1. In the searching use case, we didn’t do the doubleBookChecker() function to check whether a space has been booked twice, instead we use a “is\_taken” flag in our database to indicate whether a space has been taken. If yes, then it will not be shown on the map so user will not see it. We made this change because we believe this way is more efficient and easy to implement. Also, this “taken” flag solves our exception.
2. In the renter profile creation use case, as first we want to use Facebook API for authentication but we can’t make that work. Therefore we only use an email validation API, and only if the user enters a nonexistent email the app would not allow him/her to create a profile.
3. Also we sort of combine two use cases together—the renter profile creation and the space owner profile creation. We created a user table in our database and any user can either rent available spaces or put their unused space online. We believe this is a more reasonable choice because it’s not necessary to make a clear distinction between renters and space owners. (Sometimes a space owner may want to rent spaces in another place, for example.) As a result, we made a new use case call “user profile creation” and delete the previous “renter file creation” and “space owner profile creation”.
4. We added a new use case called “parking space registration” because when we implemented our app we found this use case is quite important. And our app actually has that functionality now (Part of the function are not finished, such as the space validation part).
5. For the other use cases “Haggling”, “Scheduling/Reservation” and Review and Comment haven’t been implemented now and they will be completed in the future.