**DoAjou**

아주인을 위한

수강신청 도우미 웹어플리케이션

201220961 김기홍

201220962 이인태

201321001 박승현

201421019 고보원

**1. 프로젝트 개요**

**1.1. 프로젝트 선정 배경**

수강신청을 하기 위해서 해줘야 할 작업은 수강할 과목 정하기, 시간표 만들기, 수강신청 서버 시각 주시하고 있기, 쉽게 복사할 수 있도록 수강할 과목의 과목코드 적어 놓기(모두는 아니지만 거의 대부분의 학생들이 사용하는 방식) 등이 있다. 이미 이런 작업들에 익숙해져 있는 아주대 학생들은 큰 불편을 느끼지 못할 수도 있다. 하지만 우리는 분명 불편하다고 생각하는 학생들 역시 있을 것이고, 수강신청을 처음 해보는 편입생/신입생들에게 불친절한 방식임은 틀림없다고 판단했다. 그래서 기존처럼 원시적인 방식 말고 좀 더 세련된 방식으로 통합된, 그리고 더 향상된 서비스를 제공해보면 좋겠다고 생각했고, 프로젝트의 주제로 DoAjou(아주인을 위한 수강신청 도우미 웹어플리케이션)를 선정했다.

**1.2. 기대효과**

DoAjou는 실제 수강신청 자체는 지원하지 않는다. 다만 실제 수강신청 자체를 하기 위해 아주대 학생이 해야 할 모든 것들을 제공하는 통합된 하나의 서비스이다. 따라서 아주대 학생들은 더 이상 메모장에 과목코드를 적어 놓고 서버시각을 띄울 필요가 없으며, 예비수강신청 기간이 지난 후에도 자유롭게 수강 신청할 목록을 편집하고 시간표로 실시간 확인할 수 있다.

**2. 프로젝트 구현**

**2.1. 목표**

DoAjou는 Single Page Application이며 기능에 따른 2개의 탭이 있다. 두 탭은 동일한 계정 내에서 서로 연동되는 기능을 갖는다. 첫번째 탭인 ‘수강신청’ 탭에서는 수강신청 당일 날 수강신청에 도움이 되는 여러가지 기능들을 제공하는 것이 목표이다. 두번째 탭인 ‘시간표 조회’ 탭에서는 수강신청 이전에 교과목 정보를 기존의 방식보다 더 효율적이고 다양하게 얻을 수 있도록 하는 것과 얻은 교과목 정보를 이용하여 예비 수강신청 목록에 대한 시간표를 쉽게 만들 수 있는 기능을 제공하는 것이 목표이다.

**2.2. 기능 및 요구사항**

◎ 로그인/로그아웃/회원가입

1) DoAjou의 모든 기능은 로그인한 후에 이용 가능해야 한다.

2) 사용자는 id, password를 통해 로그인할 수 있어야 한다.

3) 사용자는 개개인의 계정마다 자신만의 정보를 가지고 있어야 한다.

4) 사용자는 로그아웃 전까지 페이지 내에서 자신이 작업했던 내용들을 계속 유지할 수 있어야 한다.

5) 사용자는 로그아웃을 할 수 있어야 한다.

3-1) 시스템은 사용자가 로그아웃을 할 경우 세션을 종료해야 한다.

6) 사용자는 회원가입을 할 수 있어야 한다.

6-1) 시스템은 사용자가 회원가입 시에 입력한 아이디의 중복을 체크할 수 있어야 한다.

6-2) 시스템은 사용자가 회원가입 시에 비밀번호를 2번 입력하도록 하여 비밀번호의 정확성을 체크할 수 있어야 한다.

◎ 수강신청 리스트 구성

1) 사용자는 ‘시간표 조회’ 탭에서 등록한 과목에 대한 정보를 ‘수강신청’ 탭의 ‘수강신청 리스트’에서 볼 수 있다.

1-1) 사용자는 ‘수강신청 리스트’ 각각의 항목에 존재하는 ‘복사’ 버튼을 이용하여 해당 과목의 과목번호를 클립보드에 복사할 수 있다.

1-2) 사용자는 ‘수강신청 리스트’ 각각의 항목에 존재하는 ‘삭제’ 버튼을 이용하여 해당 과목을 ‘수강신청 리스트’에서 삭제할 수 있다.

1-2-1) 시스템은 사용자가 ‘삭제’ 버튼을 이용하여 해당 과목을 수강신청 리스트에서 삭제했더라도 ‘시간표 조회’ 페이지의 리스트에는 반영되지 않도록 해야 한다.

2) 사용자는 ‘교과목’과 ‘과목 번호’를 입력한 후 ‘추가’ 버튼을 눌러서 ‘수강신청 리스트’에 과목을 추가할 수 있다.

2-1) 시스템은 사용자가 ‘추가’ 버튼을 이용하여 ‘수강신청 리스트’에 추가하려는 과목의 ‘과목 번호’가 DB에 존재하는 값인지 확인할 수 있어야 한다.

