**Realm**

Realm은 안드로이드에서 사용할 수 있는 데이터베이스 오픈 소스 라이브러리 중 하나이다. SQLlite, RoomDB 등 여러가지 선택지가 존재했지만 Realm을 이번 프로젝트의 데이터베이스 시스템으로 정한 이유에는 몇 가지가 있다. 첫 번째로 복잡한 Query 문을 사용하지 않고 편리하게 데이터들을 다룰 수 있다. 또한 2.5MB 정도의 저장공간을 소모하지만 RoomDB보다 더 높은 성능을 보여준다는 점이다. 그리고 ORM(Object Relational Mapping) 방식을 차용한 데이터베이스이기에 편리하게 데이터들을 모델링하여 클래스로 관리하고 이를 별다른 과정 없이 그대로 데이터베이스에 저장하여 사용할 수 있다는 점이 상당히 매력적이다.

CEE 프로젝트의 경우 경제 용어들과 그 용어들을 주제로 한 퀴즈들의 데이터를 가지고 있어야 했다. 또한 이러한 경제 용어들은 분류 방식, id 등 여러가지 속성들을 가지고 이러한 속성들로 검색이 가능해야 했기에 용어들과 퀴즈들을 객체로 모델링하고 이를 Realm으로 관리하여 효율적으로 용어들과 퀴즈 데이터들을 저장하고 사용할 수 있었다.

**Web Crawling (JSoup)**

CEE 프로젝트는 시각장애인을 위한 경제 교육 어플리케이션이라는 정체성이 있다. 따라서 여러 방면으로 시각장애인 분들을 배려하는 요소들을 적용하였는데 그 중 하나가 Web Crawling 방식을 사용하여 어플리케이션 업데이트를 지양하는 방법이다. 시각장애인 분들과 인터뷰를 진행하였을 때 어플리케이션을 업데이트 해야 할 경우 상당히 불편함을 느끼신다는 정보를 얻을 수 있었다. 그리하여 추가되는 용어들이나 퀴즈들을 위한 서버가 필요했지만 실시간으로 통신하는 서버의 경우는 유저가 네트워크 연결이 항상 되어 있어야 한다는 점이 단점으로 작용했기에 한 번에 데이터를 받아와 데이터베이스 시스템을 구축하는 방식으로 구성해야 했다. 그리하여 GitHub의 무료 Web Hosting 기능을 활용하여 데이터들을 담고 있는 가벼운 웹 사이트를 호스팅하는 방식을 사용했다. 그리고 첫 다운로드 시점에 해당 사이트를 JSoup을 이용하여 크롤링하여 Realm을 활용한 데이터베이스 시스템에 데이터를 담는 기능을 구현하였다.

**TTS (Text-To-Speech) & Handling Accessibility Events**

안드로이드에는 기존의 talkback(Voice Assistant)라는 시각장애인을 위한 TTS 서비스를 제공하고 있다. 다만 CEE 프로젝트에 해당 TTS 서비스를 그대로 적용하기에는 몇 가지 애로사항이 있었다. 우선 시각장애인 분들은 기존의 talkback 서비스의 speech rate(읽어주는 속도)를 거의 2배에 가까이 설정해두고 쓰신다는 점이 문제로 다가왔다. CEE 어플리케이션은 상당히 긴 텍스트들과 복잡한 정보들을 담고 있기에 빠른 속도로 듣기에는 무리가 있기 때문이다. 그렇다고 매번 speech rate를 다시 조작하는 것은 사용자들 입장에 큰 단점으로 다가올 것이 분명했다. 또한 기존 talkback 서비스는 한 텍스트 내에서 뒤로 가기 혹은 앞으로 가기 등의 기능이 없고 다시 선택하게 될 경우 항상 처음부터 다시 읽어주는 메커니즘이었기에 한 텍스트가 담고 있는 컨텐츠가 큰 CEE 프로젝트의 경우에는 이 역시 매우 큰 단점으로 작용했다. 이러한 상황의 해결책으로 기존 talkback을 막고 CEE 만의 새로운 TTS 서비스를 도입하기로 하였다.

자체 TTS 서비스 도입을 위해서는 안드로이드 Voice Assistance를 위한 Accessibility 이벤트들을 hooking하여 CEE 만의 TTS로 대체를 해줘야 했다. 문제는 발생하는 모든 Accessibility 이벤트를 hooking 할 수 없었는데 안드로이드의 기본 네비게이션 버튼들 혹은 edge panel에서 발생하는 이벤트들은 Activity의 Accessibility 이벤트에 포함되지 않았기 때문이다. 이렇게 된다면 자체 TTS와 talkback이 겹치게 되는 문제가 발생하게 된다. 이를 해결하기 위해 여러 타입의 Accessibility 이벤트들이 정확히 어떤 순서로 발생하는지 체크하고 이를 활용하여 예외들을 처리함으로써 해당 문제를 해결하였다.

자체 TTS는 google의 TTS engine을 사용하여 개발하였다. 기존 google의 TTS engine은 stop 할 경우 재생 중이던 텍스트를 저장하지 않고 초기화되는 문제가 있었다. 따라서 재생하고자 하는 문자열들을 ArrayList로 관리하고 index 변수를 두어 어디까지 읽었는지 표시하게끔 하였다. 이 과정에서 TTS API에서 제공하는 onRangeStart 메소드를 활용하여 특정 시점마다 index 변수를 업데이트 시킬 수 있었다. 위의 방식으로 뒤로 감기, 멈춤 후 재생 등의 기능들을 수행하도록 개발하여 기존 talkback을 개선할 수 있었다.