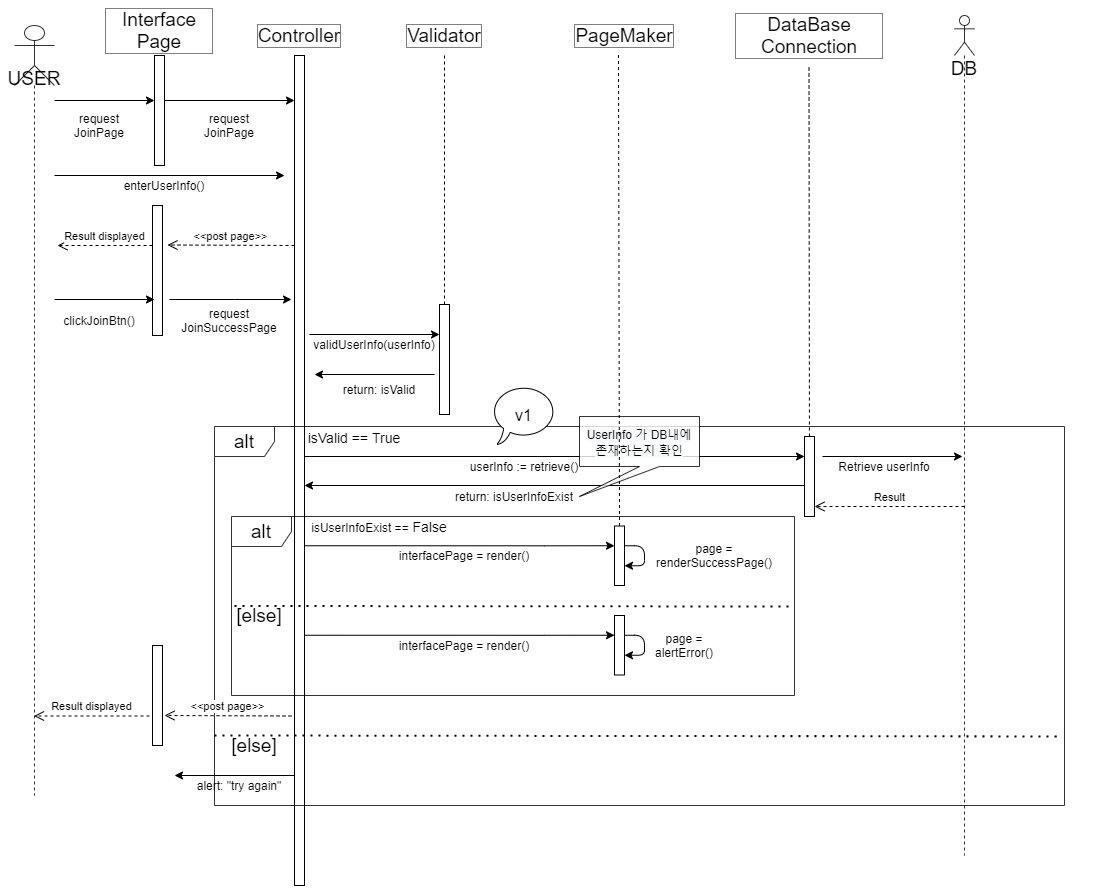
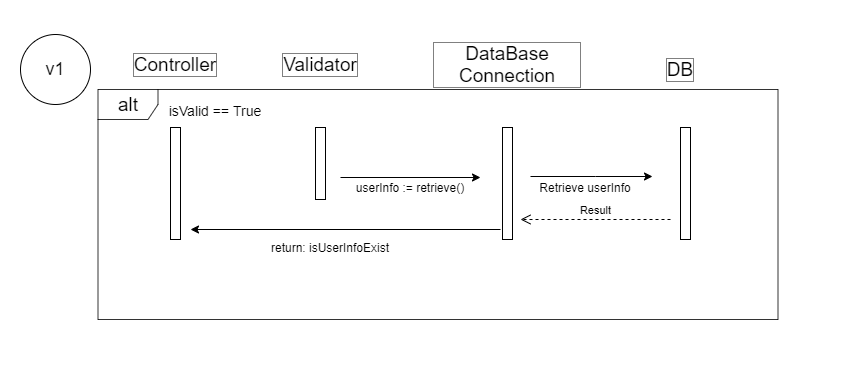
**Sub group1**

