**캡스톤 디자인 I**

**종합설계 프로젝트**

|  |  |
| --- | --- |
| 프로젝트 명 | COWAY |
| 팀 명 | COWAY |
| 문서 제목 | 중간보고서 |

|  |  |
| --- | --- |
| **Version** | 1.3 |
| **Date** | 2018-04-08 |

|  |  |
| --- | --- |
| **팀원** | 신종혁 (조장) |
| 박세용 |
| 이성준 |
| 최인정 |
|  |
| **지도교수** | 윤상민 교수 |

|  |
| --- |
| **CONFIDENTIALITY/SECURITY WARNING**  이 문서에 포함되어 있는 정보는 국민대학교 전자정보통신대학 컴퓨터공학부 및 컴퓨터공학부 개설 교과목 캡스톤 디자인I 수강 학생 중 프로젝트 “코웨이(Coding Wherever Easily)”를 수행하는 팀 “Coway”의 팀원들의 자산입니다. 국민대학교 컴퓨터공학부 및 팀 “Coway”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다. |

**문서 정보 / 수정 내역**

|  |  |
| --- | --- |
| **Filename** | 중간보고서-COWAY.doc |
| **원안작성자** | 신종혁, 박세용, 이성준, 최인정 |
| **수정작업자** | 신종혁, 박세용, 이성준, 최인정 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
| 2018-04-07 | 박세용 | 1.0 | 최초 작성 |  |
| 2018-04-08 | 신종혁 | 1.1 | 내용 추가 | 영상처리 부분 내용 추가 |
| 2018-04-08 | 이성준 | 1.2 | 내용 추가 | 서버, 인터프리터 내용 추가 |
| 2018-04-08 | 최인정 | 1.3 | 내용 추가 | 안드로이드 FE 부분 내용 추가 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

내용

[1 프로젝트 목표 4](#_Toc510991141)

[1.1 프로젝트 설명 4](#_Toc510991142)

[1.2 프로젝트 추진 배경 및 필요성 4](#_Toc510991143)

[2 수행 내용 및 중간결과 5](#_Toc510991144)

[2.1 계획서 상의 연구내용 5](#_Toc510991145)

[2.2 수행내용 9](#_Toc510991146)

[3 수정된 연구내용 및 추진 방향 12](#_Toc510991147)

[3.1 수정사항 12](#_Toc510991148)

[4 향후 추진계획 14](#_Toc510991149)

[4.1 향후 계획의 세부 내용 14](#_Toc510991150)

[5 Q & A 16](#_Toc510991151)

