**프로젝트 설계실습 최종 작품 보고서**

**<제출할 자료 목록>**

**1. 졸업 작품 발표회 때 발표한 발표 자료**

**2. 프로젝트를 개발할 때 사용한 라이브러리를 포함한 전체 개발 코드**

**--> 한글 이름을 사용했다면 이를 모두 영문으로 바꾼 후 압축하세요.**

**3. 최종 작품 보고서**

**4. 세 가지 자료를 압축해서 교수의 이메일로 전송하세요.**

**<최종 작품 보고서>**

0. 팀원 정보 (이름, 학번, 이메일, 연락가능 번호, 사진, 팀에서의 역할), 팀명/프로젝트명, 제출일자 등 팀 및 팀원 기본 정보

- **팀명/프로젝트명 : Medipass**

**- 제출일자 : 2016/6/22**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 이름 | 사진 | 학번 | 이메일 | 전화번호 |
| 신나라 |  | 2013111539 | [robotluv1226@naver.com](mailto:robotluv1226@naver.com) | 010-6669-6985 |
| 유소현 |  | 2013111544 | [gusthdb94@naver.com](mailto:gusthdb94@naver.com) | 010-3690-2687 |

**1. 개요**

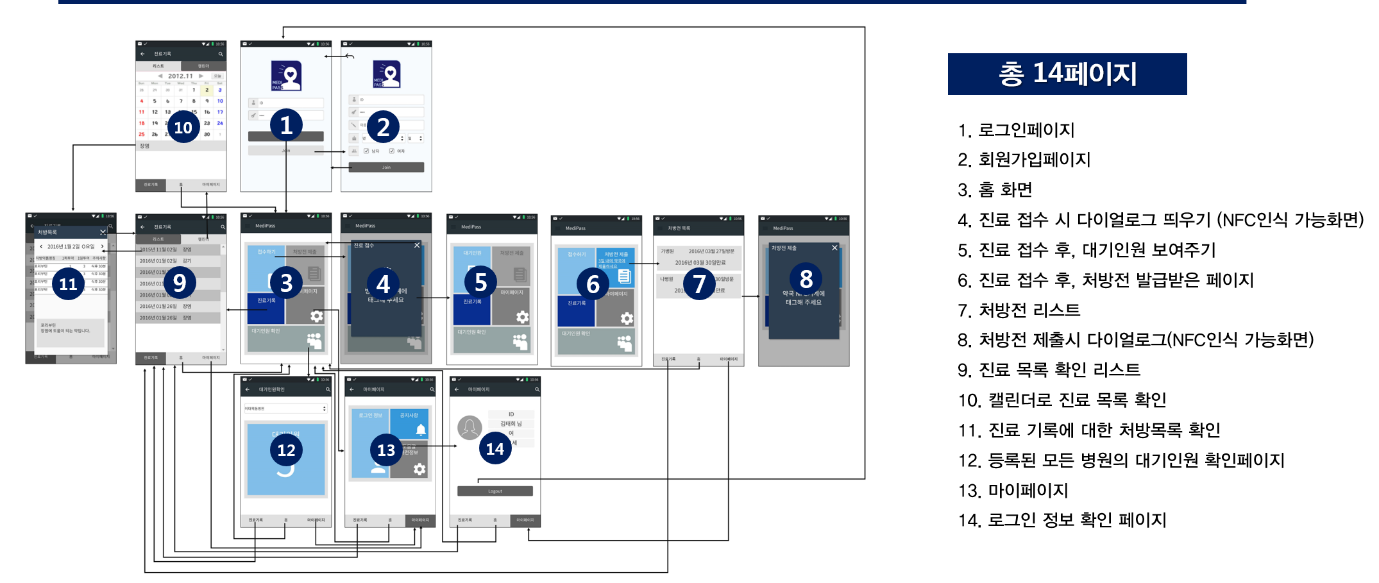
스마트폰 하나로 ‘NFC태그’라는 행위를 통하여 진료 접수부터 처방전 제출까지 한 번에 진행하고 본인의 진료/처방 기록을 관리할 수 있습니다. 따라서 종이 처방전의 감소를 통한 종이 절약과 등록/접수 시간을 단축시키며 환자의 알 권리를 보장할 수 있는 장점이 있습니다.

특정 장소에 대한 불편함을 IT기술로 해결해보고자, 장소에 대한 불편함을 찾기 시작하였습니다. 현재 병원의 시스템은 구두로 진료 접수를 하고, 진료를 받은 후에는 종이 처방번을 발급받아 약국에 제출하는 시스템입니다. 따라서 'Medipass'를 통하여 접수 시간 단축과, 종이절약을 해보고자 했습니다.

앱의 개발 과정은 ‘Oven’으로 프로토타이핑을 진행하면서 필요기술을 함께 조사하였습니다. 이후 개발 우선순위 지정하고 앱과 웹에 필요한 DB 설계 및 구축을 완료하였습니다. 최종적으로는 클라이언트단(안드로이드/웹)과 서버개발을 완료하였습니다. github를 이용하여 형상관리도 함께 해왔습니다.