**[ sequence diagram for UC-1 ]**

**UC1: JOIN**



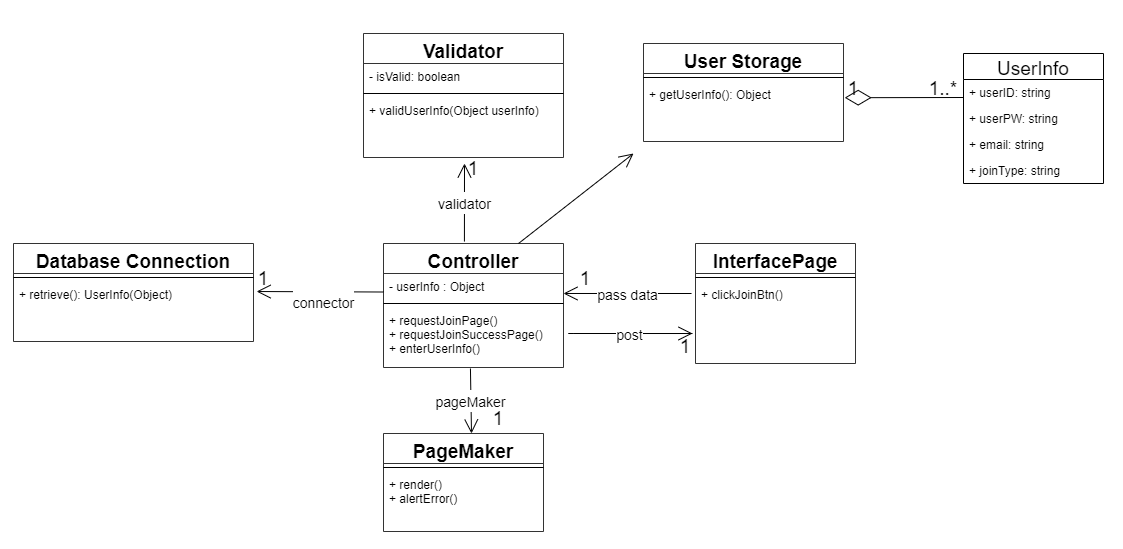
**Variation 1**



[Process]