# 프로젝트 목표

## 프로젝트 설명



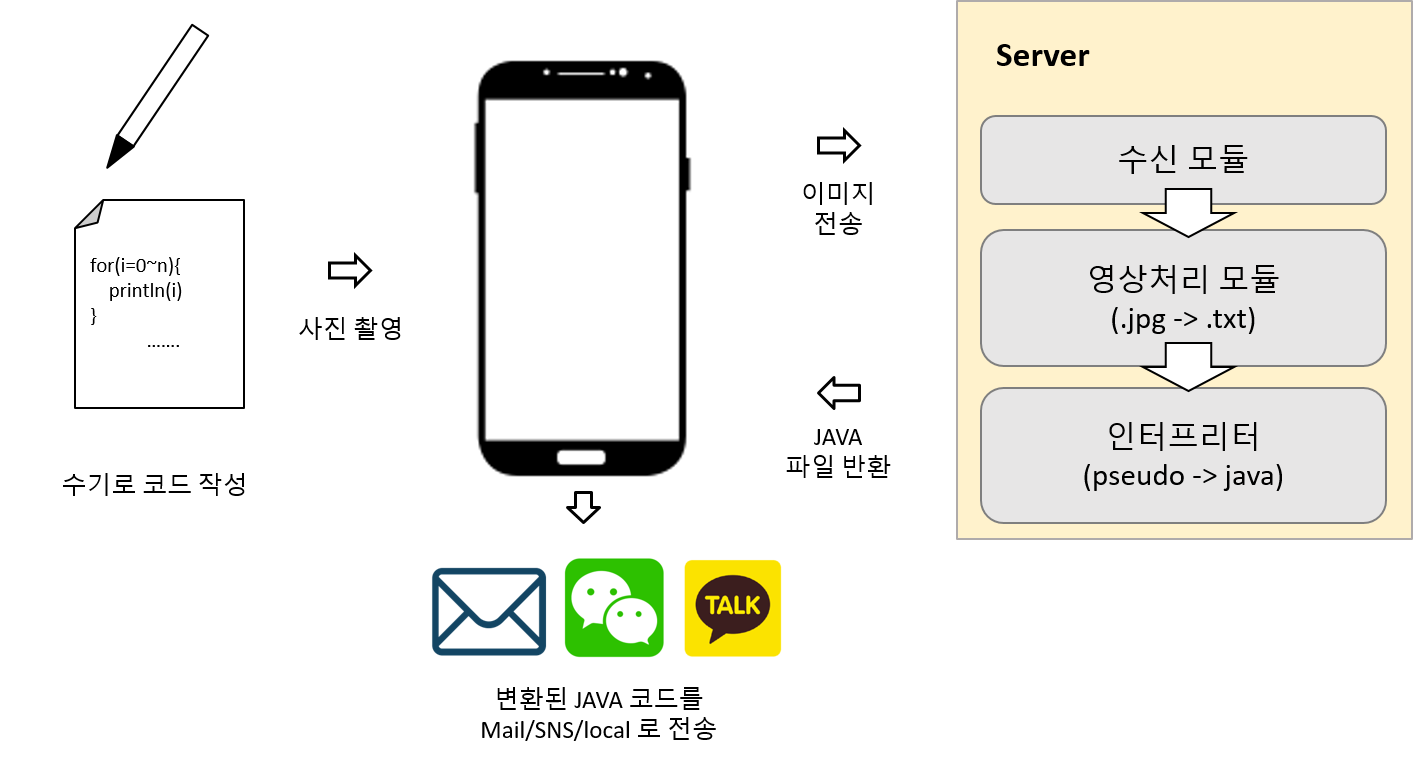
* + 본 프로젝트(이하 Coway)는 **손으로 작성된 수도 코드**를 사진으로 찍어 변환 과정을 통해 텍스트 형태의 **완성된 Java 코드로 변환해주는 Application**

## 프로젝트 추진 배경 및 필요성

1. **코드 작성의 시공간적 제약**
   * 출퇴근, 외출, 통학 시간과 같은 개발 환경에 접근하기 힘든 상황에서 해결해야 할 문제에 대한 아이디어가 떠올라도 아이디어 기록 및 코딩을 하기 힘든 상황이 발생하며 간단한 수도 코드 형식으로 기록이 되더라도 분실 혹은 문법 미 적용 등의 문제로 실제 코드를 작성할 때 활용되지 못하는 일회성 정보로 사라지는 경우 발생
2. **수 많은 언어간 상이한 문법, 높은 진입 장벽**
   * 존재하는 수 많은 언어간 상이한 문법으로 인해 알고리즘 구현, 아이디어 공유와 같은 상황에서 아이디어에 초점을 맞추고 발전시키기도 아까운 시간에 각 언어간 특징적인 문법적인 요소 고려 등의 문제로 더 많은 시간과 노력이 낭비되는 상황이 발생

