**모바일/웹서비스 프로젝트 공통평가 01**

**수행 결과 보고서**

**학번 : 2020110556**

**이름 : SUN QIGAN**

[적/부판정 범례] ○:적합, ×:부적합, △:점검제외, N/A:해당없음

첨부 그림은 모두 문서에 포함하는 형식으로 저장 할 것

| **수업 요구사항** | | | | **이행현황** | | |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **구분** | **요구사항명** | **요구사항 내용** | **완료**  **여부** | **적/부 판정** | **관련증빙** |
| 1 | 제출문 | 폴더 구조 | - ROOT  - PhotoBlogServer[Dir]  : Django Server  - PhotoViewer[Dir]  : Android Client  -모바일/웹서비스 프로젝트 공통평가 01\_수행 결과 보고서.docx[File]  제출 : 탐색기, 파인더 또는 Git레포지토리 폴더 구조 캡처 화면 |  |  |  |
| 2 | 제출문 | Github 레파지토리 push | 아래 평가 항목에 대하여 구현 혹은 테스트를 완료하고, 캡처 이미지를 제공된 MS-Word 파일로 정리하여 Github URL을 제출 한다.  제출 : URL |  |  | URL:https://github.com/shenxuandada/web\_HW |
| 3 | 강의 자료 | 포토 블로그 – API | 강의 자료 [7페이지, POST]  제출 : 1. 작성 완료 웹페이지 작성 화면 캡처  2. 게시 완료된 페이지 화면 캡처 |  |  |  |
| 4 | 강의 자료 | 포토 블로그 – API 검증(2/3) | 강의 자료 [9페이지]  제출 : 터미널 화면 캡처 |  |  |  |
| 5 | 강의 자료 | 포토 블로그 – API 검증(3/3) | 강의 자료 [10페이지]  제출 : [codebeautify](https://codebeautify.org/jsonviewer) 서비스 테스트 화면 캡처 |  |  |  |
| 6 | 강의 자료 | Server/Client 연동 테스트 [Sockec Server 요청 메시지] | 강의 자료 [24페이지]  제출 : 수신 파일 VSCODE 편집화면캡처 |  |  |  |
| 7 | 강의 자료 | Server/Client 연동 테스트 [Server 브라우저 출력화면] | 강의 자료 [25페이지]  제출 : Server 브라우저 출력화면 (localhost) |  |  |  |
| 8 | 강의 자료 | Server/Client 연동 테스트 [Client 시작 화면] | 강의 자료 [25페이지]  제출 : Client 시작 화면 |  |  |  |
| 9 | 강의 자료 | Server/Client 연동 테스트 [Client 동기화 완료 화면] | 강의 자료 [25페이지]  제출 : Client 동기화 완료 화면(localhost 서버연동) |  |  |  |
| 10 | 강의 자료 | MainActivity 클래스 [Upload 구현, 19page] | MainActivity 클래스 [Upload 구현, 19page]  1. Upload (Hard cording) : 중  (1. 2. 중 하나만 제출 ) |  |  | public class MainActivity extends AppCompatActivity {  private static final int *MY\_PERMISSIONS\_REQUEST\_WRITE\_EXTERNAL\_STORAGE* = 1;  ImageView imgView;  TextView textView;  String site\_url = "https://button.pythonanywhere.com/";  JSONObject post\_json;  String imageUrl = null;  Bitmap bmImg = null;  CloadImage taskDownload;   @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_main*);  textView = findViewById(R.id.*textView*);  }   public void onClickUpload(View view) {  Toast.*makeText*(this, "Upload button clicked", Toast.*LENGTH\_SHORT*).show();  }    public void onClickDownload(View v) {  if (taskDownload != null && taskDownload.getStatus() == AsyncTask.Status.*RUNNING*) {  taskDownload.cancel(true);  }  taskDownload = new CloadImage();  taskDownload.execute(site\_url + "/api\_root/Post/");  Toast.*makeText*(getApplicationContext(), "Download", Toast.*LENGTH\_LONG*).show();  }   private class CloadImage extends AsyncTask<String, Integer, List<Bitmap>> {  @Override  protected List<Bitmap> doInBackground(String... urls) {  List<Bitmap> bitmapList = new ArrayList<>();  try {  String apiUrl = urls[0];  String token = "bf46b8f9337d1d27b4ef2511514c798be1a954b8";  URL urlAPI = new URL(apiUrl);  HttpURLConnection conn = (HttpURLConnection) urlAPI.openConnection();  conn.setRequestProperty("Authorization", "Token " + token);  conn.setRequestMethod("GET");  conn.setConnectTimeout(3000);  conn.setReadTimeout(3000);   int responseCode = conn.getResponseCode();  if (responseCode == HttpURLConnection.*HTTP\_OK*) {  InputStream is = conn.getInputStream();  BufferedReader reader = new BufferedReader(new InputStreamReader(is));  StringBuilder result = new StringBuilder();  String line;  while ((line = reader.readLine()) != null) {  result.append(line);  }  is.close();   String strJson = result.toString();  JSONArray aryJson = new JSONArray(strJson);  for (int i = 0; i < aryJson.length(); i++) {  post\_json = aryJson.getJSONObject(i);  imageUrl = post\_json.getString("image");  if (!imageUrl.equals("")) {  URL myImageUrl = new URL(imageUrl);  conn = (HttpURLConnection) myImageUrl.openConnection();  InputStream imgStream = conn.getInputStream();  Bitmap imageBitmap = BitmapFactory.*decodeStream*(imgStream);  bitmapList.add(imageBitmap);  imgStream.close();  }  }  }  } catch (IOException | JSONException e) {  e.printStackTrace();  }  return bitmapList;  }   @Override  protected void onPostExecute(List<Bitmap> images) {  if (images.isEmpty()) {  textView.setText("불러올 이미지가 없습니다.");  } else {  textView.setText("이미지 로드 성공!");  RecyclerView recyclerView = findViewById(R.id.*recyclerView*);  ImageAdapter adapter = new ImageAdapter(images);  recyclerView.setLayoutManager(new LinearLayoutManager(MainActivity.this));  recyclerView.setAdapter(adapter);  }  }  } } |
| MainActivity 클래스 [Upload 구현, 19page]  2. Upload (Post에 따른 UI제공) : 상  (1. 2. 중 하나만 제출 |  |  |  |
| 11 |  | Pythonanywhere 클라우드상 동작 | Pythonanywhere 클라우드상 서버 동작  제출 : Server 브라우저 출력화면 (클라우드 서버연동) |  |  |  |
| 12 |  | Pythonanywhere 클라우드상 동작 | Pythonanywhere 클라우드상 서버 동작  제출 : Client 동기화 완료 화면 (클라우드 서버연동) |  |  |  |
| 13 |  | 기타 기획 되지 않은 Client 편의 기능 [최대 5개] | 제출 : 해당 기능 정의, 설명 및 동작 증빙 화면 캡처  (서버/클라이언트 각각 구현이 필요한 경우 별도 요구사항으로 등록 함) |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |