캡스톤 개인보고서

이름 : 차무송

학번 : 32154577

학과 : 소프트웨어학괴

교수님 : 송 인식 교수님

이번 주는 개발 후 기능 테스트 중 일어난 이슈를 조금 수정하였습니다. 첫 번째로 저번에 구현해 놨던 타이머는 액티비티 형태였습니다. 실제 프로젝트에 합치려면 프래그먼트로 바꿀 필요가 있었습니다. 바꾸는 과정에서 여러가지 이슈를 수정했습니다.

이태까지 프래그먼트들을 viewpager형태로 장현준 조원이 전부 구현해왔습니다. 제가 만든 기능을 장현준 조원이 합치면서 문제가 생겼고 이로 인해 프래그먼트 공부를 하면서 이슈를 수정해봤습니다.

지금의 프래그먼트 형태는 여러 개의 화면을 좌, 우 슬라이드나 하단에 있는 네비게이션 바를 탭하여 슬라이드 하면서 움직이게 끔 만들었습니다. 그리고 타이머 액티비티를 프래그먼트로 변경 후 합치 결과 프래그먼트의 특성 때문에 타이머가 제대로 작동하지 않았습니다. 프래그먼트들을 슬라이드 형식으로 사용하기 위해서는 어댑터가 필요합니다. 원래 사용하던 어댑터는 프래그먼트를 두 개 이상 이동하면 원래 사용하고 있던 프래그먼트를 지워버립니다. 메모리의 효율을 위해서 입니다. 결론적으로 타이머가 백그라운드에서 실행은 되고 있지만 프래그먼트를 두 칸 이상 이동하게 되면서 새로운 프래그먼트가 생성되기 때문에 진행되고 있는 타이머의 형태는 볼 수가 없는 이슈가 생겼습니다.

```
class AdapterMainFragment(fm : FragmentManager, private val fragmentCount : Int) :
FragmentStatePagerAdapter(fm){
    override fun getCount(): Int = fragmentCount

    override fun getItem(position: Int): Fragment {
        return when(position){
            0 -> FragmentMainTimer_Java()
            1 -> FragmentCalendar()
            2 -> FragmentMainHome()
            3 -> FragmentMainVideo()
            4 -> FragmentMainSetting()
            else -> FragmentMainHome()
        }
    }
}
```

[수정전]

```
class AdapterMainFragment(fm : FragmentManager, private val fragmentCount : Int) :
FragmentPagerAdapter(fm){
    override fun getCount(): Int = fragmentCount

    override fun getItem(position: Int): Fragment {
        return when(position){
            0 -> FragmentMainTimer_Java()
            1 -> FragmentCalendar()
            2 -> FragmentMainHome()
            3 -> FragmentMainVideo()
            4 -> FragmentMainSetting()
            else -> FragmentMainHome()
        }
    }
}
```

[수정후]

FragmentStatePagerAdapter 에서 FragmentPagerAdapter로 변경하였습니다. 저희 어플리케이션을 프래그먼트의 수가 고정적입니다. 그리고 프래그먼트를 지우고 다시 생성하게 되면 작성하고 있던 데이터들이 화면 전환시마다 다시 입력해야 하는 상황이 생깁니다. 따라서 어댑터 인자를 변경 후 실행해본결과 타이머 동작 중 2칸 이상 움직여도 문제 없다는 것을 확인했습니다.

하지만 그 이후 타이머에 시간을 설정 후 동작을 하지 않은 상태에서 프래그먼트를 옮기고 다시 와보니 초기 00:00:00 값으로 리셋이 되어있었습니다. 따라서 데이터의 값이 꼬이기 시작하면서 오류가 생기는 이슈가 생겼습니다.

```
private fun configureBottonNavigation() {
    xml_main_viewpager.adapter = AdapterMainFragment(supportFragmentManager, 5)
    xml_main_tablayout.setupWithViewPager(xml_main_viewpager)

    val viewBtmNaviMain : View =
this.layoutInflater.inflate(R.layout.btm_navigation_main, null, false)
    xml_main_tablayout.getTabAt(0)!!.customView =
viewBtmNaviMain.findViewById(R.id.xml_btmnv_btn_timer)
    xml_main_tablayout.getTabAt(1)!!.customView =
viewBtmNaviMain.findViewById(R.id.xml_btmnv_btn_calendar)
    xml_main_tablayout.getTabAt(2)!!.customView =
viewBtmNaviMain.findViewById(R.id.xml_btmnv_btn_main)
    xml_main_tablayout.getTabAt(3)!!.customView =
viewBtmNaviMain.findViewById(R.id.xml_btmnv_btn_video)
    xml_main_tablayout.getTabAt(4)!!.customView =
viewBtmNaviMain.findViewById(R.id.xml_btmnv_btn_setting)
}
```

위의 코드는 MainActivity중 하단 내비게이션과 viewpager를 이어주는 부분입니다. 이 부분에서 작동중이지 않은 데이터는 마음대로 지우기 때문에 이슈가 발생했습니다.

```
xml_main_viewpager.setOffscreenPageLimit(5);
```

제일 하단에 위의 코드를 추가하여 viewpager가 데이터를 기억할 수 있는 page 수(Fragment 수)를 5개로 늘리고 실행한 결과 문제없이 합쳐졌습니다.

그 다음 장현준 조원이 구현한 하단의 네비게이션 바, 좌 우 슬라이드 기능으로 프래그먼트 사이를 이동할 수 있었습니다. 이제 구현했던 메인 화면에서 버튼을 클릭해서 프래그먼트들을 이동시키기 위해 작업을 하였으나 Activity에서 호출하는 것과는 달리 Fragment에서는 많은 제약이 있었습니다. Stackoverflow를 참고하여 간단한 함수를 생성 후 이동이 가능해졌습니다.

```
fun selectFragment(position : Int) {
    xml_main_viewpager.setCurrentItem(position, true)
}
```

위와 같은 함수를 MainActivity에서 생성해주었습니다. 호출하고자 하는 Fragment에서 위의 함수를 Int값과 함께 호출해주면 viewpager객체가 해당 번호랑 맵핑되어있는 Fragment로 이동시켜줍니다.

```
override fun onClick(v: View?) {
    (activity as MainActivity?)!!.selectFragment(3)
}
```

Github - https://github.com/hyeonjun414/Homenagement