# 수행 내용 및 중간결과

## 계획서 상의 연구내용



**1.작동 흐름 개요**

**1**. COWAY 어플리케이션을 실행시켜 수기로 작성한 수도 코드를 촬영하거나 혹은 기존에 저장한 사진을 불러와 서버로 전송한다.

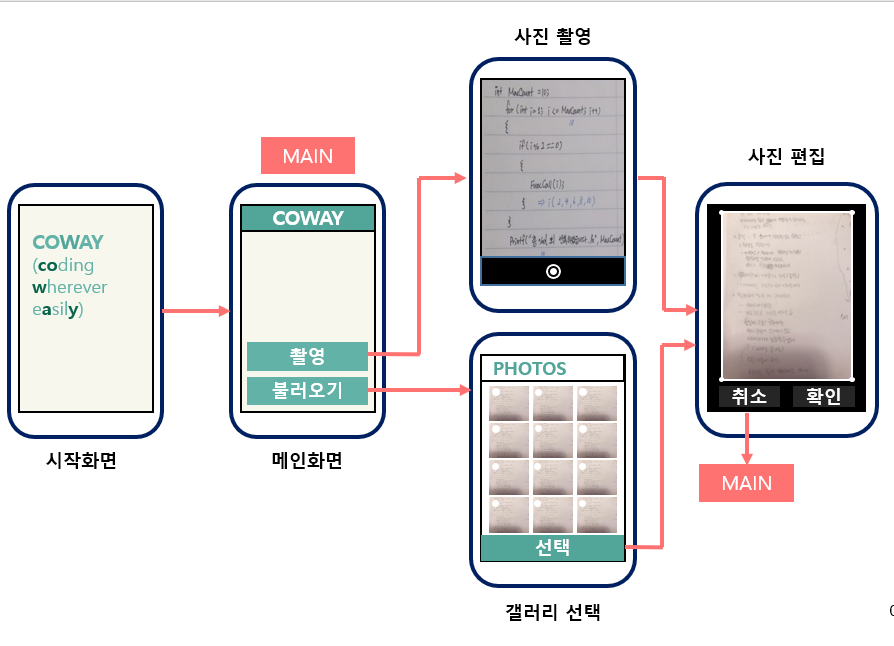
**2.** 서버로 전송된 수도 코드 이미지를 영상처리를 통해 텍스트를 추출한다.

**3.** 추출된 텍스트 수도 코드를 전처리하여 변환에 용이한 형태로 변형한다.

**4.** 전처리가 완료된 텍스트를 변환기를 통해 자바 코드로 완성한다.

**5.** 자바 코드를 사용자의 스마트 폰으로 전송하여 어플을 통해 작성한 수도 코드와 변환이 완료된 자바 코드를 동시에 확인 할 수 있게 한다.

**2.Application (Front)**



사용자가 어플리케이션을 처음 구동하면 간단한 튜토리얼을 진행하여 수도코드의 문법을 스스로 작성하며 사용법을 익힐 수 있게 한다. 처음 구동이 아니라면 사진 촬영 버튼을 눌러 사진을 촬영한 후, 사용자 임의로 사진을 수정하여 촬영을 완료한다. 혹은 갤러리에서 불러오기를 통하여 이전에 작성한 수도 코드 사진을 불러올 수 있다. 이후 확인 버튼을 클릭하여 사진을 서버로 전송한다.

**3.Server (Backend)**

**3.1.텍스트 추출**



서버는 사용자가 전송한 사진을 수신하면, 서버 내부의 텍스트 추출 모듈을 실행한다. 영상 처리 모듈은 내부 텍스트 추출 알고리즘을 활용하며, 기존에 구현한 알고리즘을 라이브러리로 불러와 추출을 실시하여 최종적으로 수도 코드 이미지 파일에서 수도 코드 텍스트 파일을 생성한다