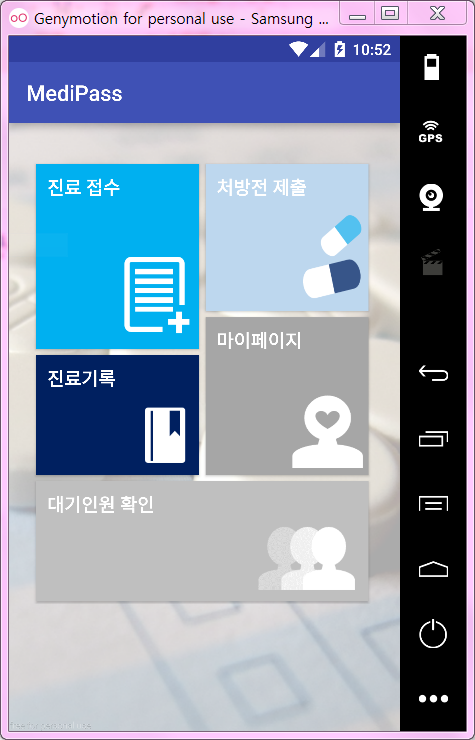
**2. 앱의 구성**

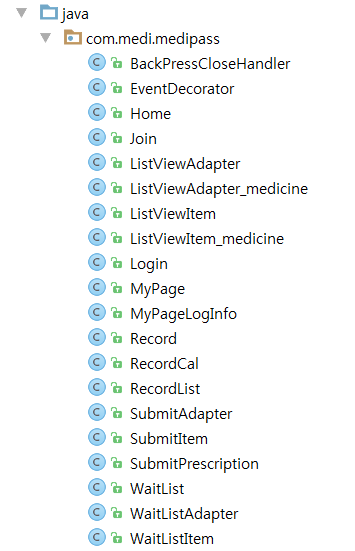
**2.1 앱에서 사용한 View와 View의 구성.**



1. 첫 화면은 (1)로그인 페이지입니다. 로그인 페이지에서 Join버튼을 누르면, (2)회원가입 페이지가 나옵니다. 회원가입을 완료한 후, 로그인을 이용할 수 있습니다.
2. (1)페이지에서 로그인 버튼을 누르면 (3)홈 페이지가 나옵니다.
3. (3)홈 페이지에서 진료 접수 버튼을 누르면, (4)진료 접수 다이얼로그가 뜹니다. 다이얼로그가 있는 상태에서 병원의nfc에 태그를 하면 진료 접수가 완료되고, (5)번의 대기인원 확인 페이지가 나옵니다.
4. 나의 진료가 끝나면, (6)번 페이지로, 홈 화면이 뜨고 처방전 확인이 가능합니다.
5. (6)의 홈 페이지에서 처방전 제출 버튼을 누르면, (7)번의 처방전 리스트 페이지를 볼 수 있습니다.
6. (7)번 페이지에 있는 처방전 중, 제출 할 처방전을 클릭하면, (8)번의 처방전을 제출하라는 다이얼로그가 뜹니다. 다이얼로그가 있는 상태에서 약국의 nfc에 태그를 하면 처방전 제출이 완료되고, 리스트가 보이지 않습니다.
7. 다시, (3)홈 페이지로 돌아와서 진료 기록 버튼을 클릭하면, (9)번의 진료 목록 확인 리스트 페이지가 나옵니다. 이는 환자의 과거 진료 기록들이다. 그 기록 중 하나를 클릭하면, 그 진료에 대한 처방목록인 (11)번 페이지가 나옵니다.
8. (9)번 리스트 페이지의 상단에, 캘린더 탭을 클릭하면 (10)번의 캘린더로 진료목록 확인하기 페이지로 들어갈 수 있습니다. (10)번 캘린더에서도 진료 받았던 날을 클릭하면, 진료 기록과 그 처방기록을 확인할 수 있습니다.
9. (3)번 홈 페이지에서 대기인원 확인을 클릭하면, (12)번 화면처럼 등록된 모든 병원의 대기인원을 확인할 수 있습니다.

**2.2 주요 클래스 및 사용한 외부 라이브러리 소개**

**<클래스 목록> - 총 20개의 클래스**



Home.java

BackPressCloseHandler.java

SubmitAdapter.java

SubmitItem.java

Submit Prescription.java

Home.java

WaitList.java

WaitListAdapter.java

WaitListItem.java

MyPage.java

MypageLogInfo.java

Record.java

RecordCal.java

RecordList.java

ListViewAdapter.java

ListViewAdapter\_medicine.java

ListViewItem.java

ListViewItem\_medicine.java

\* **HojungNFCLibrary\_v01**

nfc에 태그하므로써, nfc에 값을 쓰거나 읽도록 하는 라이브러리입니다.

(오픈되어있는 라이브러리가 아닌, 친구의 자체 라이브러리라 외부 공유에는 조금 불편할 수 있는 사항입니다.)

