as a string, time as a string and note as a string.

Use case Diagram:



- The main use case for Event calendar is to give the user the possibility to create an Event, when creating an event, the user has to enter date, time and a small note (max 30 characters), once the user adds the event it will immediately be stored into the firebase Realtime database. The secondary use cases are Edit and Delete. Edit will be available only if the user decides to change something in the event, whether its time, date or the note itself. The delete use case simply deleted an event from the database, by sending the events unique id.