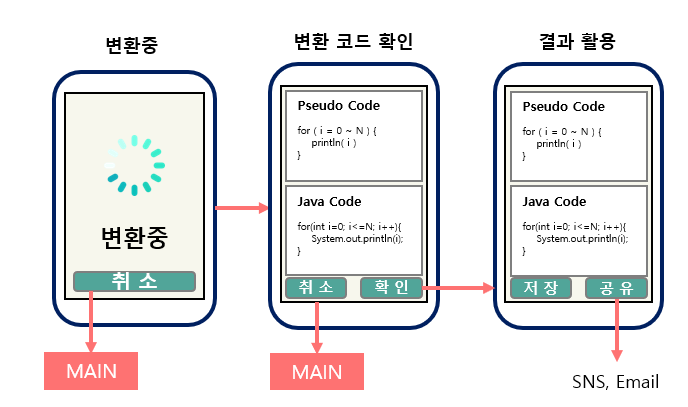
**3.2.수도코드 전처리**

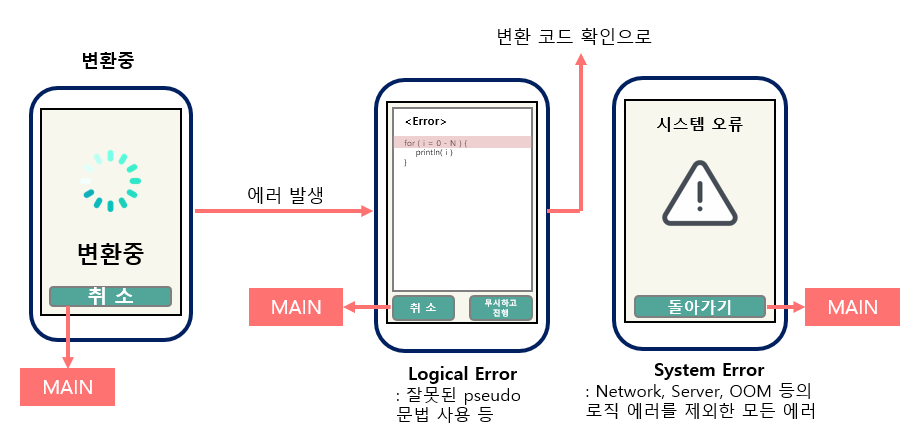
수기로 작성된 수도 코드는 사람의 필기체에 따라 글자의 명확성이 달라질 가능성이 존재하기 때문에 전처리를 통하여 수도 코드를 정제한다. 추출된 텍스트를 자바로 작성된 전처리 프로그램을 실행시켜 라인 바이 라인으로 검사를 실시하고, 수정이 가능한 오류는 변환 프로그램이 인식할 수 있는 수도 코드 문법으로 수정한다. 인식할 수 없는 문법이거나, 혹은 텍스트가 잘못 추출되었을 경우 변환을 하지 않는다

**3.3.수도 -> Java 변환**

전처리를 완료한 수도 코드는 예약어 매칭 방식을 활용한 인터프리터에서 자바 코드로 변환이 완료된다. 인터프리터는 수도 코드를 한문자씩 읽어와 해당 문자와 예약 되어있는 자바 코드 매칭을 확인하여 변환을 실시한다. 이 작업은 라인 단위로 진행이 되며, 만약 매칭되지 않은 예약어가 발견되어 변환에 오류가 발생할 경우 해당 라인에 표시를 하고 변환을 멈춘다. 변환이 성공적으로 완료되면 완성된 자바 코드를 텍스트로 사용자에게 송신하게 되고, 문법적인 오류가 발생할 경우 에러가 발생한 수도 코드 라인을 표시하여 사용자에게 전송한다

**3.4.사용자 수신**





변환이 완료된 자바 코드는 다시 사용자의 앱으로 전송되며 사용자가 작성한 수도 코드와 변환이 완료된 자바 코드를 상하로 배치하여 확인할 수 있다. 사용자는 내용을 확인하여 변환이 정확할 경우 이를 스마트폰에 따로 저장하거나 이메일, SNS등의 경로로 외부로 보낼 수 있다. 변환이 정확하지 않은 경우 다시 1번으로 돌아갈 수 있다. 변환 자체가 이루어지지 않은 경우에도 초기화면으로 돌아갈 수 있으나, 변환 자체를 무시하고 남은 과정을 진행 할 수 있다.