* HojungNFCReadLibrary **hojungNFCReadLibrary**: 라이브러리 사용을 선언합니다.
* **public void** onReceiveMessage(NfcModel[] models): nfc에서 읽은 값을 받는 함수입니다.
* **private void** initNFC():  앱이 실행될때 NFC 어댑터를 활성화 합니다
* **public void** onResume():  앱이 실행될때 NFC 어댑터를 활성화 합니다
* **protected void** onPause():  앱이 종료될때 NFC 어댑터를 비활성화 합니다
* **public void** onNewIntent(Intent intent): NFC 데이터 활성화에 필요한 인텐트를 생성합니다.

**\* com.prolificinteractive:material-calendarview:1.2.0**

달력을 보여주는 라이브러리입니다. 아래 참고 링크 캘린더에 설명 링크가 있습니다

* **com.prolificinteractive.materialcalendarview.MaterialCalendarView**: 캘린더를 보여줄 xml파일의 레이아웃에 선언합니다.
* **private static final** DateFormat ***FORMATTER*** = SimpleDateFormat.*getDateInstance*();: 날짜 형식을 설정합니다.
* @Bind(R.id.***calendarView***)
* MaterialCalendarView **widget**;: 아래 ButterKnife에서 설명하겠습니다.
* **private** String getSelectedDatesString(): 선택된 날의 String 값을 반환하는 함수입니다.
* **public void** onDateSelected(@NonNull MaterialCalendarView widget, @Nullable CalendarDay date, **boolean** selected): 선택된 날에 할 작업을 작성하는 함수입니다.
* **widget**.setOnDateChangedListener(**this**);

**\* Butterknife**

 findViewById(viewID)를 하는 것을 내부적으로 ButterKnife가 도와줍니다.

* ButterKnife.*bind*(**this**, view);: 바인드를 해준다.

**\* Home.java**

이 파일은 홈 화면을 띄워주고, 진료 접수를 가능하게 하는 클래스입니다. onCreate에서 화면을 띄우고 버튼 클릭을 가능하게 하며, 해당 버튼에 대한 각각의 기능이 구현되어있습니다. 특히 진료접수에 대해서는 다이얼로그를 띄우고, registerWatiList()함수를 호출하여 서버의 대기목록에 추가한 후,showWaitNum(), myWaitNum(), updateWait()함수를 통하여 대기번호를 보여줍니다.

* **public void** onCreate(Bundle savedInstanceState): 첫 화면을 실행시킵니다.
* **public void** onBackPressed(): 뒤로가기 버튼 클릭시, 두번 눌러야 종료되게 합니다.
* **public void** registerWaitList(): php와 연동하여 대기목록 테이블에 태그한 환자 정보를 insert합니다.
* **public void** showWaitNum(): php와 연동하여 접수를 진행한 환자 본인의 대기번호를 보여줍니다.
* **public void** myWaitNum(): php와 연동하여 접수를 진행한 환자 본인의 대기번호를 저장합니다.
* **public void** updateWait(): php와 연동하여 접수를 진행한 환자 본인의 대기번호를 업데이트한다. 앞의 환자가 진료를 마치면, 대기번호가 줄어듭니다.
* **class** GettingPHP **extends** AsyncTask<String, Integer, String>: php와 통신하기 위한 클래스입니다.
* **protected** String doInBackground(String... params): php에서 데이터를 읽어오는 함수입니다. (여기에서는 nfc에 있는 병원 코드를 전송하여, 나의 대기번호를 받아옵니다.)
* **protected void** onPostExecute(String str): php에서 받아온 데이터를 이용하여 원하는 작업을 합니다. (여기에서는 대기번호를 텍스트에 보여줍니다.)

**\* SubmitPrescription.java**

이 파일은 발급받은 처방전을 약국에 nfc태그를 통하여 제출하는 역할을 합니다. onCreate에서 다이얼로그를 띄워서 nfc를 태그하여 submitPrescription()을 호출하여 처방전을 제출합니다. 그리고 registePharm()을 통하여 약국의 대기목록에 태그한 환자를 삽입합니다.

* **public void** onCreate(Bundle savedInstanceState): nfc를 태그하므로써 처방전을 제출하는 함수입니다.
* **public void** onRefresh(): 화면을 새로고침하는 함수입니다. 리스트에 변동이 생긴경우, 변경된 화면이 보입니다.
* **public void** submitPrescription(): php와 연동하여 nfc에 태그한 사용자의 처방전을 제출하도록 합니다.
* **public void** registePharm(): php와 연동하여 nfc에 태그한 사용자를 약국의 대기목록에 접수합니다.
* **class** GettingPHP **extends** AsyncTask<String, Integer, String>: php와 통신하기 위한 클래스입니다.
* **protected** String doInBackground(String... params): php에서 데이터를 읽어오는 함수입니다. (여기에서는 nfc에 있는 약국 코드를 전송하여, 나의 대기번호를 받아오고, 처방전을 접수할 수 있도록 합니다.)
* **protected void** onPostExecute(String str): php에서 받아온 데이터를 이용하여 원하는 작업을 합니다. (여기에서는 처방전의 각 값을 리스트로 만듭니다.)