2-2) 시스템은 사용자가 ‘추가’ 버튼을 이용하여 ‘수강신청 리스트’에 추가하려는 과목의 시간이 ‘시간표 조회’ 페이지의 목록에 추가되어 있는 과목들의 시간과 중복되는지 확인할 수 있어야 한다.

2-3) 시스템은 사용자가 ‘추가’ 버튼을 이용하여 해당 내용을 ‘수강신청 리스트’에 추가했더라도 ‘시간표 조회’ 탭의 목록에는 반영되지 않도록 해야 한다.

3) 사용자는 ‘수강신청 리스트’의 과목을 클릭한 후 상단의 화살표를 클릭하여 ‘수강신청 리스트’ 내의 순서를 변경할 수 있어야 한다.

4) 시스템은 사용자의 로그인 세션이 유지되는 동안 ‘수강신청 리스트’를 보존해야 한다.

◎ 수강신청 페이지 시간 확인

1) 사용자는 ‘수강신청’ 탭에서 현재 시각을 볼 수 있어야 한다.

◎ 알람 설정

1) 사용자는 ‘hour’, ‘min’, ‘sec’을 입력한 후 ‘SET’ 버튼을 눌러서 현재 시각이 ‘hour : min : sec’이 되는 순간을 알림 받을 수 있어야 한다.

1-1) 시스템은 사용자가 설정한 알람 시각 값이 현재 시각보다 과거인지 확인할 수 있어야 한다.

1-2) 시스템은 사용자가 설정한 알람 시각 값이 유효한 시각 값인지 확인할 수 있어야 한다.

◎ 교과목 조회

1) 사용자는 6개의 항목으로 수강과목을 조회할 수 있어야 한다.

2) 사용자는 교과구분에 따라 수강과목을 조회할 수 있어야 한다.

3) 사용자는 전공에 따라 수강과목을 조회할 수 있어야 한다.

4) 사용자는 요일에 따라 수강과목을 조회할 수 있어야 한다.

5) 사용자는 강의 교시에 따라 수강과목을 조회할 수 있어야 한다.

6) 사용자는 과목명에 따라 수강과목을 조회할 수 있어야 한다.

7) 사용자는 과목명의 일부를 입력한 후 조회버튼을 클릭해도 원하는 과목을 조회할 수 있어야 한다.

8) 사용자는 교수명에 따라 수강과목을 조회할 수 있어야 한다.

9) 사용자는 교수명의 일부를 입력한 후 조회버튼을 클릭해도 원하는 과목을 조회할 수 있어야 한다.

10) 사용자는 6개의 항목들을 중복 사용하여 수강과목들을 조회할 수 있어야 한다.

◎ 시간표 편성

1) 사용자는 조회한 수강과목들 중에서 임의의 수강과목을 선택하여 수강신청항목에 등록시킬 수 있어야 한다.

2) 사용자는 시간이 겹치는 과목들을 동시에 수강신청항목에 등록시킬 수 없어야 한다.

3) 사용자는 수강신청항목 상에 등록 되어있는 수강과목들을 삭제할 수 있어야 한다.

4) 사용자는 수강신청항목에 있는 수강과목들을 시간표상에서 확인할 수 있어야 한다.

4-1) 수강신청항목에 추가되는 수강과목들은 시간표상에 추가되어야 한다.

4-2) 수강신청항목에서 삭제되는 수강과목들은 시간표상에서 삭제되어야 한다.

**3. 프로젝트의 Challenge**

◎ AWS를 이용한 서버 구축

- 아마존 웹 서비스의 EC2를 이용하여 인스턴스 생성 후, 리눅스를 기반으로 Nodejs, Express, MongoDB가 동작할 수 있도록 시스템을 구성하였다.

◎ 수강 정보 데이터베이스 구축

- 아주대학교 수강정보 데이터를 얻을 수 있거나 혹은 사용할 수 있는 Open API가 있는지 아주대학교 교무팀 및 전산실에 문의했지만, 공개하지 않는다고 하여 대안으로 직접 MongoDB에 몇 개의 수강정보 데이터를 간단하게 입력하여 데이터베이스를 구축하였다.

◎ Github을 통한 협업

- 팀원들 간의 태스크를 정확하게 분배하여, 서로 침범하지 않도록 개발 진행하였다. 한 가지의 기능이 개발될 때마다 PULL과 PUSH를 실행함으로써, 팀원들 간의 Conflict를 최대한 줄였다.

◎ 서버 시간 가져오기

- 기존 계획은 아주대 수강신청 페이지(http://sugang.ajou.ac.kr)의 서버 시간을 가져오는 것이었다. 그러기 위해서는 해당 페이지의 표준 시간과 HTTP 메시지의 latency, NTP 서버 등 다양한 것을 고려해야 하는데, 이를 해결하지 못했다.