## 수행내용

**1.영상처리**

* + 명암 제거 및 바이너리 이미지 생성 : 히스토그램 균일화 알고리즘과 퓨리에 변환 알고리즘을 활용.
  + 라인별 인식. : 라인 별 시작 캐릭터의 좌표를 기준으로 일정 범위 이내의 좌표들을 모아 라인으로 처리. 리스트 동적 생성을 구현하여 다양한 개수의 라인 처리 가능.
  + 캐릭터 윤곽선 정렬 및 통합 : 라인 별로 분류된 캐릭터의 x 좌표를 기준으로 정렬, 서로 분리된 2개의 구성으로 나누어지는 글자들(i , = , j)의 윤곽선을 통일함.
  + 윤곽선 기준 캐릭터 추출. : 각 윤곽선의 좌표 정보(x,y,w,h)를 활용하여 캐릭터를 분리하여 추출
  + 폰트 텍스트 인식 및 추출 성공. : 테저렉트 기반 수기가 아닌 폰트 텍스트의 인식 및 추출을 성공
  + 실행 스크립트 작성. : Java base의 서버에서 python 파일을 구동할 수 있도록 실행 스크립트를 작성.
  + **계획서의 내용과 동일한 속도로 진행중**

**2. Server**

* + Socket 통신 구현 완료
  + FE (Android) – BE(Coway Server) 간 통신 프로토콜 정립: DTO Packet Class 생성
  + Success, Logical Error, System Error 상황에 대한 통신 시퀸스 구현 완료
  + 내부에서 ImageProcessing 모듈, Interpreter 모듈 호출 구현중
  + **계획서의 내용과 동일한 속도로 진행 중**

1. **Interpreter**
   * for문 변환규칙 구현 완료
     + ex) for(i=0~N) -> for(int i=0;i<=N;i++)
   * print, println 변환규칙 구현 완료
     + ex) println(“Hello World”) -> System.out.println(“Hello World”);
   * 변수 추론 변환 규칙 구현중
     + ex) var = 1 -> int var = 1;
     + 제너릭스 매칭 진행 중
   * 세미콜론 붙이는 경우의 수 연구 진행중
     + ex) e.printStackTrace() -> e.printStackTrace();
   * 변환된 코드 인덴트 정렬 기능 구현중
   * **계획서에 비해 다소 뒤쳐져 있음**
     + 중간에 미처 예상치 못한 구현 내용들이 추가되어 지연중
2. **Android**
   * 권환 처리 및 프로세스에 따른 화면 구성
   * 갤러리 사진 연동 및 카메라, 사진 crop 기능 추가: 목표 환경으로 설정한 android 6.0 이상 기기에서 동작 가능하도록 permission 처리
   * 사진 전송 및 변환 결과 수신을 위한 통신 구현: 서버와 공통적으로 주고 받을 packet 구조를 기반으로 비동기 통신환경 구축
   * 변환 결과에 따른 결과 화면 구현(Success / Logical Error / System Error): 전송 받은 packet의 state에 따라 각기 다른 결과물을 사용자에게 안내.
   * **계획서의 일정과 동일하게 진행중**
3. **Analysis & Test**

* 외부 Pseudo code 수집 및 분석을 통한 인터프리터에서 발생 가능한 문제 도출
* 수도코드에서 단순하게 표현되는 사용자의 입력 부분을 java 문법과 매칭
* 변수 선언 시 변수의 문맥에 맞는 java 변수 타입 자동 매칭
* **계획 외의 일정으로 이후 전체 서비스 test까지 진행될 예정**

# 수정된 연구내용 및 추진 방향

## 수정사항

1. **영상처리**

* 텍스트 추출 방법의 변경 : 기존의 방향이었던 모든 이미지에 대하여 테저렉트를 활용하는 방안의 문제점을 인식하여 변경이 필요함. 테저렉트는 폰트 캐릭터 인식에서 높은 정확도를 보여주나, 수기로 작성된 이미지에서는 작동하지 않아 텍스트 추출이 불가능하였다. 이를 보완하기 위해 구글 비전 api를 활용하여 이미지를 캐릭터 별로 추출하여 인식하는 방향을 시도 중.