**\* SubmitItem.java**

* **public class** SubmitItem: 리스트에 보여주기위한 아이템을 담은 클래스입니다.

**\* SubmitAdapter.java**

이 파일은 서버에서 받은 처방전 정보를 리스트로 만들고, 정보를 화면에 보여주는 역할을 합니다.

* **public int** getCount(): 처방전의 개수만큼 화면에 보여줘야 하므로, 리스트의 사이즈를 측정합니다.
* **public void** addItem(String vTitle, String vDate, String vHospital, String vDisease, String vPresnum): 처방전에 들어갈 정보들을 받아서 값을 추가합니다.
* **public** View getView(**int** position, View convertView, ViewGroup parent): xml과 연동하여 받아온 값을 화면에 보여줍니다.

**\* WaitList.java**

이 파일은 등록되어있는 모든 병원의 대기인원을 화면에 보여주는 파일입니다.

* **public void** onCreate(Bundle savedInstanceState): 각 병원들의 대기인원을 보여줍니다.
* **lass** GettingPHP **extends** AsyncTask<String, Integer, String>: php와 통신하기 위한 클래스입니다.
* **protected** String doInBackground(String... params): php에서 데이터를 읽어오는 함수입니다. (여기에서는 각 병원의 대기인원을 읽어옵니다.)
* **protected void** onPostExecute(String str): php에서 받아온 데이터를 이용하여 원하는 작업을 합니다. (여기에서는 병원의 이름과 대기인원 값을 리스트로 만듭니다.)

**\* WaitListItem.java**

**public class** WaitListItem: 리스트에 보여주기위한 아이템을 담은 클래스입니다.

**\* WaitListAdapter.java**

이 파일은 서버에서 받은 병원별 대기인원 정보를 리스트로 만들고, 정보를 화면에 보여주는 역할을 합니다.

* **public int** getCount(): 처방전의 개수만큼 화면에 보여줘야 하므로, 리스트의 사이즈를 측정합니다.
* **public void** addItem(String hosname, String waitnum): 처방전에 들어갈 정보들을 받아서 값을 추가합니다.
* **public** View getView(**int** position, View convertView, ViewGroup parent): xml과 연동하여 받아온 값을 화면에 보여줍니다.

**\* BackPressCloaseHandler.java**

이 파일은 사용자가 뒤로가기 버튼을 누르면 ‘뒤로 버튼을 한번 더 누르시면 종료됩니다’라는 메시지를 띄운 후, 2초내에 연속해서 누르면 앱을 종료시키는 역할을 합니다.

**\* EventDecorator.java**

**\* ListViewAdapter.java**

이 파일은 서버에서 받은 진료 목록을 화면에 리스트의 형식으로 띄우는 역할을 합니다.

* **private** ArrayList<ListViewItem> **listViewItemList** = **new** ArrayList<ListViewItem>();
* **public void** addItem(String date, String disName) : 서버에서 데이터를 읽어와 addItem에 접근하여 리스트에 내용을 추가합니다.
* **public void** init() : RecordList와 RecordCal이 같은 어댑터를 사용하기 때문에 호출될때마다 초기화를 통해 직전의 데이터를 없애기 위해 필요한 메소드입니다.

**\* ListViewItem.java**

이 파일은 ‘진료일자 – 병명’의 형태로 리스트의 정보를 가지고 있는 객체입니다.

**\* ListViewAdapter\_medicine.java**

이 파일은 서버에서 처방받은 약 목록을 읽어와 리스트의 형식으로 띄우는 역할을 합니다. (ListViewAdapter과는 리스트의 디자인이 달라 따로 작성하였습니다.)

* **private** ArrayList<ListViewItem\_medicine> **listViewItemList\_medicine** = **new** ArrayList<ListViewItem\_medicine>();
* **public void** addItem(String medName, **int** onceNum, **int** dayNum, String notice)

**\* ListViewItem\_medicine.java**

이 파일은 ‘약이름-하루몇번-몇일-주의사항’의 형태로 리스트의 정보를 가지고 있는 객체입니다.

**\* Record.java**

이 파일은 ‘리스트’, ‘캘린더’버튼에 따라 RecordList.java, RecordCal.java를 프레그먼트로 띄워주는 파일입니다. 하단에는 하단바를 가지고 있어 ‘진료기록’, ‘홈’, ‘마이페이지’버튼을 누르면 각각 해당 Activity로 CLEAR\_TOP, SINGLE\_TOP Flag와 함께 Intent를 시켜줍니다.