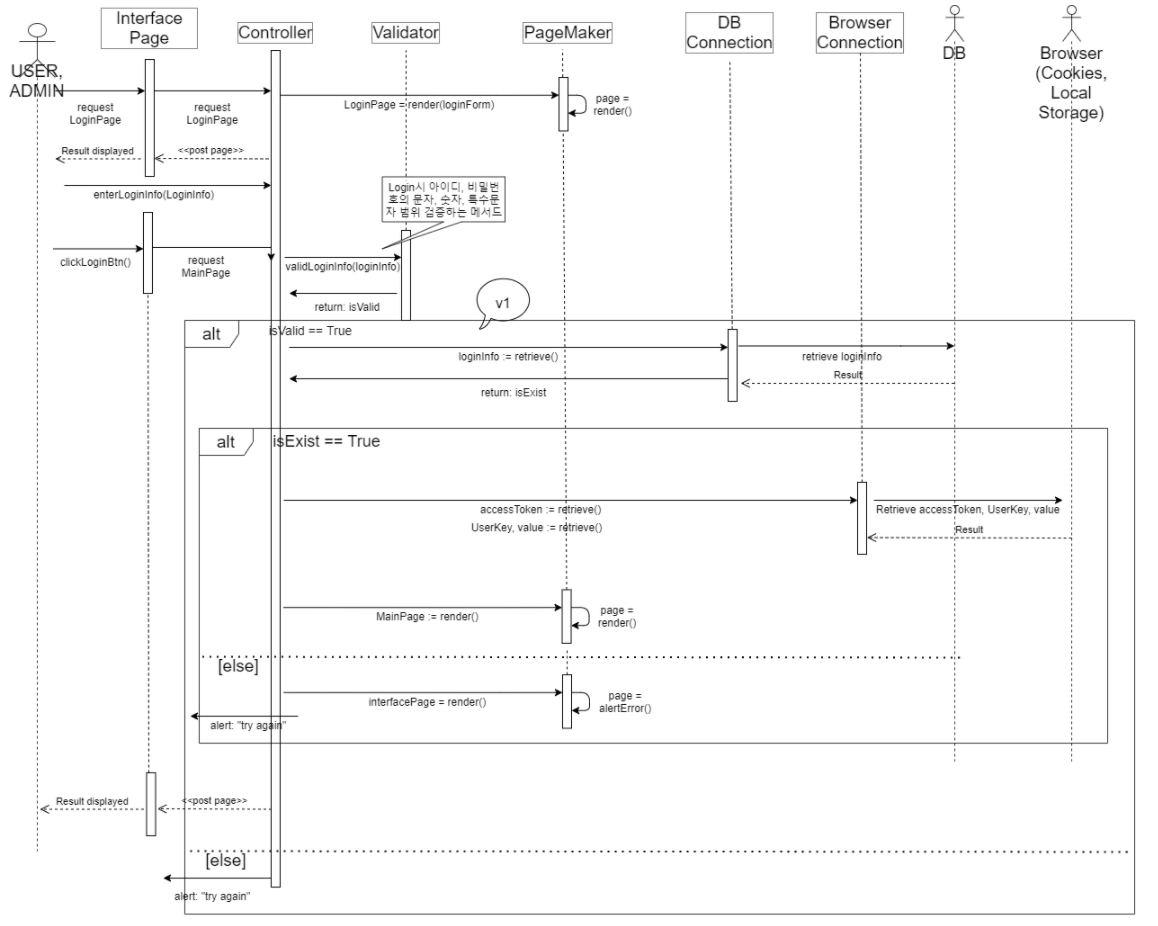
초기 설계 당시에는 Controller에 전문성을 부여하여 validator에서 회원가입 양식의 form이 validate하다면 controller에서 valid한 회원가입 정보를 데이터베이스에 전달하는 방식이였다면 variation1에서는 validator에서 자체적으로 검증하고 controller에 valid를 던지는 방식이 아닌 해당 step에서 바로 데이터베이스에 회원가입 정보를 전달하는 방식을 고안하였다. 이를 통해 얻게 되는 이점은 controller에 하중된 책임들을 다른 domain concept로 분산시킴으로써 workload를 밸런스하게 맞출 수 있고 각 object간의 short communication chain이 가능하다.

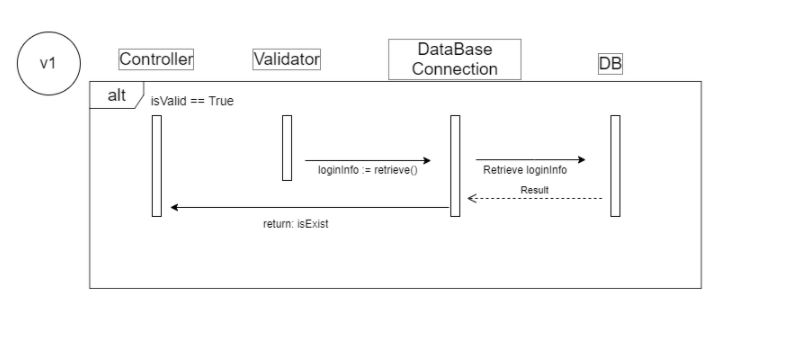
**[ class diagram for UC-1 ]**



**[ sequence diagram for UC-2 ]**

**UC2: Login**



**Variation 1**

[Process]

Usecase 1과 마찬가지로 초기 설계 당시에는 Controller에 전문성을 부여하여 validator에서 로그인의 정규표현방식이 validate하다면 controller에서 valid하다는 것을 리턴받고 valid한 로그인 정보를 데이터베이스에 전달하는 방식이였다면 variation1에서는 validator에서 자체적으로 검사하고 controller에 valid를 던지는 방식이 아닌 해당 step에서 바로 데이터베이스에 로그인 정보를 전달하는 방식을 고안하였다. 이를 통해 얻게 되는 이점은 마찬가지로 controller에 하중된 책임들을 다른 domain concept로 분산시킴으로써 workload를 밸런스하게 맞출 수 있고 각 object간의 short communication chain이 가능하다.

**[ class diagram for UC-2 ]**