1. **Server**

* 영상처리 모듈이 Java API로 구동되는게 아닌, Python API로 동작하게 바뀌어서, 이에 맞춰 Python Script를 호출하는 메서드를 구현해야 함.
* 영상처리 모듈 스크립트가 작성되는 대로 구현할 예정

1. **Interpreter**

* 변수 추론 기능이 새롭게 추가 됨.
  + 1. 이에 맞춰 제너릭스 추론, 기존에 사용한 변수의 재 사용 상황 추론 등의 새로운 기능들에 대한 구현이 필요
    2. 핵심 기능을 위주로 빠르게 구현한 뒤, 보완하는 방식으로 진행할 예정
* 인식 가능한 예약어(수도코드)의 범위 산정이 복잡함
  + 1. 너무 추상적인 수도코드의 인식은 지양하고, 포멀한 문법에 대해서 확실하게 변환해 주는 방향으로 진행 예정

ex) 키보드 입력을 받을 때 readKey() 등의 수도코드 1개를 받아서 변환해 주는 것은 지양

* Server, Interpreter, Android 등에서 공통적으로 발생하는 작업들에 대해 통일시킬 필요가 있었음 (파일 I/O, Packet DTO 이용 등)  
  -> 이를 바탕으로 commons 라이브러리를 새로 생성하여 같이 업데이트 및 배포 중