**\* RecordList.java (extends Fragment)**

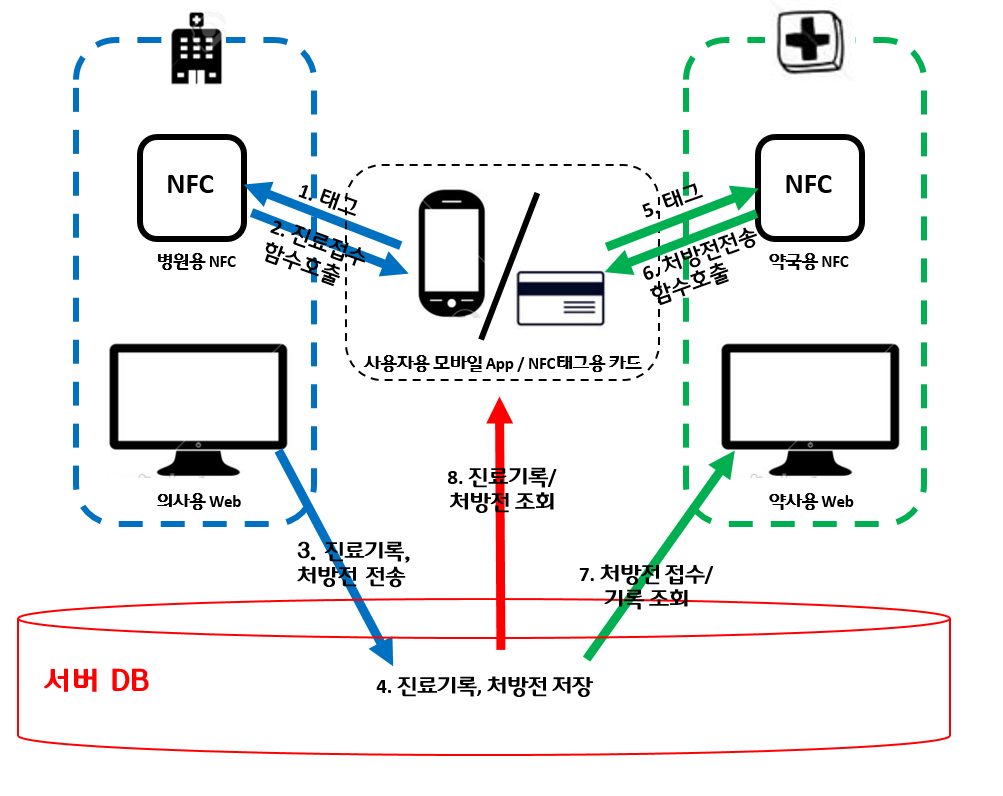
**private static** ArrayList<String> *dateArrayList* = **new** ArrayList<String>();  
**private static** ArrayList<String> *nameArrayList* = **new** ArrayList<String>();

**\* RecordCal.java (extends Fragment)**

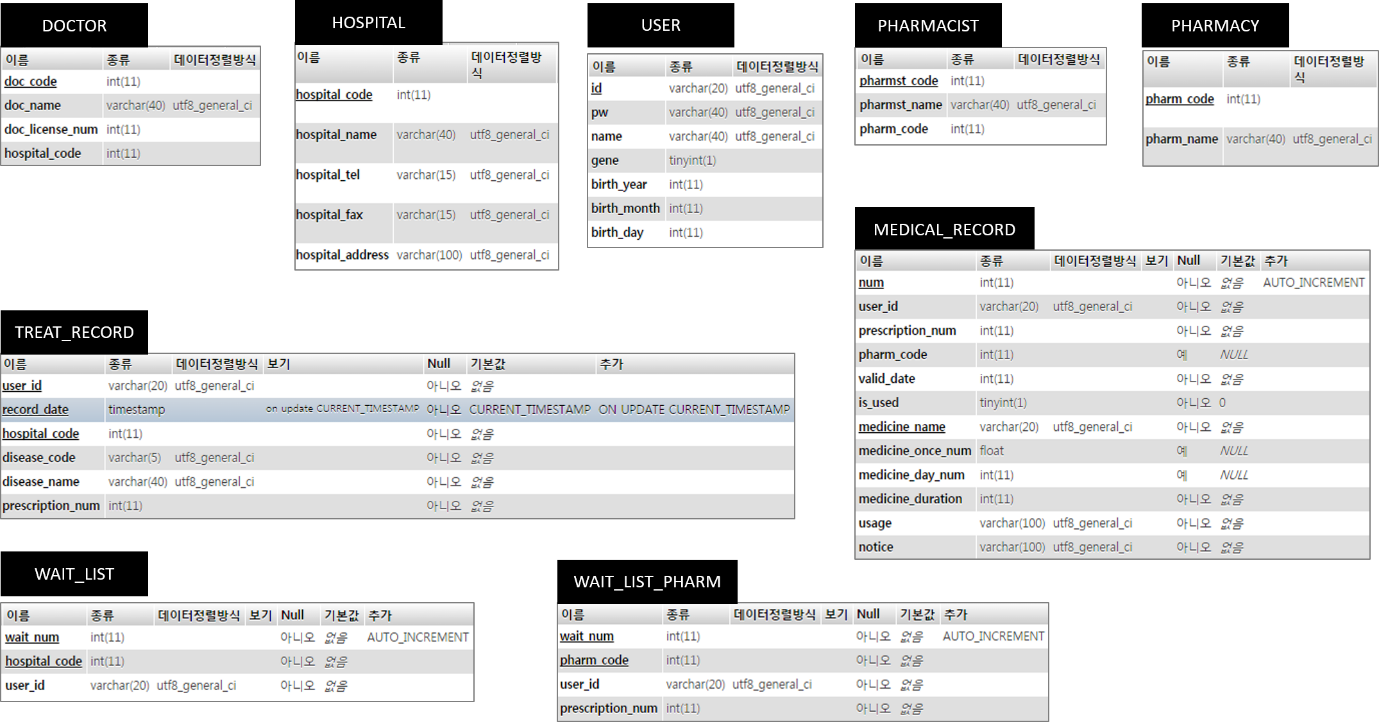
* **public void** onDateSelected(@NonNull MaterialCalendarView widget, @Nullable CalendarDay date, **boolean** selected) : 달력에서 날짜 선택 시 서버에서 읽어온 진료목록 정보 중 해당 날짜와 같은 날짜의 진료목록만 리스트화 합니다.
* **private** String getSelectedDatesString() : 달력에서 날짜 선택 시 해당 날짜정보를 String의 형태로 반환합니다.