- 그래서 한국의 표준 시간을 가져오는 것으로 대체하였다.

◎ SSL, SSL 인증서를 활용한 HTTPS를 통한 요청 처리

- AWS로 서버를 구축하면서, 서버와 클라이언트가 통신할 때에 필요한 보안에 대해서도 고려하여 시스템을 구축하였다.

- Letsencrpyt를 이용하여 SSL인증서를 제공받았으며, ssl과 nginx를 통하여 port#443(HTTPS)를 통한 통신이 가능하도록 하였고, http를 통한 접속은 자동으로 https로 redirect되도록 설정해서 보안성을 높일 수 있도록 했다.

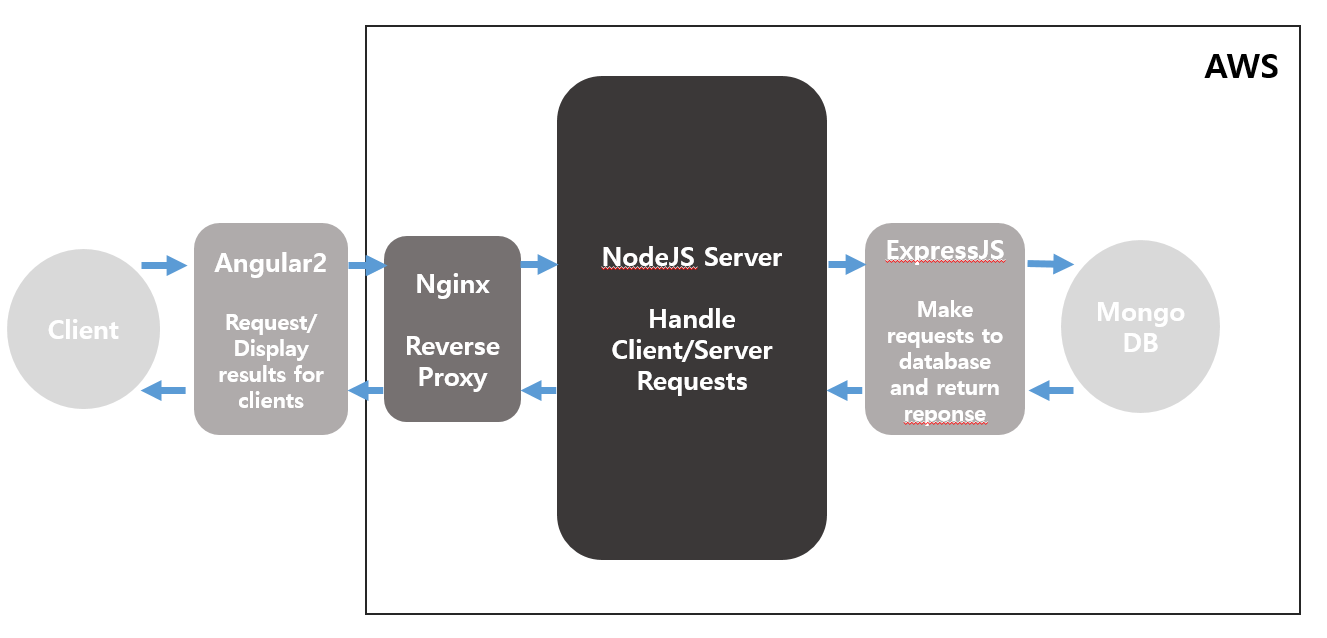
◎ Express-Session을 사용한 Session 유지

- Express에서 제공하는 ‘express-session’을 사용하여 어려움을 해결하였다. 들어오는 요청의 Session마다 서버가 사용자의 userID를 저장하여 서버는 클라이언트를 식별할 수 있다.

- 로그인 시 서버는 들어오는 요청의 Session에 사용자의 userID를 저장한다. 사용자가 로그인 후 새로 고침이나 새 창을 띄워도 서버는 session에 저장된 userID로 별다른 로그인 없이 사용자를 식별할 수 있다.

**4. 프로젝트 시스템 구조**

아래 그림은 본 프로젝트의 시스템 구조를 나타낸다.



1. 기본적으로 MEAN 스택을 구조로 구성하였다.

2. AWS EC2 인스턴스의 리눅스 환경에서 서버가 작동하고 있으며, 사용자는 [tonymony.com](https://tonymony.com), [www.tonymony.com](http://www.tonymony.com) 을 통해서 웹 어플리케이션에 접속할 수 있다.

3. Client는 서버에 port#443과 port#80을 통해서 접근할 수 있다. SSL를 활용한 HTTPS 요청을 가능하게 함으로써 보안적인 이슈를 향상시켰다.

4.. Client의 서버로의 접근은 가장 먼저 Nginx를 거친다. Nginx를 활용하여 리버스 프록시를 둠으로써, 보안적인 부분을 향상시키고, 동시적으로 발생하는 요청에도 대비할 수 있도록 하였다.