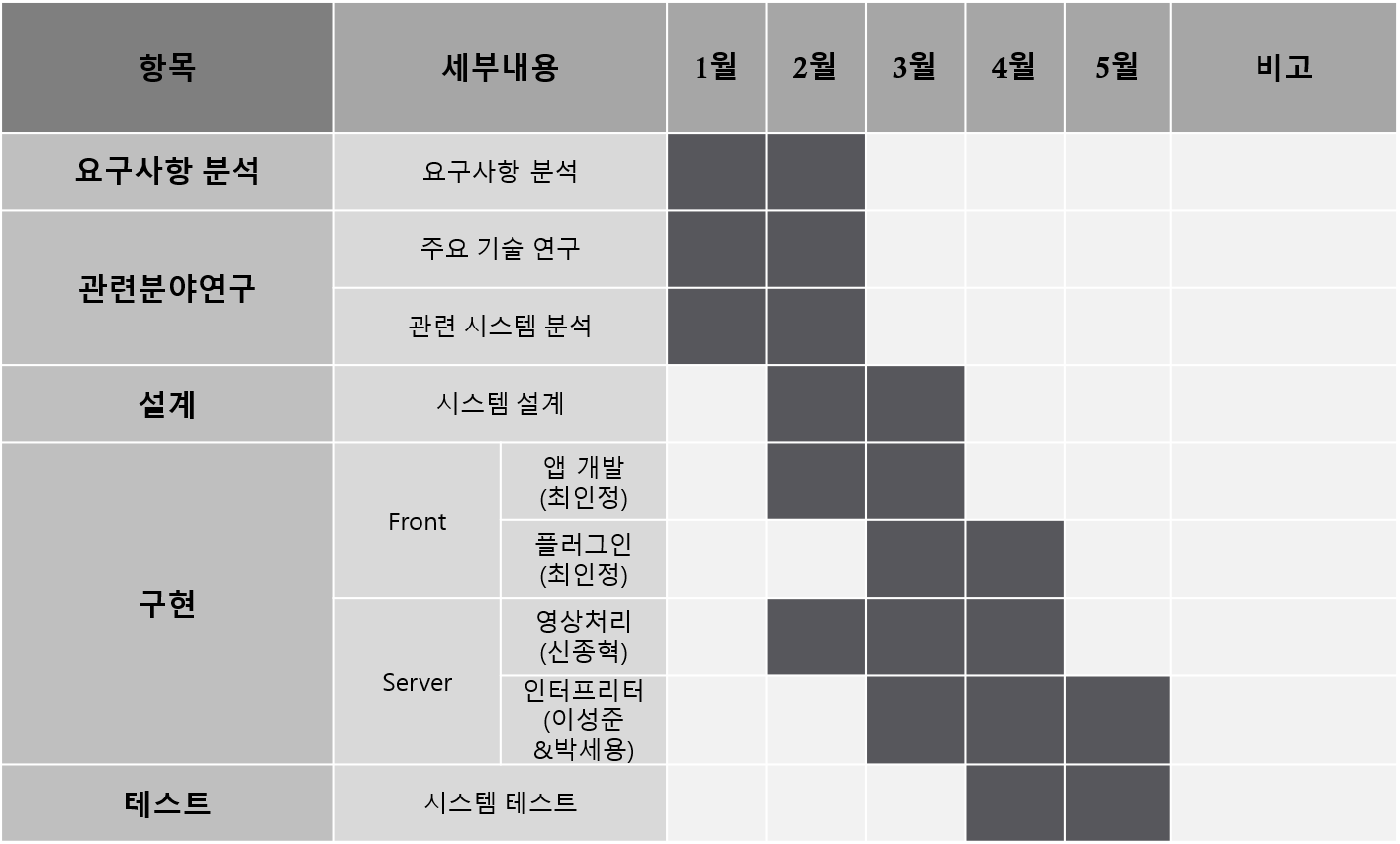
# 향후 추진계획

## 향후 계획의 세부 내용

1. **영상처리**
   * 수기로 작성된 수도 코드 이미지에 대하여 영상처리를 완료한 이후 각 캐릭터 이미지를 추출하여 구글 비전 api로 인식을 시험. 각 캐릭터 이미지의 네이밍을 통해 라인 구별을 실시하려 추출된 텍스트의 형태를 최대한 유지하고자 함.
2. **Server**
   * 영상처리 모듈 호출 및 인터프리터 호출 부분 최종 작성을 진행할 예정
   * FE와의 통신 부분에 있어서 고도화를 진행할 예정
3. **Interpreter**
   * 변수 추론 부분에 대해서 완성도를 높일 예정
   * 현재의 인식가능한 수도코드 외 더 필요한 부분에 대해 연구하고 구현할 예정
   * 제대로 변환이 되지 않을 경우 어떻게 처리해야 할 지에 대해 연구하고 구현할 예정
4. **Android**
   * 코드 변환 성공할 경우의 결과물에 대한 저장 및 전송 기능 추가: local computer / email / 메신저(카카오톡)으로 확장
   * Server와의 통신 중 발생 가능한 오류예외처리 추가
   * 앱 디자인적 요소 추가
   * 앱 사용을 위한 튜토리얼 가이드 제작
5. **Analysis & Test**

* 외부 수도 코드 수집 및 분석: 발생 할 수 있는 수도 코드 알고리즘에 대한 데이터 수집 및 분석으로 변환 실패로 발생할 logical error 최소화
* 수기 데이터 수집 및 분석: 영상 처리의 정확도를 높이기 위해 손글씨 데이터 수집 및 분석으로 영상처리에서 발생할 오류 최소화

## 개발 일정



# Q & A

|  |
| --- |
| 심사의견 or 질문 |
| 평가기준까지 제시. 최종 평가시 이 정량적 평가기준에 맞추어 진행. 텍스트 추출 부분의 정확도를 확인요. |
| 답변 |
| 평가 기준은 저희가 80%이상의 정확도와 5초 내의 사용자에게 결과를 돌려주는 것을 목표로 하고 있다. 80%가 왜 그렇게 정해졌는가를 분류 잘 할 것.  20%에는 영상처리에서 발생하는 오류와 변환 과정간 발생하는 오류를 모두 포함한것.  현재 텍스트 추출을 비전 api로 진행 중인데 이 부분의 정확도 역시 고려가 필요함 |