**2.3 서버 환경 요약**

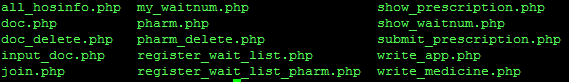
**<솔루션 구성도>**



**<DB Table> -총 9개**

****

**<php파일 목록> -총 15개**



툴은 ‘putty’를 사용하였으며, condi.swu.ac.kr(2002)서버를 사용하였습니다.

**\* register\_wait\_list.php (Home.java): 병원 대기목록에 nfc태그한 환자를 추가하는 파일입니다.**

$sql = "INSERT INTO WAIT\_LIST(hospital\_code, user\_id) VALUES ('$hos\_code','$id')"; : nfc에서 읽어온 병원코드와, 로그인된 환자의 ID를 WAIT\_LIST테이블에 삽입하라는 쿼리문입니다.

**\* my\_waitnum.php(Home.java): 태그한 환자 본인의 대기번호를 확인해오는 파일입니다.**

$sql = "SELECT wait\_num FROM WAIT\_LIST WHERE hospital\_code='$hospital\_code' AND user\_id='$id'"; : nfc에서 읽어온 병원 코드와, 환자의 id를 통하여, 환자의 대기번호를 조회하는 쿼리문입니다.

\* **show\_waitnum.php(Home.java): 데이터베이스에 있는 대기 인원을 확인하는 파일입니다.**

$sql = "SELECT \* FROM WAIT\_LIST WHERE hospital\_code='$hospital\_code' AND wait\_num<='$my\_wait\_num'"; : nfc에서 읽힌 병원의 코드와 일치하고, 나의 대기번호보다 작거나 같은 사람들의 목록을 조회합니다.

$num = mysqli\_num\_rows($result); : 조회한 쿼리문의 행 수를 세서, 대기 번호를 확인할 수 있습니다.

**\* show\_prescription.php(SubmitPrescription.java): 처방전을 조회하는 파일입니다.**

$sql = "SELECT DISTINCT M.prescription\_num, date\_format(T.record\_date, '%Y. %c. %e.') as record\_date, H.hospital\_name, T.disease\_name, M.valid\_date";

$sql .= " FROM MEDICAL\_RECORD M, TREAT\_RECORD T , HOSPITAL H ";

$sql .= " WHERE T.hospital\_code=H.hospital\_code AND M.prescription\_num=T.prescription\_num AND M.is\_used='0'"; : 처방전의 정보를 조회하는 쿼리문입니다. 조인을 한 이유는 진료정보와 처방정보, 병원정보가 각각 다른 테이블에 저장되어있어서, 조인을 하여 필요한 정보를 조회하였습니다.

**\* register\_wait\_list\_pharm.php(SubmitPrescription.java): 약국의 대기목록에 nfc태그한 환자를 추가하는 파일입니다.**

$sql = "insert into WAIT\_LIST\_PHARM(pharm\_code, user\_id, prescription\_num) values ('$pharm\_code','$id', '$presnum')"; : 약국에 가서 약국의 nfc에 태그한 즉시, nfc에 쓰여있는 약국의 고유 코드와, 사용자의 id, 처방전 번호를 이용하여 대기목록에 삽입하는 쿼리문입니다.

**\* submit\_prescription.php(SubmitPrescription.java): 약국에 nfc로 태그한 처방전을 제출하는 파일입니다.**

$sql = "UPDATE MEDICAL\_RECORD SET is\_used ='1', pharm\_code='$pharm\_code' WHERE user\_id='$id' AND prescription\_num='$presnum'"; : 태그한 즉시 nfc에서 읽은 약국의 코드와, 태그한 사용자의 정보를 이용하여, 처방전의 사용 여부인 is\_used를 1로 바꾸어 제출했다고 표시합니다. 그리고 태그된 약국의 고유 코드인 pharm\_code를 바꿔서 제출된 약국을 식별할 수 있습니다.

**\* all\_hosinfo.php(WaitList.java): 전체 병원의 목록과, 대기인원을 조회하는 파일입니다.**

$sql = "SELECT H.hospital\_name, Count(W.wait\_num) FROM HOSPITAL H, WAIT\_LIST W WHERE H.hospital\_code=W.hospital\_code GROUP BY W.hospital\_code"; : 전체 병원의 목록과 대기인원을 조회하는 쿼리문입니다. 병원 이름과, 대기 목록인 wait\_num의 행수를 카운트해서 총 인원을 조회할 수 있습니다.

**\* doc.php:** 의사용 웹을 구성하기 위한 php 파일입니다. (‘제조완료’버튼 클릭시 doc\_delete.php파일에 링크됩니다. 이 때 doc\_delete.php파일에게 받은 변수값을 사용해 오른쪽에 호출된 사람의 정보를 띄우게 됩니다.)

**\* doc\_delete.php:** doc.php파일의 대기목록의 링크를 클릭하면 이 파일로 링크가 됩니다. 해당 list에 대한 name, gender, id값을 만족시키는 WAIT\_LIST의 레코드를 지우고, 다시 doc.php에게 전달됩니다.

**\* input\_doc.php:** 의사용 웹에서 의사가 입력한 처방목록과 진료사항을 db에 저장하는 파일입니다.

**\* pharm.php:** 약사용 웹을 구현하기위한 php파일입니다. 사용자의 약의 개수만큼 append함수를 호출함으로써 약 목록을 동적으로 출력하게됩니다. (제조완료 버튼을 누르면 pharm.php파일을 호출하여 다시 새로운 환자를 받을 준비를 합니다.)

**\* pharm\_delete.php:** pharm.php파일의 대기목록의 링크를 클릭하면 이 파일로 링크가 됩니다. 해당 list에 대한 name, gender, pre\_num값을 만족시키는 WAIT\_LIST\_PHARM의 레코드를 지우고, 다시 pharm.php에게 전달됩니다.

**\* write\_app.php(RecordList.java, RecordCal.java):** 진료기록 메뉴에 필요한 병명과 진료일자를 검색하는 파일입니다.

**\* write\_medicine.php(RecordList.java, RecordCal.java):** 진료기록에 대한 상세정보로 해당 일자에 처방받은 약 목록을 검색하는 파일입니다.

**2.4 미래에 추가될 가능성이 있는 기능의 개략 소개**

여러분의 코드는 GitHub에 등록하기 바랍니다. 완성된 코드의 등록에 대한 안내는 다시 하도록 하겠습니다.

차후에 여러분 혹은 여러분의 후배들이 여러분의 코드에 새로운 기능을 추가하고자 한다면, 어떤 기능들이 추가되었으면 좋을지에 대한 생각을 간단히 요약합니다. 어떤 기능이 추가되면 더 나은 프로그램이 될 지에 대해 생각해보고 추가될 기능 및 화면 인터페이스 등을 서술하면 됩니다.

1. 병원에서의 결제와 약 비용까지 결제할 수 있는 시스템
   1. 스마트폰 하나로 의료기관에서의 절차를 한 번에 해결할 수 있다. 따라서 시간 감축의 효과를 크게 볼 수 있을 것이라 생각합니다.
2. 사용자-의사/약사 커뮤니티
   1. 사용자가 병원을 갈 수 없는 상황에 의사 또는 약사와 소통할 수 있습니다. 예를 들면, 약을 제시간에 복용하지 못했을 때의 대처법에 대하여 약사에게 문의할 수 있습니다.
3. 보안
4. 로그인/회원가입 기능: 시간상 우선순위를 맨 뒤로 정해놓아서 개발을 하지 못한 상태입니다. 현재는 php파일에 ‘so’라는 아이디로 등록을 해놓은 상태여서, 로그인 한 각자의 아이디로 접수하고 처방전을 제출할 수 있는 기능이 구현되면 좋을 것 같습니다.

**3. 참고자료**

본 프로그램을 작성하는 동안 참고했던 서적이나 Web Site, 여러분이 기존에 작성했던 코드의 경우 그 코드를 얻은 출처 등, 가능한 모든 참고 서적을 이곳에서 나열합니다. 이 내용들이 차후에 보다 나은 프로젝트 주제를 선정하기 위한 좋은 자료가 될 것입니다.