|  |
| --- |
| 심사의견 or 질문 |
| 코웨이의 아이디어는 좋으나, 결과 예측에 대하여 아직 불명확한 목표를 가지고 있습니다. |
| 답변 |
| 1번과 요지가 동일 |

|  |
| --- |
| 심사의견 or 질문 |
| pseudo code의 형식이 배우기 쉬운지? |
| 답변 |
| 어플리케이션 내부에서 튜토리얼을 제공하여 쉽게 학습할 수 있도록 할 예정입니다.  또한 수도코드 형식 자체가 C나 Java 계열을 배운적이 있다면, 크게 어렵지 않게 익숙해질 수 있을 것입니다. 왜 자바의 수도 코드를 정하였는가를 답변  (패스)예를 들어 제일 많이 사용하는 for 문의 형식은 아래와 같이 표현됩니다.  for(i = 0 ~ N) {     println(i)  } |

|  |
| --- |
| 심사의견 or 질문 |
| 순서도 그림 형태의 pseudo code도 처리 가능한지 궁금합니다. |
| 답변 |
| 순서도의 형식의 수도 코드의 의미를 파악하기 위해서는 엔티티(도형) 내부의 캐릭터의 의미, 화살표 방향의 인식과 순서를 인지해야 하는 것이 필요합니다. 현재까지의 개발 과정에서는 이 부분에 대하여 고려를 하지 않았으며, 당장 기능을 추가하기에도 무리가 있습니다.  저희가 프로젝트를 계획하는 초기 조사 과정에서 순서도를 변환하여 주는 오픈 소스 , 상용 서비스 둘 다 이미 존재하여 개발 방향에서 제외하였습니다. 현재 목표로 하는 프로그램의 구현이 완료된 이후에 오픈 소스를 활용하여 확장할 수 있을 것입니다. |

|  |
| --- |
| 심사의견 or 질문 |
| 전체 프로젝트의 목표는 pseudo code > Java code인데 영상처리를 강조하는 이유가 무엇인가요? |
| 답변 |
| 최종 output인  Java코드로 변환하는 과정이 프로젝트의 주요 목표이긴하나  당연히 사용자로부터 받은 수도코드 이미지로부터 추출한 수도코드 텍스트의  정확도가 기반되어야 올바른 변환 과정 및 높은 변환 정확도가 가능하므로  서로 분리하여 따로 강조할 수 있는 부분이 아닌 진행 과정 중 하나의 단계로서  마찬가지로 중요한 부분이므로 강조했습니다  물론 메모장에 직접 작성하여 전송할 경우 더욱 정확하게 변환이 가능. |

|  |
| --- |
| 심사의견 or 질문 |
| 어떤 알고리즘을 이용해서 pseudo code > Java code를 변환할 예정인가요? |
| 답변 |
| (중복)  관련 링크: <http://me2.do/5ElLeO5t>  자세한 설명은 발표시 진행하도록 하겠습니다. |

|  |
| --- |
| 심사의견 or 질문 |
| 어느정도 복잡한 pseudo code를 다룰 수 있나요? |
| 답변 |
| 수도코드의 복잡성에 대한 명시가 모호(우리가 모호함을 정의하면 됨)   * 한줄에 하나의 문법만을 허용 * 영어 표현식이 아닌 기호 기반의 표현식 허용 * 외부 라이브러리 기능 제외 |

|  |
| --- |
| 심사의견 or 질문 |
| Code Lib 가 Pseudocode mapping 으로 정확하게 mapping 이 될 수 있을지의 기능성이 모호합니다. |
| 답변 |
| import java.util.Arrays; 처럼 패키지 import 관련 질문이라고 이해 하였습니다.  초기 프로젝트 기획 시 라이브러리 import 기능은 외부 라이브러리 연동 등 복잡한 내용이 많아서, 진행하는데 어려움이 있다고 판단하였습니다.  따라서 복잡한 외적 사항들 보다는, 핵심적으로 표현하고자 하는 코드에 대한 정확한 변환에 집중하기로 하였습니다. |