* 책 ([**Do it**!**안드로이드** 앱 프로그래밍](http://book.naver.com/bookdb/book_detail.nhn?bid=10120935), PHP 프로그래밍 입문)
* 아라비안나이트 블로그: <http://arabiannight.tistory.com/category/Android>
* 미르의 IT정복기 블로그: <http://itmir.tistory.com/category/Development/App>
* 캘린더: <http://naver.me/5LtmuZ7k>
* 리스트뷰: <http://jo.centis1504.net/?p=1009>
* 리스트뷰: <http://recipes4dev.tistory.com/43>
* 카드뷰: <https://www.binpress.com/tutorial/android-l-recyclerview-and-cardview-tutorial/156>
* 카드뷰: <https://www.simplifiedcoding.net/android-recyclerview-and-cardview-tutorial/>
* 뒤로가기 버튼 동작 시, Home으로 가기: <http://diyall.tistory.com/781>
* 뒤로가기 버튼 동작 시, Home으로 가기: <http://comxp.tistory.com/109>
* SwipeRefresh: <http://blog.miyu.kr/163>
* SwipeRefresh: <http://ggari.tistory.com/528>

**4. 소감 + 건의사항**

프로젝트를 진행하면서 느낀 점들을 각 팀원별로 작성합니다. 좋았던 점, 나빴던 점, 개선해야 할 점 등을 쓰세요. 그리고 이런 방식은 고치면 좋겠다거나, 이런 방식은 계속 유지해야 한다거나 하는 내용을 꼭 하나 이상씩 쓰세요. 또한 친구들에게 들었거나 복수전공으로 다른 과에서 수강했던 나 이외의 다른 교수님들의 프로젝트 과목과 비교하여 이랬으면 좋겠다는 생각도 있으면 적기 바랍니다. 그리고 여러분의 프로그램을 어떻게 개선하면 좋은지 써보기 바랍니다.

**공동 소감**

* 산업디자인 학과와 팀을 이뤄서 디자인+개발을 함께 진행하면 더 완성도 있는(개발 뿐 아닌, UX/UI적인 부분까지) 결과물 얻을 수 있을 것 같습니다.
* 개발자만이 아닌 디자이너와 협업하면서 실무능력도 함께 키울 수 있을 것 같습니다.

**신나라**

이번 졸업프로젝트를 통해 저학년부터 배워온 Java, DB, 알고리즘, Unix 등 여러 과목을 스스로 합쳐볼 수 있는 좋은 경험이 되었던 것 같습니다. 그리고 배운 내용에서 나아가, 심화내용들을 검색하고 질문하는 등 혼자 터득하는 법을 배울 수 있었습니다. 그리고 팀원과 함께 개발하고 고민할 수 있는 점도 좋았습니다. 다만, 학교 시스템만 가능하다면, 주제선정과 팀 선정만이라도 방학 때 완료가 되면 좋을 것 같습니다. 왜냐하면, 개강주부터 주제선정을 시작하고, 주제가 확정이 나서 개발을 시작하면 실 개발기간이 짧을 수 있기 때문입니다. 따라서 더 좋은 질을 위해서라면, 기간적인 면에서의 조절이 필요할 수도 있을 것 같습니다. 그리고 최종보고서에 참고 서적이나 자료들을 적는 란이 있는데, 개발이 거의 끝난 단계에서 이 항목을 접하다 보니 적어둔 링크가 많지 않아서, 많은 정보를 주기 어려웠습니다. 따라서 개발을 시작할 때 알려주신다면 학생들이 찾아본 링크들을 처음부터 적기 좋을 것 같습니다. 마지막으로는 학과의 ‘졸업’프로젝트이므로 수강생 모두 완성도 있는 모습을 보여줄 수 있도록 노력하는 것이 중요하다고 생각합니다. 하지만, 이번 발표회에서 보니, 그렇지 못한 팀도 보였고, 정해져있던 발표일에 불참한 팀도 있었습니다. ‘만약 P/F과목이 아니었어도 이런 모습을 보였을까’ 하는 생각도 들었습니다. 그래서 이런 팀들에게는 졸업 프로젝트의 모습을 다시 갖출 수 있는 기회를 주는 것도 모든 수강생에게 좋은 방법이 될 수 있을 것 같습니다.

**유소현**

컴퓨터학과에 진학하여 지난3년간 배운 컴퓨터지식들을 프로젝트설계실습을 통해 총정리해볼 수 있는 시간을 가져서 소중한 시간이었다고 생각합니다. 과제단위의 혼자 주먹구구로 개발하는것이 아닌 아이디어, 기획, 디자인, 세부구성까지 팀원과 상의하는 과정은 팀워크를 가짐에있어서 저의 발전에 도움이된것같습니다. 또 서로 개발한것을 하나로 합치는 과정은 다른사람의 코드를 대략적으로 보고 흐름을 읽는연습이 된것같습니다. 개발을 하면서 함께하여 좋았던점은 결과물이 개발한것의 2배로 나왔다는 것입니다. 아쉬웠던 점은 최종발표날 모든 팀의 발표를 듣지 못했다는것입니다. 컴퓨터학과의 졸업프로젝트인만큼 다른팀의 다양한 작품들을 기대하였는데 미리 정해진 발표날 발표를 하지않은 팀과 더불어 그 자격이 부족하다고 생각이되는팀은 남은학기가 없어 졸업에 직접적으로 연결되지않는이상 한번 더 개발을할 기회를 주는 것이 모두에게 공정하고 좋은 방법이라고 생각합니다.